VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

MODELOVÁNÍ A ANALÝZA ŘÍZENÍ SAMOČINNĚ PARKUJÍCÍHO VOZIDLA

MODELING AND ANALYSIS OF SELF-PARKING VEHICLE CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK KRUCINA**
AUTHOR

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

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Krucina Marek**
Program: Informační technologie
Název: **Modelování a analýza řízení samočinně parkujícího vozidla**
Modeling and Analysis of Self-Parking Vehicle Control
Kategorie: Modelování a simulace

Zadání:

1. Zdokumentujte požadavky, problémy, pojmy a principy související s činností samočinně parkujícího vozidla (SPV); proveďte detailní rešerši v této oblasti. Navrhněte vhodnou abstrakci SPV, jeho okolí a aspektů klíčových z hlediska jeho 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 SPV 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, varianty parkovišť, intenzita a druh provozu na parkovišti).
5. Diskutujte a zhodnoťte možnosti vytvořeného modelu z hlediska analýzy dopadu zkoumaných vlivů a navrhněte možné směry pokračování v projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání, představení možných směrů řešení, vytvoření základního výpočetního modelu samočinně parkujícího 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 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Tato bakalářská práce se zabývá samočinně parkujícími vozidly a jejich chováním. Popisuje různé stupně automatizace parkování a s tím spojené technologie. Zabývá se kinematickým modelem vozidla a různými způsoby reprezentace pohybu. Shrnuje základní modelovací nástroje a věnuje se především nástroji UPPAAL SMC, ve kterém je následně vytvořen model samočinně parkujícího vozidla a jeho okolí, jehož implementace je v práci popsána. V závěru práce je vytvořený model podroben analýze pomocí metody statistického ověřování modelu a následně je zkoumán vliv hustoty provozu.

Abstract

This bachelor thesis deals with self-parking vehicles and their behavior. It describes the levels of parking automation and related technologies. It describes the kinematic model of the vehicle and various ways of representing movement. It summarizes the basic modeling tools and focuses mainly on the UPPAAL SMC tool, in which a model of a self-parking vehicle and its surroundings is subsequently created. Its implementation is described in the work. At the end of the work, the created model is analyzed by using the method of statistical model checking of the model and then examines the effect of traffic density.

Klíčová slova

parkovací asistent, automatické parkování, samočinně parkující vozidlo, parkovací senzory, modelování, časové automaty, statistické ověřování modelu, UPPAAL

Keywords

parking assistant, automatic parking, self-parking vehicle, parking sensors, modelling, stochastic timed automata, statistic model checking, UPPAAL

Citace

KRUCINA, Marek. *Modelování a analýza řízení samočinně parkujícího vozidla.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Modelování a analýza řízení samočinně parkujícího vozidla

Prohlášení

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

.....
Marek Krucina
19. května 2021

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Josefu Strnadelovi, Ph.D. za odborné vedení, cenné rady, poskytnuté materiály, trpělivost a pozitivní a klidný přístup nejen na konzultacích po celou dobu řešení této práce.

Obsah

1 Úvod **3**

2 Samočinně parkující vozidla **4**
2.1 Historie parkování 4
2.2 Parkovací senzory 5
2.2.1 Ultrazvukové senzory 5
2.2.2 Elektromagnetické senzory 6
2.3 Úrovně pomoci při parkování 6
2.4 Typy parkovacích míst 8
2.5 Kinematický model vozidla 9
2.6 Reprezentace pohybu 12
2.6.1 Dubinsovy křivky 12
2.6.2 Reeds-Sheppovy křivky 12
2.6.3 Trajektorie pohybu při kolmém parkování 14

3 Modelování a modelovací nástroje **15**
3.1 Základní pojmy 15
3.2 Dostupné modelovací nástroje 16
3.2.1 MATLAB/Simulink 16
3.2.2 Dymola/Modelica 16
3.2.3 SIMLIB 17
3.3 UPPAAL 18
3.3.1 Základní informace 18
3.3.2 Uživatelské rozhraní 18
3.3.3 UPPAAL SMC 20

4 Návrh řešení a popis modelu **22**
4.1 Návrh řešení 22
4.2 Použité abstrakce 23
4.2.1 Abstrakce parkoviště 23
4.2.2 Abstrakce samočinně parkujícího vozidla 24
4.2.3 Abstrakce zatáček 25
4.3 Popis modelu 25
4.3.1 Základní funkce a proměnné 25
4.3.2 Časové automaty 26

5 Zhodnocení řešení **35**
5.1 Základní scénáře a vlastnosti 35

5.1.1 Ověření koncových stavů 35
5.1.2 Parkování na určené místo 37
5.1.3 Chování pohyblivých překážek 39
5.1.4 Změna rychlosti při reakci na okolí 39
5.2 Vliv hustoty provozu 41

6 Závěr **44**

Literatura **45**

A Grafy rozdělení pravděpodobnosti **48**

B Obsah přiloženého CD **49**

Kapitola 1

Úvod

V dnešní době jsou auta běžnou součástí života, každá rodina vlastní alespoň jedno, většinou dokonce více. Neoddělitelnou součástí používaní těchto dopravních prostředků je i jejich parkování. Často to bývá jedna z nejproblematičtějších částí jak v autoškole, tak při řízení v běžném životě. Proto se lidé snažili parkování usnadnit téměř od dob, kdy se samotná auta začala vyrábět a používat. Většina dnes vyráběných aut má parkovací senzory, které řidiči signalizují, kolik místa ještě kolem auta zbývá, plno aut má parkovací kamery, kde může řidič přímo vidět, kam couvá. Nejpokročilejší technologie už zaparkují úplně za vás.

Automatický parkovací systém ovšem nesmí svým jednáním ohrozit okolí. Musí tak při parkování dávat pozor na okolní překážky, kterými jsou převážně chodci, a také na ostatní vozidla. Vozidlo navíc nemusí být autonomní pouze při samotném parkovacím manévru. Je pravděpodobné, že v budoucnu budou auta jezdit sama po celých parkovacích domech a volná parkovací místa i aktivně vyhledávat. Taková vozidla už budou muset disponovat vysokým stupněm autonomnosti.

Tato práce rozebírá v kapitole 2 jednotlivé typy parkování, typy senzorů, model vozidla a jeho pohyb. V kapitole 3 popisuje současné modelovací nástroje a zejména program UPPAAL. Kapitola 4 obsahuje návrh řešení a popis implementace modelu samočinně parkujícího vozidla a jeho okolí. Nakonec je model v kapitole 5 podroben testům na požadované vlastnosti pro ověření modelu a dalším simulacím, které zjišťují vliv hustoty provozu na řízení vozidla.

Kapitola 2

Samočinně parkující vozidla

Za samočinně parkující vozidlo lze označovat vozidlo se zabudovaným automatickým parkovacím systémem, které samostatně dokáže zaparkovat vozidlo z jízdního pruhu na příslušné parkovací místo. Má za cíl zvýšit komfort a bezpečnost při parkování, a to nejen ve stísněných prostorách. Parkovací manévr systém provádí koordinováním rychlosti a úhlu kol, potažmo volantu.

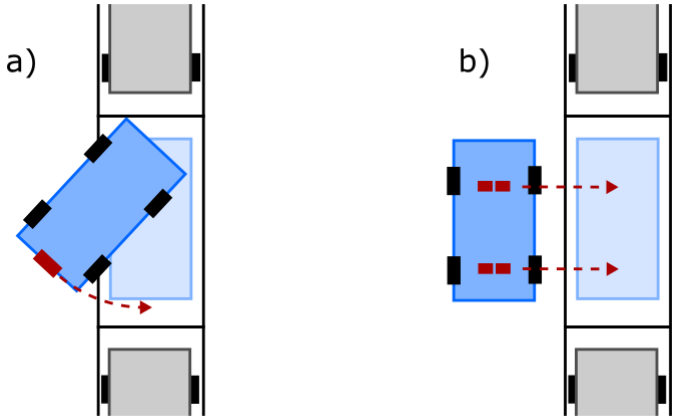
K zorientování se v terénu používá různé druhy senzorů a kamer. Nejčastěji se automaticky parkuje podélně, protože je to obecně považováno za nejtěžší druh parkování, pokročilejší systémy by ale měly zvládat i parkování kolmé, šikmé nebo dokonce nalezení volného parkovacího místa a zaparkování v celém parkovacím domě.

2.1 Historie parkování

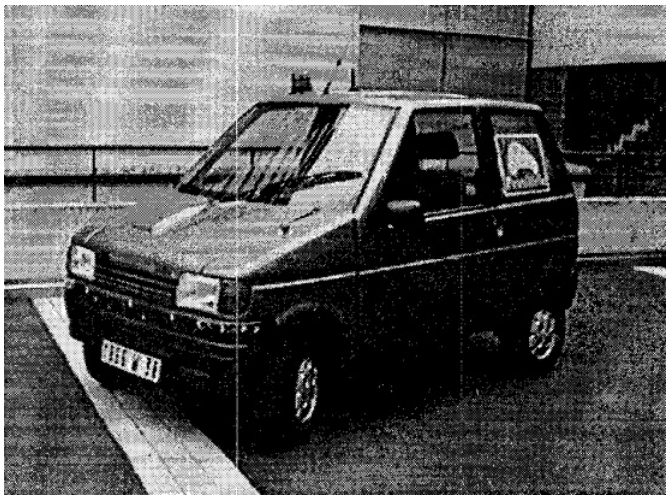
První pokusy usnadnit řidičům parkování sahají až do 30. let 20. století. Nejstarší známý model je patrně auto značky Packard z roku 1933, kdy využili páté rezervní kolo jako kolo parkovací. Samotný parkovací manévr začínal najetím předku auta do podélného parkovacího místa, poté se vysunulo páté kolo, které zároveň nadzvedlo zadní nápravu. To umožňovalo pootočit zadní část auta z vozovky na parkovací místo způsobem naznačeným na obrázku 2.1 a). Tento princip popisuje Brooks Walker ve svém patentu podaném roku 1932 [21].

Koncept dalšího z prvních parkovacích asistentů lze nalézt v časopise Popular Science Monthly z roku 1934 [8]. Poté, co se vozidlo zastavilo vedle podélného parkovacího místa, stisknutím tlačítka vyjely z podvozku čtyři zvedáky s kolečky, které celé auto nadzvedly. Následně se mohlo celé vozidlo posunout směrem na vybrané místo, jak je znázorněno na obrázku 2.1 b). Walker dál pracoval na svém systému, roku 1953 představil parkovací systém na sedanu Packard Cavalier [11], nezaujal ovšem žádné investory, proto se tyto mechanické parkovací systémy nikdy ve větší míře neuchytily.

První návrhy automatických parkovacích systémů spadají do poloviny 80. let, kdy M. Sugeno a K. Murrakami představili fuzzy algoritmus pro kolmé parkování do garáže [30]. Roku 1996 Paromtchik s Laughierem vytvořili iterativní algoritmus pro automatické paralelní parkování, který zkoušeli na elektrickém autě Ligier [24]. Na této technologii staví dnes většina světových automobilek s jejich parkovacími systémy.



Obrázek 2.1: Schéma dvou historických parkovacích manévrů



Obrázek 2.2: Ligier auto [24]

2.2 Parkovací senzory

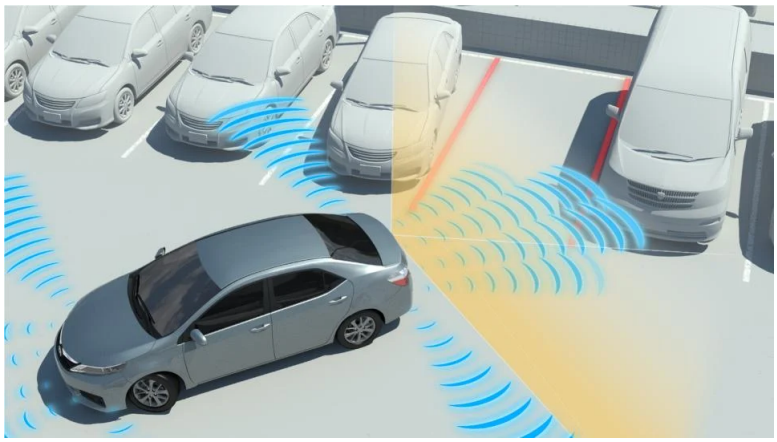
Parkovací senzory pomáhají parkovat bez nehod. Následuje souhrn používaných typů a jejich odlišností. [13]

2.2.1 Ultrazvukové senzory

Ultrazvukové parkovací senzory bývají zabudované především na předních a zadních nárazníčích, kde jsou na každé straně rovnoměrně rozmístěny po celé délce, většinou po 4 kusech. Vzdálenost, na kterou jsou schopny detekovat překážky závisí na přesném modelu a použité frekvenci. Většina parkovacích senzorů dokáže detekovat překážku od 15-30 cm do vzdálenosti 2 až 5,5 metrů [6].

Vysílají zvukové vlny, které se odráží od překážek a vrací se zpátky. Systém je umí zachytit a následně spočítat vzdálenost od dané překážky. Tu dává následně najevo zvukovým projevem, který informuje řidiče vozidla. Zpravidla čím rychlejší je frekvence pípání, tím blíž objekt je, při maximálním přiblížení bývá pak výstražný tón nepřetržitý.

Někdy je parkovací systém i optický, překážky jsou pak vizualizovány na displeji na palubní desce. Uprostřed obrazovky bývá vyobrazeno vozidlo a jeho okolí je pak vybarveno podle toho, jak daleko je překážka.



Obrázek 2.3: Parkovací senzory [13]

Nevýhody ultrazvukových senzorů

Nevýhodou ultrazvukových senzorů je to, že ne všechny předměty a potenciální překážky odrážejí zvuk dostatečně dobře. Například úzké nebo velmi malé předměty nemusí být vůbec zachyceny. Ploché předměty, které mají odrazovou plochu pod úhlem, mohou odrážet ve špatném směru a překážky z měkkého materiálu, který dobře absorbuje zvuk, mohou vlnu pohltit tak, že je pak odraz příliš slabý.

Špatné povětrnostní podmínky jako sníh nebo déšť také ruší správnou činnost senzorů, stejně jako špína nebo bláto na nárazníčích v místě senzorů - zvuková vlna přes ně nemusí projít až k opravdové překážce.

Nevýhodou pro některé je i to, že tyto senzory musí být navrtány přímo do nárazníků, aby fungovaly, takže může být narušena estetičnost vozidla.

2.2.2 Elektromagnetické senzory

Senzory vysílají elektromagnetické vlny, které dokáží detekovat předměty nezávisle na jejich tvaru nebo materiálu. Rozpoznat překážku dokážou zpravidla už od vzdálenosti kolem 10 cm zhruba do 80 cm [4], v závislosti na konkrétním modelu. Snadno se instalují, nemusí se do vozidla nijak vrtat, stačí je pouze přilepit na nárazník zevnitř. Ve výsledku tedy nejsou zvenku vidět, na rozdíl od ultrazvukových senzorů.

Nevýhody elektromagnetických senzorů

Elektromagnetické senzory pro detekování potřebují, aby se vozidlo pomalu kontinuálně pohybovalo. Pokud tedy řidič při parkování často zastavuje, senzory překážky nemusí vůbec detekovat.

2.3 Úrovně pomoci při parkování

Stupně pomoci při parkování se dají rozdělit do čtyř úrovní [16].

Parkovací senzory

Senzory jsou nejběžnější a nejjednodušší pomocí při parkování. Prvním masově vyráběným modelem s parkovacími senzory byla Toyota Prius, která byla vyrobena již v roce 2003. V dnešní době je mají takřka všechna nově vyráběná vozidla.

Obrazové kamery

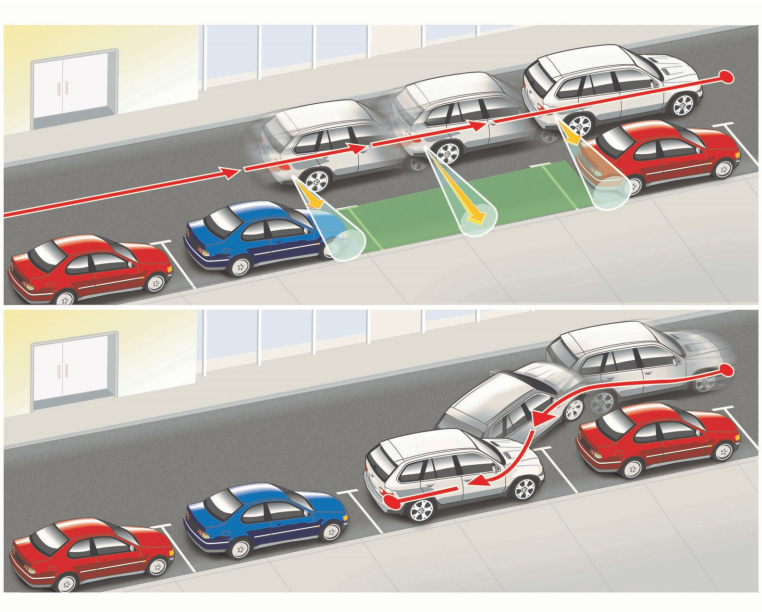
Pro lepší přehled při couvání na parkovací místo slouží zadní obrazové kamery. Snímají celý prostor, kam řidič při parkování najíždí a zobrazuje ho na displej palubní desky. Obyčejně se kamery kombinují se senzory pro zajištění co nejlepšího přehledu.

Některá vozidla mají kamer hned několik, zpravidla čtyři, a dohromady tvoří 360° pohled z ptačí perspektivy. Většinou nabízí i možnost přepínání mezi jednotlivými kamerami, pokud chce řidič sledovat pouze určitý směr.

Parkovací asistent

Když má auto systém parkovacího asistenta, znamená to, že má nahraný software, který umí zaparkovat automaticky. Vozidlo mívá zabudované boční senzory, kterými jako první hledá volné parkovací místo. Pokud ho najde, samo dokáže na vybrané místo zaparkovat. Vše hlídá pomocí senzorů a kamer. Mělo by dokázat reagovat i na náhlé nebezpečí kolize, například kolemjdoucího. Některá vozidla jsou schopna také vyparkovat zpátky na silnici, obvykle z podélného parkování.

U většiny aut ale pořád zůstává řidiči možnost převzít zpět řízení, i třeba v půlce automatického manévru.



Obrázek 2.4: Ukázka fungování parkovacího asistenta [34]

Plně autonomní parkování

V dnešní době už existují automatizované parkovací domy, kde stačí nechat auto na začátku a vystoupit. Vozidlo pak samo dokáže najít a zaparkovat na nějaké volné parkovací místo v daném parkovacím domě. Pokud chce člověk následně odjet, vozidlo si může nechat zavolat například pomocí mobilní aplikace. [2]

2.4 Typy parkovacích míst

V závislosti na prostředí, kde je potřeba parkovat se vyvinuly různé typy parkovacích míst. Nejčastější jsou místa pro kolmé, šikmé a podélné parkování.

Kolmé parkování

Při kolmém parkování stojí auta bok po boku, většinou kolmo ke zdi nebo chodníku. Tento typ parkování umožňuje zaparkovat nejvíce aut na délku silnice ze všech ostatních typů. Nevýhodou je naopak potřeba většího prostoru do šířky kvůli délce vozidla a v neposlední řadě také kvůli manévrování. Nejvíce se tak používá na parkovištích nebo v parkovacích domech.

Na parkovištích se ve většině případů vyskytují dvě řady parkovacích míst naproti sobě, které dělí volné uličky. Řidič může parkovat předem nebo zadem. Pokud po zaparkování předem na protilehlém parkovacím místě neblokuje jiné auto, lze dokonce opět vyjet předem bez couvání.

V České republice jsou častým problémem příliš úzká parkovací místa, protože stávající vyhláška [10] stanovuje minimální šířku pouze 2,5 metru, přičemž počítá s průměrnou šířkou vozidla 1,75 metru. V dnešní době ovšem mívají osobní auta i se zrcátky šířku přes 2 metry [14]. Majitelům parkovišť sice podle normy nic nebrání v tom nechat udělat parkovací místa širší, většinou je ale jejich prioritou počet míst, obvzlášť jsou-li parkoviště placená.

Šikmé parkování

Šikmá parkovací místa se navrhují zejména v úhlech 45° a 60°. Je podobné kolmému parkování, ale je jednodušší na manévrování, protože se vozidlo nevytáčí o celých 90°, ale jen o polovinu či dvě třetiny.

Díky menší potřebě místa na šířku cesty lze šikmá parkovací místa vidět podél silnic ve městech, kde by se auta na kolmo nevešla, nebo na parkovištích. V takových případech je pak parkování povoleno v předem stanoveném směru, většinou najetím předkem vozidla. Při couvání za účelem vyjetí z místa podél silnice je ale nebezpečí sražení například cyklisty, protože se k autu může přiblížit v mrtvém úhlu. Z tohoto důvodu jsou v některých případech používána reverzní šikmá parkování, kam musí řidič zacouvat.

Podélné parkování

Podélné parkování je považováno za nejtěžší druh parkování, zejména pro začátečníky, kdy se řidič snaží umístit vozidlo do řady mezi ostatní. Nejčastěji tak bývá podél silnic ve městech. Správný postup začíná najetím až za vybrané prázdné parkovací místo na úroveň následujícího a pokračuje zacouváním na určené místo.

Prostor pro parkování bývá vyznačen čarou, která ho odděluje od silnice, v mnoha případech se už ale neoddělují samotná místa od sebe. Řidič pak musí pouze odhadovat,

jestli se dokáže do místa vejít. Existuje ale rovnice, pomocí které lze spočítat, kolik nejméně místa na délku určité vozidlo potřebuje [9]. Výsledkem je samotná délka auta plus

$$\sqrt{(r^2 - l^2) + (l + k)^2} - (\sqrt{r^2 - l^2} - w)^2 - l - k \tag{2.1}$$

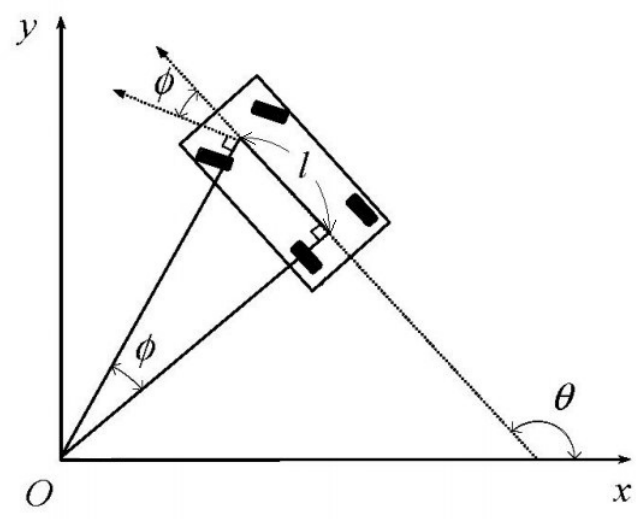
kde:

r = poloměr otáčení auta
l = rozvor kol, tedy vzdálenost mezi předními a zadními koly
k = převis přední karoserie, tedy vzdálenost od předních kol k přednímu nárazníku
w = šířka auta, za které se parkuje

Mohlo by se zdát, že by bylo jednodušší na místo necouvat a místo toho najíždět na parkovací místo předkem auta. To je ale většinou možné jen pokud je k dispozici mnohem více místa, a to minimálně o velikosti dvou prázdných parkovacích míst za sebou. Platí to u vozidel, které mají řízení pouze předních kol, a těch je většina. Je tomu tak, protože při parkování popředu zadní kola pouze následují samotný pohyb vozidla a je tak mnohem náročnější je dostat na místo až po předních kolech, které lze řízeně natáčet. Naopak při postupu, kdy se nejprve couvá a zadní kola jsou najeta do parkovacího místa jako první, je pak snažší a méně prostorově náročné navést zbytek auta, tedy předek, na parkovací místo, protože s přední částí lze manévrovat mnohem ostřeji. [17]

2.5 Kinematický model vozidla

Základní kinematický model vozidla je zobrazen na obrázku 2.5. Jedná se o tzv. bicycle model, který se využívá jako výchozí model pro simulaci běžného automobilu.



Obrázek 2.5: Kinematický model vozidla

Lze popsat pomocí rovnice (za použití Newtonovy notace derivace, zde podle času):

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v \cos\theta \\ v \sin\theta \\ \frac{v}{l} \, tan\, \phi \end{pmatrix} \tag{2.2}$$

kde:

v = lineární rychlost vozidla
l = rozvor kol
ϕ = úhel natočení kol
θ = úhel orientace auta

Pomocí úhlu natočení kol ϕ a rozvoru kol l lze spočítat také poloměr otáčení r.

$$r = \frac{l}{\tan \phi} \tag{2.3}$$

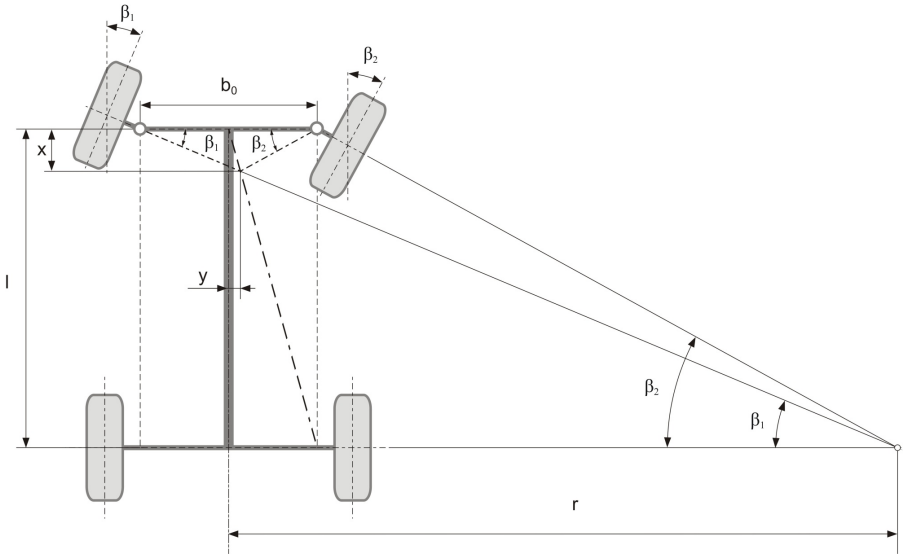
U většiny aut se v dokumentech uvádí ovšem spíše průměry otáčení. Pro uživatele je to totiž lépe zpracovatelný parametr – ví tak, kolik prostoru potřebují, aby se dokázali najednou s autem úplně otočit. Často se v článcích uvádí pojem poloměr otáčení ovšem s hodnotou průměru a dochází tak k dezinterpretaci.

U dnešních aut se průměry otáčení pohybují průměrně mezi 9,5 až 11 metry [5]. Pokud známe jejich rozvor kol, lze jednoduše spočítat jaký je maximální úhel natočení kol.

Například Volkswagen Golf má rozvor kol l = 2,637 m, a průměr otáčení 10,9 m [7]. Průměr je třeba podělit dvěma a ještě odečíst šířku vozidla, protože maximální úhel natočení bude mít vnitřní kolo. Po dosazení do vzorce nám vyjde maximální úhel natočení zhruba 36°, což je průměrná hodnota. U některých speciálně konstruovaných aut může hodnota dosahovat například až 53° [3].

Ackermannova geometrie řízení

V Ackermannově geometrii řízení se navíc rozlišuje vnitřní a vnější kolo při průjezdu zatáčkou. Kola totiž opisují část kružnic o dvou různých poloměrech. Vnitřní kolo je středu otáčení blíže a má tedy menší poloměr, u vnějšího kola je to naopak. Zároveň také musí platit Ackermannova podmínka – ta je splněna, pokud ke středu otáčení míří nejen osy předních kol, ale i osy obou kol zadní nápravy.



Obrázek 2.6: Ackermannova geometrie řízení [29]

Na obrázku 2.6 lze vidět pohled shora na model vozidla splňujícího Ackermannovu podmínku. Pojmenujeme-li si veličiny jako na obrázku, tedy:

l = rozvor kol
b_0 = rozchod kol
r = teoretický poloměr zatáčení
β_1 = úhel natočení vnějšího kola
β_2 = úhel natočení vnitřního kola

vyplyne nám z obrázku následující vztah, který je matematickým vyjádřením podmínky:

$$\operatorname{cotg} \beta_1 - \operatorname{cotg} \beta_2 = \frac{b_0}{l} \tag{2.4}$$

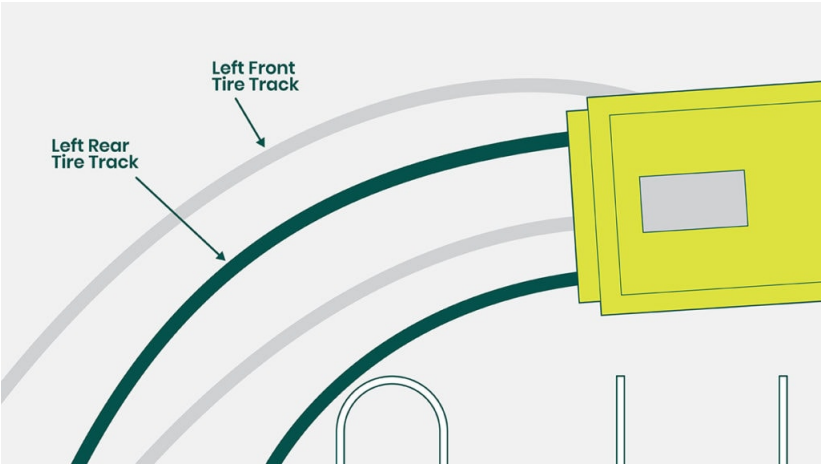
V praxi platí Ackermannova podmínka pouze pro tuhá a nepoddajná kola a pro malé rychlosti, kterých dosahujeme například při parkování. V ostatních případech, hlavně tedy při vyšších rychlostech, vznikají na všech kolech směrové úchylky, které jsou vyvolané především vlivem odstředivé síly. Ty posouvají skutečný střed otáčení více dopředu mimo teoretický střed, který je v rovině se zadními koly. [29]

Ujetá vzdálenost jednotlivých kol

S rozličným poloměrem otáčení kol souvisí také ujetá vzdálenost jednotlivých kol. Vnější kola ujedou větší vzdálenost než kola přední, rozdíl je ovšem i mezi přední a zadní nápravou, ačkoliv méně znatelný.

Pro naše potřeby zde budeme uvažovat pouze vozidla s řízením přední nápravy. Ostatní typy jako řízení zadní nápravy, které se používá například u vysokozdvižných vozíků, nebo systém 4WS, který může být u některých autobusů či aut ve vyšší cenové relaci, nebudeme brát v potaz.

Zadní kolo ujede menší vzdálenost než přední hlavně proto, že přední náprava je řízena, tedy je jí aktivně zatáčeno, zatímco zadní náprava se pouze snaží následovat danou trajektorii vozidla. Nejede ale po úplně stejné dráze jako přední kola. Zadní kola vždy směřují směrem k předním, během zatáčení je tak jejich dráha posunuta víc ke středu otáčení vzhledem k předním kolům, jak lze vidět na obrázku 2.9. [12]



Obrázek 2.7: Trajektorie jednotlivých kol v pravotočivé zatáčce [28]

Holonomní a neholonomoní systém

Auto a ostatní vozidla jemu podobné konstrukce jsou neholonomní systémy. Holonomní systém je takový, který má řiditelný stejný nebo větší počet stupňů volnosti než je celkový počet generalizovaných souřadnic nutných k popisu polohy systému. [22] Například při pohybu v rovině má těleso dva stupně volnosti, posun podél osy x a podély osy y. Aby bylo vozidlo holonomní, musí se umět pohybovat přímo minimálně v těchto dvou osách.

Automobil je schopen se v ploše pohybovat pouze v jedné ose, tedy dopředu a dozadu. Má tak řiditelný menší počet stupňů volnosti, než je počet souřadnic potřebných k určení polohy. Auto je tak neholonomní systém.

Kdyby bylo systémem holonomním a umělo by se přímo pohybovat v obou osách, nebyl by problém například s podélným parkování - autu by stačilo zastavit vedle parkovacího místa a do parkovacího místa se nasunout podobně, jako to v historii některá auta zkoušela viz obrázek 2.1 b). Pro jednoduché kolmé parkování by stačilo holonomní vozidlo, které se umí pohybovat v jedné ose a zároveň se umí otáčet kolem vlastní osy, což je další stupeň volnosti. V tomto případě tak vozidlo zastaví před parkovacím místem, otočí se zpravidla o 90° a na místo najede směrem dopředu či dozadu.

Jak už ale bylo zmíněno, automobil je systémem neholonomním, k parkování a pohybu v zatáčkách tedy potřebuje složitější postupy - musí se pohybovat po křivkách.

2.6 Reprezentace pohybu

2.6.1 Dubinsovy křivky

Dubinsovy křivky reprezentují nejkraší cestu, která spojuje dva body v ploše. Zároveň je daná počáteční i koncová orientace vozidla v daných bodech. Popisuje chování vozidel, která jezdí pouze dopředu, křivka se tak skládá ze tří základních pohybů, a to: zatočit doprava (R), zatočit doleva (L), jet rovně (S). Systém lze popsat rovnicí:

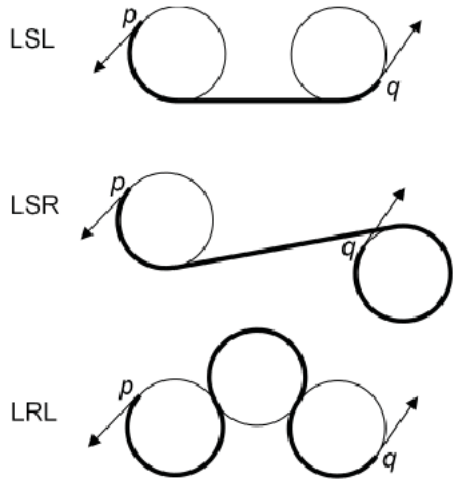
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ u \end{pmatrix} \tag{2.5}$$

kde u je míra zatáčení, $u \in \{-\tan\phi_{max}, 0, \tan\phi_{max}\}$, což odpovídá hodnotám při maximálním zatočení doprava, jízdě rovně a maxinálním zatočení doleva, z čehož plyne, že se neuvažují stavy o menším než maximálním natočení kol, které by trasu zbytečně prodlužovaly. Jelikož bude křivka začínat vždy zatočením a zároveň nebudou nikdy dva stejné pohyby za sebou, protože by stejně splynuly v jeden pohyb, řešením bude vždy jedna z těchto 6 sekvencí: RSR, RSL, LSR, LSL, RLR a LRL. [23]

2.6.2 Reeds-Sheppovy křivky

Reeds a Shepp rozšířili Dubinsův model o možnost couvat, lze tak reprezentovat klasický automobil, a to zejména při parkování, kde je často couvání nezbytné. Systém lze popsat rozšířením rovnice 2.5, a sice následujícím způsobem:

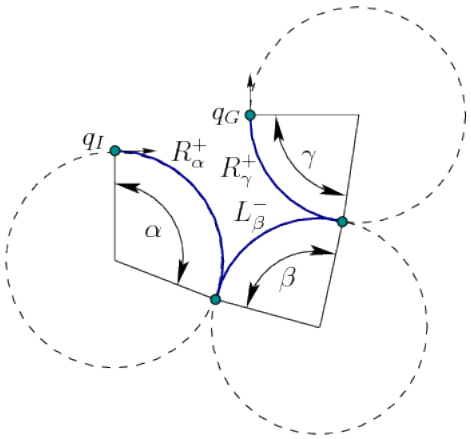
$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} u_1\cos\theta \\ u_1\sin\theta \\ u_1\,u_2 \end{pmatrix} \tag{2.6}$$



Obrázek 2.8: Příklady Dubinsových křivek z bodu p do bodu q [20]

kde $u_1 \in \{-1, 1\}$ a u_2 odpovídá u v rovnici 2.5.Veličina u_1 reprezentuje směr pohybu, dopředu ($u_1 = 1$) nebo dozadu ($u_1 = -1$). [23]

Také lehce pozměnili a rozšířili značení základních slov: otočení směru pohybu (symbol |), jet rovně (S), zatočit doprava nebo doleva (C). Zde může být několik C za sebou, při každém dalším použití ale bude opačný směr, tedy CC značí buď LR nebo RL. U zatočení je navíc specifikován přesný úhel, o který se vozidlo v průběhu otočí, jako dolní index. $C_{\pi/2}$ je tedy křivka s otočením o 90°.



Obrázek 2.9: Příklad $C|C|C$ jinak také $R^+_\alpha L^-_\beta R^+_\gamma$ Reeds-Sheppovy křivky [23]

Ve svém článku [27] také Reeds se Sheppem dokázali, že stačí pouze 48 různých sekvencí řízení vycházejících ze základních slov pro namodelování nejkratší trajektorie mezi dvěma orientovanými body. Sussmann a Tang následně ještě dokázali zredukovat celkový počet sekvencí z 48 na 46 [31]. Nejkratší cestu lze reprezentovat vždy jednou z následujících sekvencí základních slov:

$$\{C|C|C,\ \ CC|C,\ \ C|CC,\ \ CSC,\ \ CC_\beta|C_\beta C,\ \ C|C_\beta C_\beta|C,$$
$$C|C_{\pi/2}SC,\ \ CSC_{\pi/2}|C\ \ C|C_{\pi/2}SC_{\pi/2}|C\}.$$

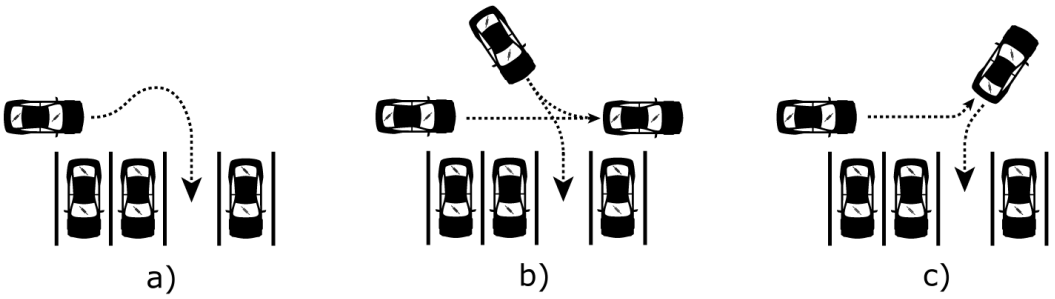
Kdybychom analogicky zapsali základní slova Dubinsova modelu, dostali bychom pouze dvojici $\{CCC,\ CSC\}$. Lze tak vidět, že přidáním možnosti jízdy pozpátku model značně nabobtnal.

2.6.3 Trajektorie pohybu při kolmém parkování

Při kolmém parkování si lze vybrat z několika typů zaparkování, jak lze vidět na obrázku 2.10. Při automatickém parkování je potřeba parkovací místo prvně detekovat pomocí senzorů. Vozidlo tak musí prázdné místo přejet, aby ho rozpoznalo. V takovém případě nemůžeme použít manévr, který je na obrázku vyobrazen jako první. Jediný případ, kdy se dá tento manévr použít, je za předpokladu, že o volném místě víme dopředu a nemusíme ho tak rozpoznávat senzory.

I při automatickém parkování tak je možnost vybrat si, zda chce parkovat popředu či pozadu. Pokud by se parkovalo jako v části b), celý pohyb by se skl8dal ze 3 částí: Pohyb dopředu, couvání doleva a zatočení doprava dopředu na dané místo. Pokud bychom zapsali manévr notací Reed-Sheppových křivek, dostali bychom $S|C|C$ jinak také $S^{+}L_{\beta}^{-}R_{\gamma}^{+}$. V případě vyparkování pak stačí zopakovat poslední dvě části pohybu obráceně.

Co se týče couvání na volné parkovací místo, lze použít mánevr vyznačený v části c), nebo dokonce upravenou verzi, kdy autu stačí pouze najet rovně za místo a rovnou z pruhu pak zacouvat do volného prostoru. V tomto případě je třeba být ale mnohem obezřetnější na okolní překážky, zde konkrétně na ty, které jsou vedle parkovacího místa, což jsou většinou ostatní zaparkovaná auta. Aby auto při couvání nenarazilo bokem do okolní překážky, je zapotřebí zvolit správný počáteční bod couvání. Čím menší máme maximální úhel natočení kol, tím dál bychom měli od řady parkovacích míst být.



Obrázek 2.10: Typy trajektorií vozidla při kolmém parkování [15]

Kapitola 3

Modelování a modelovací nástroje

Tato kapitola uvádí základní pojmy, které se týkají modelování a simulace a dále uvádí některé simulační nástroje a programy, pomocí kterých se systémy dají modelovat a simulovat.

3.1 Základní pojmy

Systém je soubor elementárních částí (prvků systému), které mají mezi sebou určité vazby (propojení prvků). Systémy je možné rozdělit podle existence na:

- reálné (existující) systémy
- nereálné (fiktivní, ještě neexistující) systémy – používají se například v počítačových hrách

nebo podle změn stavu na:

- statické systémy – nemění svůj stav v čase
- dynamické systémy – mění stav v čase

Model je napodobenina systému jiným systémem, většinou počítačovým programem. Model systému musí zahrnovat všechny podstatné vlastnosti systému.

Modelování je proces vytváření modelů systémů na základě našich znalostí.

Abstraktní model se vytváří jako první, zahrnují se do něj vlastnosti modelovaného systému, které jsou pro naše potřeby podstatné. Dochází tak ke zjednodušování, protože se zanedbávají některé detaily systému. Může mít podobu například matematických rovnic.

Simulační model se poté vytváří na základě abstraktního modelu a zahrnuje všechny jeho vlastnosti. Simulační model už je spustitelný program, ve kterém můžeme provádět experimenty.

Simulace je metoda získávání nových znalostí o systému experimentováním s jeho modelem. V praxi se obvykle simulační experimenty opakují vícekrát a s různými parametry.

Verifikace modelu ověřuje izomorfní vztah mezi abstraktním a simulačním modelem. Izomorfní vztah znamená vztah 1:1, tedy už nedochází k dalšímu zjednodušování.

Validace modelu (ověřování platnosti) je proces, v němž se snažíme dokázat, že skutečně pracujeme s modelem adekvátním modelovanému systému. Nelze absolutně dokázat přesnost modelu, vždy je snaha pouze o co největší přiblížení skutečnosti. [26]

3.2 Dostupné modelovací nástroje

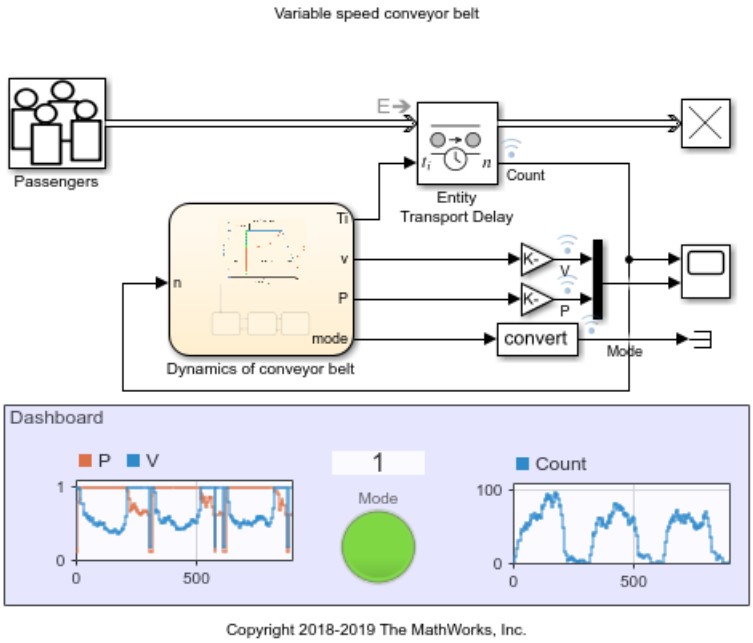
Pro vytváření modelů existují speciální programy, ve kterých se většinou s modelem následně dá i experimentovat. V této kapitole je několik takovýchto programů blíže popsáno.

3.2.1 MATLAB/Simulink

Simulink je v praxi velmi často používaný komerční systém vyvíjený společností MathWorks pro modelování a simulaci dynamických systémů. Je založený na MATLABu, což je programové prostředí a také skriptovací programovací jazyk vytvořený pro matematické účely, zejména pro práci s maticemi [26].

Simulink umožňuje uživateli snadno a rychle vytvářet modely dynamických soustav ve formě blokových schémat. Model se tak sestavuje z bloků, které reprezentují prvky reálných systémů. Bloky poté program sám automaticky převádí na kód. Spojité modely je možné popsat v maticovém tvaru [33].

Na obrázku 3.1 je možno vidět model kyberfyzikálního systému, kde se za pomocí výše zmíněných blokových schémat popisuje chování systému dopravníkového pásu s proměnnou rychlostí[1].



Obrázek 3.1: Model dopravníkového pásu s proměnnou rychlostí v Simulinku

3.2.2 Dymola/Modelica

Název Dymola je akronymní zkratkou z anglického Dynamic Modeling Laboratory, jde o komerční prostředí používající se především k modelování a simulaci fyzikálních systémů, jako jsou například elektrické obvody, mechanické systémy nebo vedení tepla.

[1] Celý příklad viz https://www.mathworks.com/help/simulink/slref/modeling-discrete-message-transport-systems.html.

K modelování se používá objektově orientovaný jazyk Modelica. Snadno se jím popisují spojité systémy pomocí rovnic, převod rovnice na blokové schéma umí jazyk dělat automaticky.

Existují také „open source“ systémy jako OpenModelica a Modelica Standard Library, což jsou volně dostupná prostředí pro jazyk Modelica [32][26].

3.2.3 SIMLIB

SIMLIB je simulační knihovna vyvíjena na FIT VUT od roku 1991 zejména doktorem Petrem Peringerem. Je určena pro modelování a simulaci v jazyce C++ a je možno s ní vytvářet modely diskrétní, spojité i kombinované. Obsahuje několik tříd pro sběr statistických dat a informací o chování modelu při simulaci, 3D modelování pro spojité systémy nebo také rozšíření pro modelování fuzzy systémů.

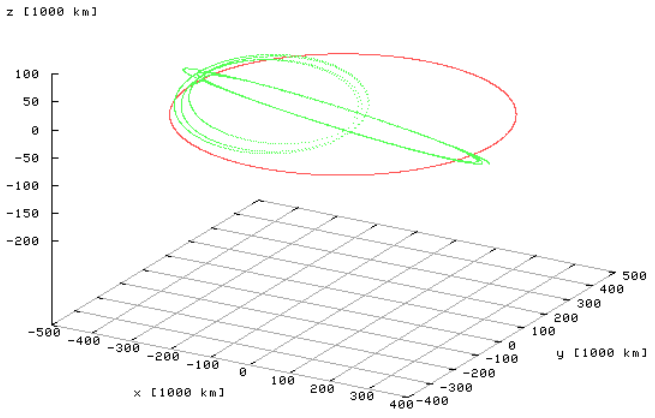
Zároveň nebrání použití všech ostatních dostupných knihoven a konstrukcí jazyka C++, lze tedy programu přidat například grafické uživatelské rozhraní pro přívětivější manipulaci s programem. Pro ladění modelu lze využít obvyklé ladicí nástroje, nevýhodou je ovšem nemožnost dodatečných syntaktických a sémantických kontrol. Knihovna pracuje s překladačem GNU C++ a funguje na operačních systémech Linux, FreeBSD a MS Windows. SIMLIB je volně šiřitelný software, přičemž zdrojové soubory jsou pod licencí GNU LGPL. Všechny lze stáhnout přímo z webu SIMLIBu [25].

Ukázka výsledků simulací, jež dokáže SIMLIB z modelu generovat, je na obrázku 3.2. V části `a)` je zachycena část statistik z diskrétní simulace jednoduchého systému hromadné obsluhy[2], v části `b)` lze vidět 3D vizualizaci výsledků spojité simulace z modelu družice pohybující se v soustavě Země-Měsíc[3].

a)

```
+-----------------------------------------+
| FACILITY Box                            |
+-----------------------------------------+
|  Status = BUSY                          |
|  Time interval = 0 - 1000               |
|  Number of requests = 100               |
|  Average utilization = 0.993702         |
+-----------------------------------------+
  Input queue 'Box.Q1'
+-----------------------------------------+
| QUEUE Q1                                |
+-----------------------------------------+
|  Time interval = 0 - 1000               |
|  Incoming  153                          |
|  Outcoming  98                          |
|  Maximal length = 55                    |
|  Average length = 33.4586               |
|  Minimal time = 4.15499                 |
|  Maximal time = 396.788                 |
|  Average time = 231.744                 |
+-----------------------------------------+
```

b)



Obrázek 3.2: Výsledek a) diskrétní a b) spojité 3D simulace v SIMLIBu

[2]Celý příklad viz https://www.fit.vutbr.cz/~peringer/SIMLIB/examples/model2.html.
[3]Celý příklad viz https://www.fit.vutbr.cz/~peringer/SIMLIB/examples/druzice3D.html.

3.3 UPPAAL

Pro modelování a analýzu samočinně parkujících vozidel byl vybrán modelovací nástroj UPPAAL.

3.3.1 Základní informace

UPPAAL je prostředí a nástroj pro modelování, validaci a verifikaci real-time systémů. Nástroj je vyvíjen ve spolupráci mezi Uppsala University ve Švédsku a Aalborg University v Dánsku. Jak si lze povšimnout, název UPPAAL je spojením počátečních tří písmen názvů obou univerzit. První vydání programu proběhlo v roce 1995, nynější verze, tzv. UPPAAL2k, se objevila o čtyři roky později . Uživatelské rozhraní je naprogramováno v jazyce Java, ostatní pak v C++. [1]

Modely se vytváří pomocí časových automatů, což je šestice $(Q, q_0, X, \Sigma, \delta, I)$ [18], kde

- Q je konečná množina stavů
- $q_0 \in Q$ je počáteční stav
- X je konečná množina hodin
- Σ je konečná množina akcí
- $\delta \subseteq Q \times 2^C \times \Sigma \times 2^X \times Q$ konečná množina hran
- $I : Q \to 2^C$ přiřazuje invariant každému stavu

3.3.2 Uživatelské rozhraní

Prostředí UPPAALu obsahuje tři hlavní části: editor, simulátor a verifikátor.

Grafický editor

V grafickém editoru se vytváří model systému pomocí stochastických časových automatů, které můžou být rozšířené o proměnné typu integer, strukturované datové typy, synchronizace nebo různé typy urgence. Časový automat je konečný automat rozšířený o časové proměnné. Skládá se tak ze stavů, které jsou propojeny hranami.

Stavy jsou reprezentovány kolečkem. Existují čtyři různé druhy:

- `initial` - počáteční stav automatu, každý automat má přesně jeden
- `urgent` - zastavuje čas, tzn. čas neběží, když je proces v tomto stavu
- `committed` - stejně jako urgentní stavy zastavuje čas, navíc pokud je proces v tomto stavu, další krok musí vést z některého z commited stavů
- `normal` - stav, který nepatří do žádné z předchozích kategorií

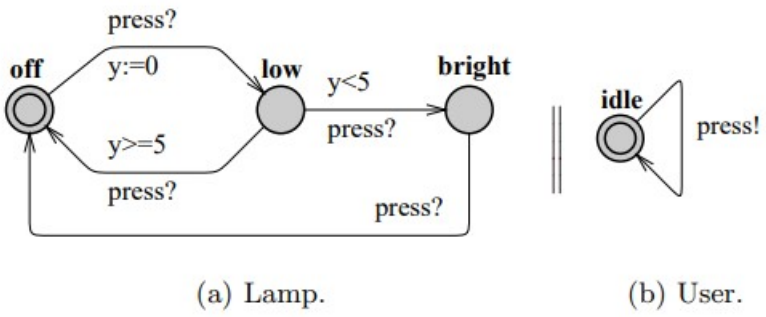
Stav může mít `invariant`, což je výraz, který určuje podmínku, která musí platit po celý čas, kdy se automat nachází v daném stavu. Používá se proto tehdy, když je potřeba nastavit horní hranici podmínky.

Hrany jsou reprezentovány čarou. Může obsahovat čtyři různé položky:

- `select` - nedeterministicky vymezuje interval, do kterého spadá hodnota proměnné
- `guard` - vymezuje podmínku pro přechod, používá se jako nastavení spodní hranice podmínky
- `synchronization` - umožňuje vysílání a přijímání signálu přes kanály, tím je možné předávat řízení mezi automaty, kanály jsou typu `chan`, `urgent chan` nebo `broadcast chan`
- `update` - aktualizuje hodnotu proměnné

Na obrázku 3.3 lze vidět příklad časových automatů modelující jednoduché vypínání a rozsvěcení lampy. V části (a) je automat znázorňující lampu, která se může nacházet ve třech různých stavech - `off` (vypnutá), `low` (nízká intenzita) a `bright` (vysoká intenzita). V části (b) je zobrazen model uživatele, který vyvolává signál `press`, tedy zmáčknutí tlačítka.

Lampa začíná ve vypnutém stavu. Pokud přijde signál `press` od uživatele, zapne se do módu s nízkou intenzitou. Lampu lze vypnout opětovným zmáčknutím tlačítka. Pokud ovšem uživatel zmáčkne tlačítko dvakrát rychle za sebou, aby lampu rozsvítil, druhým stiskem se dostane lampa do módu s vysokou intenzitou. Z tohoto módu lze lampu vypnout opět stiskem tlačítka. Hodiny `y` v modelu lampy se používají pro detekování, zda uživatel zmáčknul tlačítko dvakrát rychle za sebobu (`y < 5`) nebo ne (`y >= 5`). Před novou detekcí se časovač vždy nuluje (`y := 0`). [19]



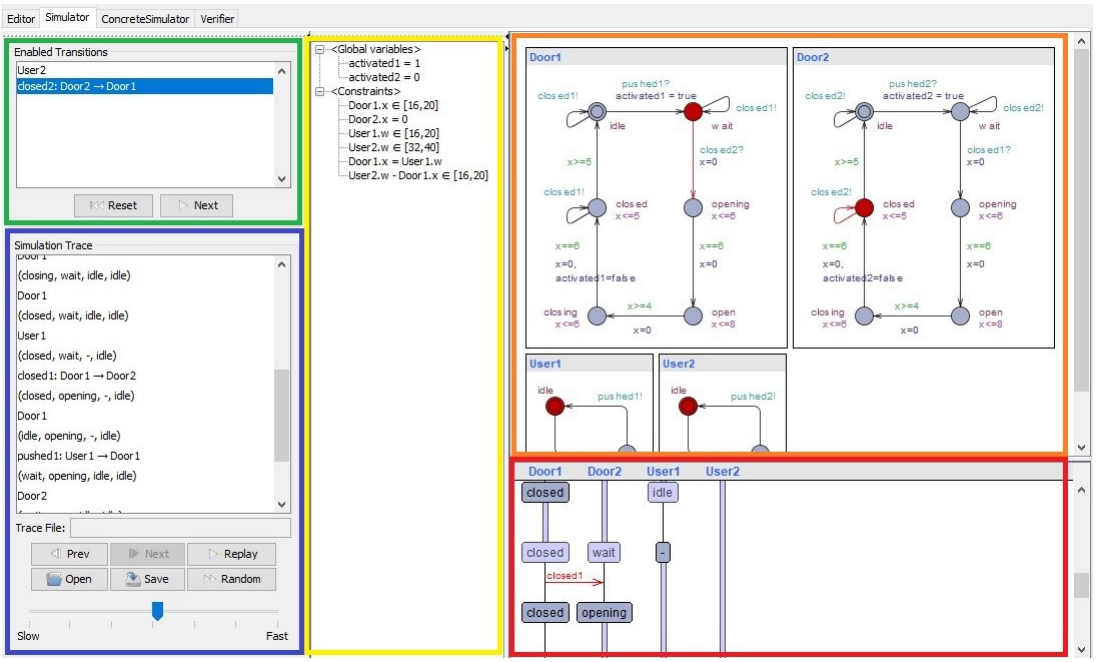
Obrázek 3.3: Jednoduchý příklad lampy v UPPAALu.[19]

Simulátor

Ukázka simulátoru je zobrazena na obrázku 3.5. V zeleně vyznačené části simulátoru lze pustit simulaci manuálně a postupovat po jednotlivých krocích pomocí tlačítka Next. V modře vyznačené části lze simulaci spustit náhodně tlačítkem `Random`, kde si systém sám náhodně vybírá cestu z těch, které jsou v daném čase možné (tzv. Enabled Transitions). Rychlost automatického průchodu stavy je možno regulovat posuvnou lištou dole. Zároveň může uživatel jednotlivé trasy ukládat a poté i znovu importovat. Následně může cestu znovu procházet a sledovat chování v jednotlivých stavech.

Ve žluté části výše zmíněného obrázku lze vidět hodnoty proměnných a časové intervaly. V průběhu simulace lze sledovat, jak se mění hodnoty proměnných po průchodech stavy. Časové intervaly jsou zobrazeny pod jménem `Constraints`. Neukazují přesný časový okamžik, ale celý interval všech možných hodnot.

V části vyznačené oranžově můžeme vidět všechny vytvořené automaty. V každém z nich je momentální aktivní lokace zobrazena červeně. V červeně vyznačené části je pak časové schéma zasílání zpráv. [19]



Obrázek 3.4: Simulátor v UPPAALu.

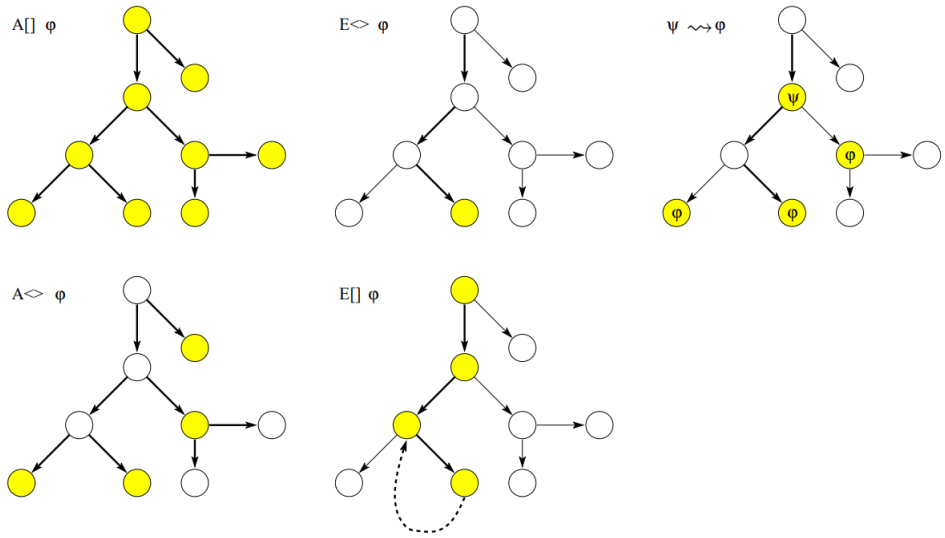
Verifikátor

Ve verifikaci se vychází z jazyka CTL - Computation Tree Logic. Může se tak kontrolovat, zda model plní požadavky, které jsou zapsány daným výrazem. Základní tvary CTL formulí jsou:

- `A[] p` - pro všechna větvení platí, že p je vždy splněno
- `A<> p` - pro všechna větvení platí, že p je splněno alespoň někdy
- `E[] p` - existuje aspoň jedno větvení, že je p vždy splněno
- `E<> p` - existuje takové větvení, že p je alespoň někdy splněno
- `p -> q` - kdykoliv je splněno p, q je také splněno
- `A[] not deadlock` - pro všechna větvení platí, že nedojde k uváznutí

3.3.3 UPPAAL SMC

UPPAAL SMC je rozšíření, které je v UPPAALu podporované od verze 4.1.4 vydané roku 2011. SMC je zkratkou pro Statistical Model Checking, tedy statistické ověřování modelu. To je sada technik, které sledují několik běhů systému s ohledem na určitou zkoumanou vlastnost, udělají z nich statistiky a jejich výsledky poté používají k vyhodnocování míry správnosti návrhu. Oproti klasickému UPPAALu uživatelské rozhraní umožňuje specifikovat rozdělení pravděpodobnosti, kterým se řídí časové chování. Ve verifikátoru pak přidává další dotazy, kterými lze stochastický model ověřovat.



Obrázek 3.5: Formule podporované v UPPAALu. Žluté stavy jsou ty, ve kterých formule ϕ platí. Zvýrazněné hrany ukazují cesty, na kterých se formule vyhodnocuje. [19]

Uživatel si může vizualizovat časový průběh hodnot proměnných v simulovaných běžích, konkrétně pomocí dotazu

$$\texttt{simulate N [<= T] \{ } E_1, .., E_k \texttt{ \}}$$

kde `N` je přirozené číslo určující počet simulací, které probíhají v časovém intervalu `<0,T>` a $\texttt{E}_1, .., \texttt{E}_k$ značí k proměnných, jejichž hodnoty jsou zjišťovány a vizualizovány.

Pravděpodobnosti, s jakou nastanou určité jevy lze zjistit pomocí výrazu

$$\texttt{Pr[<= T]} \ (\psi)$$

kde `T` určuje časový interval `<0,T>`, ve kterém simulace probíhá a ψ je zkoumaný výraz. Ve výchozím stavu probíhá simulace pouze jedna, počet `N` lze nastavit výrazem ve tvaru

$$\texttt{Pr[<= T; N]} \ (\psi)$$

Lze také testovat hypotézy výrazem

$$\texttt{Pr[<= T; N]} \ (\psi) \texttt{>= } p_0$$

kde zkoumáme, zda je pravděpodobnost větší (či menší) než práh $\texttt{p}_0$. Obdobným způsobem lze také porovnávat vůči pravděpodobnosti jiného jevu.

Kapitola 4

Návrh řešení a popis modelu

Tato kapitola se zabývá návrhem řešení a popisem implementace modelu samočinně parkujícího vozidla v prostředí většího parkovacího domu či garáže pomocí nástroje UPPAAL.

4.1 Návrh řešení

Omezení a zjednodušení

Vozidlo i jeho okolí, v tomto případě především parkoviště, je značně rozsáhlý systém, proto budeme uvažovat zjednodušený model. Co se týče okolního prostředí, nebere se v potaz žádné počasí, a tedy ani faktory jako vítr, déšť či náledí, které by mohly ovlivňovat například brzdnou dráhu vozidla či chování v zatáčkách. Vozovka je tedy suchá, bez jakýchkoli nerovností, neklesající a nestoupající. Neklade se důraznější ohled také na statické požadavky případného zastřešení či dokonce dalších pater z hlediska přesného rozmístění a počtu nosných sloupů a konstrukcí.

Model začíná v okamžiku, kdy řidič je s vozidlem na vjezdu do parkoviště a následně předává řízení systému automatického parkování, a končí v momentě, kdy je vozidlo na výjezdu z parkoviště a předává řízení zpět řidiči. Řidič mezitím nijak s autem neinteraguje, předpokládá se, že ani nemusí být ve vozidlu fyzicky přítomen. Pokyn zaparkování či vyparkování může zadat buď fyzicky v autě nebo například přes mobilní aplikaci viz poslední odstavec sekce 2.3.

Způsob řešení autonomnosti

Na systém samočinně parkujícího vozidla lze pohlížet dvěmi různými způsoby. První způsob předpokládá, že parkoviště bylo stavěno za účelem automatického parkování a má vlastní systém, se kterým vozidlo dokáže komunikovat. V parkovišti mohou být rozestaveny podpůrné senzory, které by měly autu pomáhat se pohybovat po parkovišti, aniž by s ním řidič musel jakkoli interagovat. Systém parkoviště by měl mít přehled o pozicích volných parkovacích míst, na které může vozidlo navést nejkratší, případně nejrychlejší cestou. V případě, že by chtěl systém navádět nejrychlejší cestou, musel by mít přehled i o pohybech ostatních aut (ať už autonomních nebo neautonomních), chodců a o výskytu pevných nepohybujících se překážek, jako může být například odložená krabice nebo kočárek. Polohy těchto subjektů by mohl snímat například kamerami.

V druhém konceptu figuruje obyčejné parkoviště, parkující vozidlo je tak ochuzeno o celkový přehled. Cestu si musí hledat samo - zde se předpokládá, že je vozidlo již vysoce

autonomní. Při cestě po parkovišti se může orientovat například pomocí vodorovného dopravního značení, v tomto případě hlavně pomocí bílých čar. K parkování si pravděpodobně vybere první volné parkovací místo, které svými senzory zaznamená. Pozice nemusí být ideální, ale samo auto neví, kde je, či zda vůbec existuje další volné místo - patrně tak bude výhodnější zaparkovat hned na prvním volném místě.

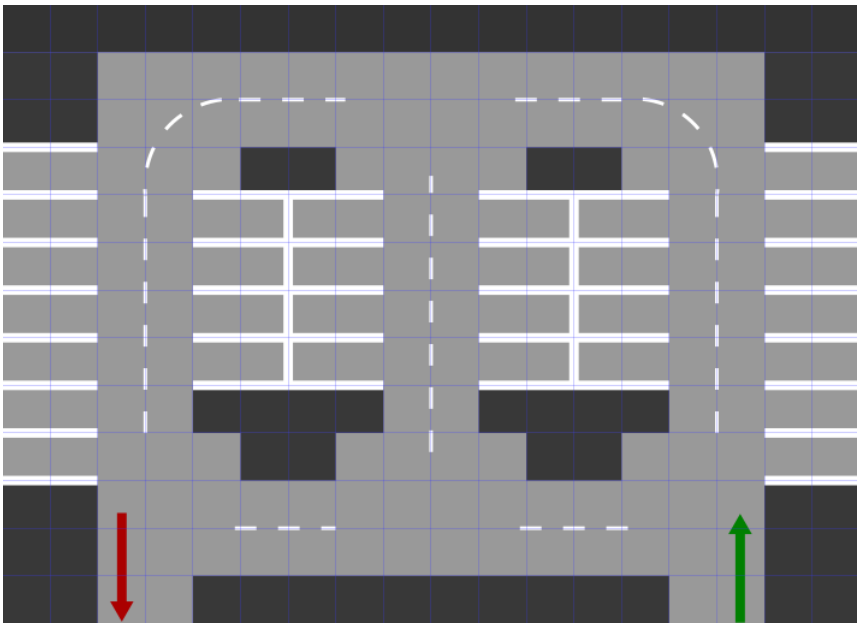
Tato práce popisuje model systému na základě principu z první zmíněné možnosti, kde parkovací systém komunikuje s vozidlem, protože takový systém působí, že bude spolehlivější a pro rychlost a efektivitu parkování výhodnější. Jedinou nevýhodu je to, že aby se takto dalo parkovat, je zapotřebí nejen automatického vozidla, ale také chytrého parkoviště. Možnost reálného využití se tak velmi ztenčuje a v budoucnu předpokládá velké investice do takovýchto parkovištních systémů.

4.2 Použité abstrakce

V této kapitole budou popsány použité abstrakce při vytváření modelu.

4.2.1 Abstrakce parkoviště

Je navrhnuto parkoviště o 30 parkovacích místech s jedním vjezdem a jedním výjezdem tak, jak lze vidět na obrázku 4.1. Tmavé plochy značí zastavěné oblasti jako mohou být například nosné konstrukce, patníky či zeleň. Není nijak vymezené, zda se musí jednat o vnitřní či venkovní parkoviště, vzhledem k popsaným zjednodušením v sekci 4.1 má návrhu blíže uzavřené parkoviště. Světle šedou barvou je vyznačena vozovka a parkovací místa. Vozovka je obousměrná, pouze vjezd a výjezd je jednosměrný.



Obrázek 4.1: Navrhnutá podoba parkoviště.

Parkoviště je navrhnuto jako 2D pole, přičemž každý čtvereček má rozměry 3×3 metry. Šířka vozovky i parkovacích míst je pak rovněž 3 metry, což odpovídá parametrům, které jsou popsané v části 2.4. Toto dělení je na obrázku 4.1 znázorněno jemnými modrými

hodnota	význam
1-6	vozovka
8	volné parkovací místo
9	zastavěná plocha
10	vybrané parkovací místo
11	výjezd

Tabulka 4.1: Tabulka používaných statických hodnot na mapě.

čarami. Každé políčko má svou statickou hodnotu podle polohy na mapě viz tabulka 4.1. Z těchto hodnot je pak vytvořena podoba parkoviště ve 2D poli podobně, jak je ukázáno na obrázku 4.2.

Aby byl pohyb vozidla po parkovišti více realistický, je potřeba mít jemnější mapu. Ta se vytvoří pomocí funkce, která každé políčko rozdělí na několik dalších políček. Hodnota nového dílku se dědí z původního políčka. Za koeficient zjemnění byla zvolena hodnota 5, nová velikost nejmenšího dílku na mapě je tedy $0,6 \times 0,6$ metrů. Model tak bude více realistický a zároveň bude únosná i výpočetní náročnost. Po vytvoření zjemněné mapy se bude v modelu pracovat už pouze s ní.

```
{{ 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9},
 { 9, 9, 0, 2, 1, 1, 1, 1, 0, 3, 1, 1, 1, 1, 1, 0, 9, 9},
 { 9, 9, 1, 0, 1, 1, 1, 5, 1, 0, 1, 1, 1, 4, 0, 2, 9, 9},
 { 8, 8, 1, 4, 0, 9, 9, 0, 1, 6, 0, 9, 9, 0, 1, 1, 8, 8},
 { 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8},
 { 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8},
 { 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8},
 { 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8, 8, 8, 1, 1, 8, 8},
 { 8, 8, 1, 1, 9, 9, 9, 9, 1, 1, 9, 9, 9, 9, 1, 1, 8, 8},
 { 8, 8, 1, 1, 0, 9, 9, 0, 6, 1, 0, 9, 9, 0, 4, 1, 8, 8},
 { 9, 9, 3, 0, 6, 1, 1, 1, 0, 0, 5, 1, 1, 1, 0, 0, 9, 9},
 { 9, 9, 0, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1, 1, 2, 3, 9, 9},
 { 9, 9,11, 1, 9, 9, 9, 9, 9, 9, 9, 9, 9, 0, 1, 9, 9}};
```

Obrázek 4.2: Navrhnutá podoba parkoviště jako 2D pole.

Každé políčko má nejen statickou hodnotu, ale i hodnotu dynamickou, která ukazuje, zda je momentálně na dané pozici nějaký objekt. Objektem je typicky vozidlo nebo překážka.

4.2.2 Abstrakce samočinně parkujícího vozidla

Rozměry parkoviště jsou koncipovány pro osobní auto běžných rozměrů, model nepočítá s většími vozidly typu nákladní auto, auto s přívěsem a podobně. Rozměry běžného automobilu jsou nastaveny na šířku 1,9 metru a délu 4,7 metru. V mapě je tak auto reprezentováno třemi políčky na šířku a osmi na délku. Hodnota poloměru otáčení je stanovena na 5,3 metru. Reálně se hodnoty maximální povolené rychlosti na venkovních parkovištích pohybují kolem 20-30 km/h, na vnitřních často jen 10 km/h. V modelu by rychlost vozidla na parkovišti měla dosahovat maximálně 15 km/h vzhledem ke spíše vnitřní povaze okolních podmínek.

4.2.3 Abstrakce zatáček

Zatáčení je řešeno pomocí Reeds-Sheppových křivek, jejichž rovnice 2.6 se používají k výpočtu nových souřadnic. Základním bodem je střed vozidla a krajní body se po každém pohybu přepočítávají podle středových souřadnic a úhlu natočení vozidla. Přední a zadní náprava tak jede po stejné trajektorii, což je zjednodušení oproti realitě, ve které každé kolo vykonává o trochu jinou dráhu. Křivky se nepoužívají pouze v samotných zatáčkách, ale i při najíždění na parkovací místo a následném vyparkování. Na parkovací manévr lze totiž také nahlížet jako na Reeds-Sheppovu křivku.

4.3 Popis modelu

V této sekci budou popsány jednotlivé časové automaty a základní funkce a proměnné.

4.3.1 Základní funkce a proměnné

Následuje seznam základních funkcí a proměnných používaných v modelu.

Konstanty a proměnné

`M_PI` - konstanta pí
`N` - počet pohybujících se překážek v jeden moment
`default_size` - velikost jednoho políčka na základní mapě, odpovídá hodnotě 300 cm
`koef` - koeficient zjemnění, používá se hodnota 5
`area` - abstrakce základní mapy parkoviště, jsou v ní obsaženy statické hodnoty políček
`fineMap` - zjemněná mapa parkoviště, obsahuje statické i dynamické hodnoty pro každé políčko
`vehicle_dimensions` - struktura obsahující rozměry vozidla
`auto_position` - struktura obsahující souřadnice `x` a `y` pro střed vozidla, všechny rohy a přední senzor
`speed` - aktuální rychlost samočinně parkujícího vozidla
`moving_time` - doba, za kterou vozidlo vykoná pohyb, vypočítává se z rychlosti `speed`
`park_sector` - číslo parkovacího sektoru, počítá se na mapě zprava
`motion_type` - typ pohybu, který se má vykonat
`position_value` - statická hodnota políčka na mapě, kde se právě nachází střed vozidla
`degrees_at_start` - uchovává směr orientace vozidla na začátku zatáčky
`degrees_turned` - počet stupňů, o které se vozidlo v zatáčce už otočilo

Globální funkce

`initialize_park_spot()` - náhodně vybere parkovací místo, kam má vozidlo zaparkovat a zapíše ho do mapy
`fill_fine_map()` - naplní zjemněnou mapu hodnotami podle základní abstrakce parkoviště
`dynamic_value_change()` - aktualizuje pozici vozidla na mapě
`initialize_vehicle()` - inicializace vozidla na pozici vjezdu na parkoviště
`update_positions()` - aktualizuje souřadnice vozidla při pohybu
`move_forward()` - vykonává pohyb dopředu
`move_backward()` - vykonává pohyb dozadu
`start_turn()` - nastaví parametry na začátku zatáčky

`turn_motion()` - vykoná pohyb vozidla při zatáčení
`rectify_from_center()` - srovná pozice krajních bodů vozidla po zatáčce, znovu je spočítá od souřadnic středu vozidla

Lokální funkce

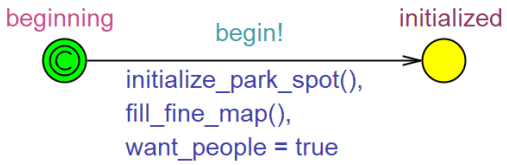
`Car: speed_update()` - zrychluje na maximální povolenou rychlost, pokud tomu nic nebrání
`Car: move_time_update()` - aktualizuje proměnnou `moving_time`
`MoveConductor: direction_decider()` - rozhodne směr na křižovatce
`MoveConductor: trace_to_parking()` - vybírá směr na křižovatce cestou k parkování
`MoveConductor: trace_to_exit()` - vybírá směr na dané křižovatce na cestě k výjezdu
`Pathfinder: side_sensoring()` - detekuje parkovací místo
`Pathfinder: front_sensoring()` - detekuje výjezd
`Parker: parking_initialize()` - nastaví parametry na začátku parkování
`Parker: set_first_center()` - nastavuje první centrum otáčení při parkování
`Parker: set_second_center()` - nastavuje druhé centrum otáčení při parkování
`Parker: park_speed_update()` - aktualizuje rychlost `speed` při parkování
`Peoplemaker: appear()` - zvolí počáteční místo chodce na kraji parkoviště nebo zastavěné cesty blízko vozovky
`Peoplmaker: walk_forward()` - vykonává pohyb překážky simulující např. přecházení cesty
`Peoplemaker: walk_chaotic()` - vykonává náhodný pohyb překážky
`Peoplemaker: appear_suddenly()` - simuluje pohyb skočení do vozovky
`ObstacleSensor: obstacle_sensoring()` - snímá prostor před vozidlem za účelem detekce překážek
`ObstacleSensor: deceleration()` - snižuje rychlost vozidla před překážkou

4.3.2 Časové automaty

V této sekci budou popsány jednotlivé časové automaty, ze kterých sestává vytvořený model. Jedná se dohromady o 8 automatů, konkrétně `Initializer`, `Car`, `Pathfinder`, `ObstacleSensor`, `MoveConductor`, `Carmover`, `Parker` a `Peoplemaker`. Počáteční stavy automatů jsou značeny zelenou barvou a synchronizační stavy (předávání řízení mezi automaty) žlutou barvou, pokud není dáno jinak.

Inicializace prostředí

Časovým automatem `Initializer` zobrazeným na obrázku 4.3 se spouští celý model. Jako první náhodně vybere místo zaparkování pomocí funkce `initialize_park_spot()`. Na vstupní mapě `area[][]` pak označí příslušné místo hodnotou 10. Z modifikované vstupní mapy následně vytvoří zjemněnou mapu parkoviště prostřednictvím funkce `fill_fine_map()`. Příkazem `want_people = true` povolí pohyb lidí po vzniklém parkovišti, který je dále popsán v automatu `Peoplemaker` viz dále. Tím je vytvořeno okolní prostředí a může být vyslán signál `begin`, kterým se předává řízení automatu `Car`.



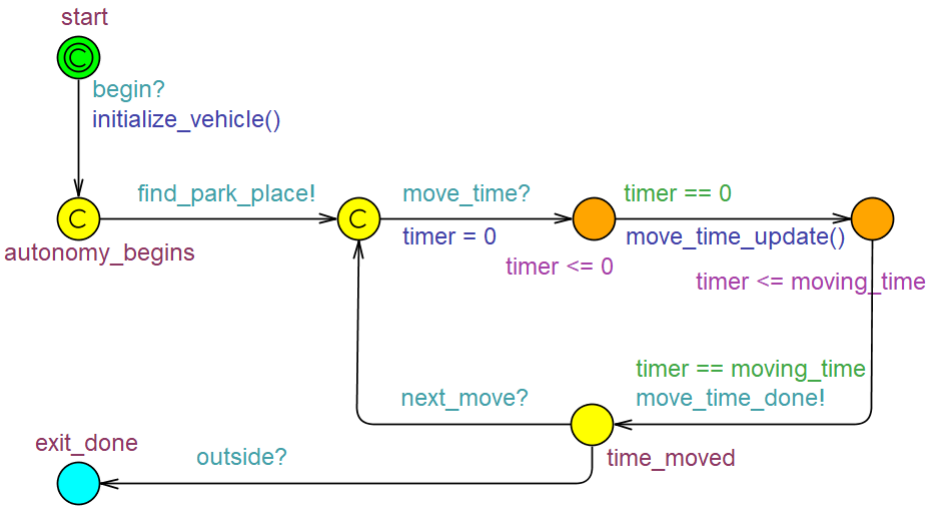
Obrázek 4.3: Initializer

Hlavní řídicí automat

Hlavním řídicím automatem je automat `Car`, který reprezentuje samotné samočinně parkující vozidlo. Jak lze vidět na obrázku 4.4, automat začíná přijetím signálu `begin` od automatu `Initializer`. Jako první inicializuje vozidlo funkcí `initialize_vehicle()`, čímž dá autu souřadnice a pozici na mapě u vjezdu na parkoviště. Tím se automat dostane do stavu `autonomy_begins`, což značí začátek autonomie, a zahajuje hledání parkovacího místa signálem `find_park_place`. Předává tak řízení automatům, které se starají o pohyb.

Po vykonání pohybu přijde signál `move_time`, kterým se automat dostane do prvního z oranžových stavů, kterými se značí místo posunu času. Jako první se vykoná funkce `move_time_update()`, která z aktualizované rychlosti vozidla (maximální rychlost vozidla na parkovišti by neměla přesáhnout 15 km/h) vypočítá veličinu `moving_time`, která značí jak dlouho vozidlu trvá daný pohyb. V druhém oranžovém stavu se pak čeká přesně tak dlouho. Po uplynutí daného času se automat dostává do stavu `time_moved` a zároveň vysílá signál `move_time_done`, kterým předává řízení opět automatům, které zajišťují další pohyb. Pokud je další pohyb žádoucí, přijde signál `next_move`, kterým se automat dostane zpátky na začátek cyklu, který posouvá čas.

V druhém případě přijde signál `outside`, kterým se dává vědět, že je vozidlo na výjezdu z parkoviště. Automat se tak dostává do světle modrého stavu `exit_done`, který je úspěšným koncem simulace poté, co auto dokázalo zaparkovat, vyparkovat a najít cestu k výjezdu.



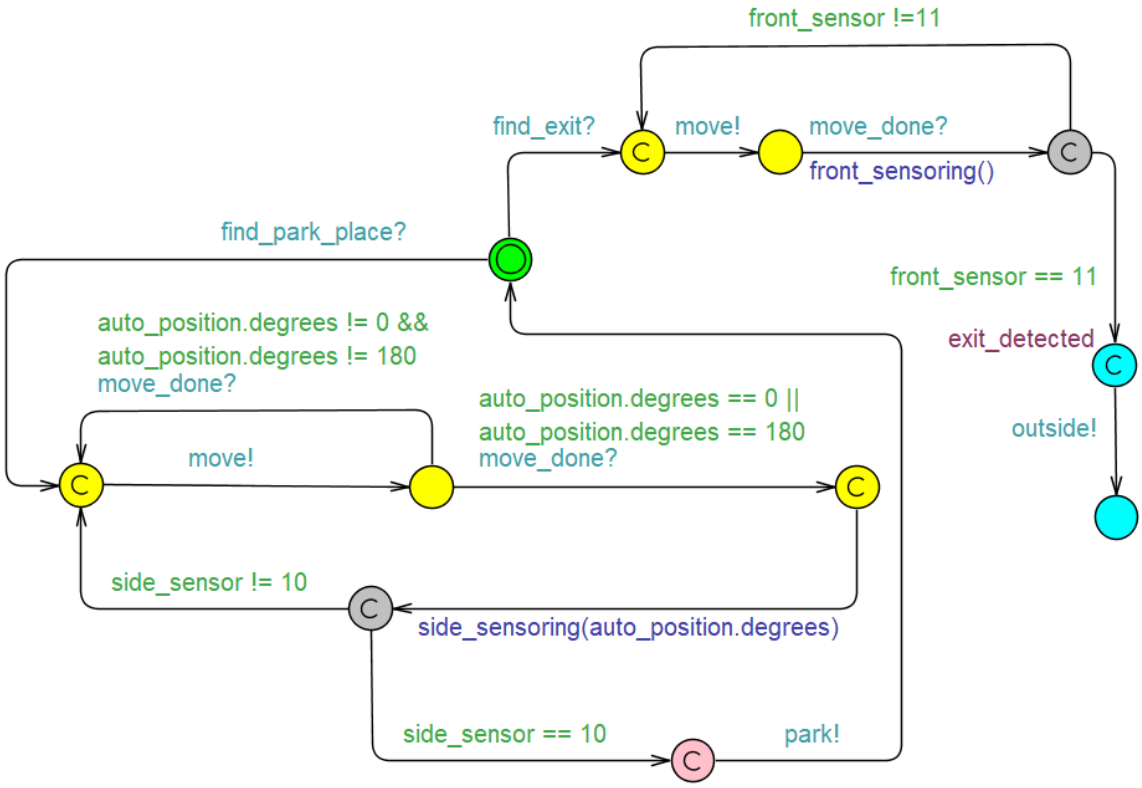
Obrázek 4.4: Car

Detekce parkovacího místa a výjezdu

Automat `Pathfinder` zobrazený na obrázku 4.5 začíná přijetím signálu `find_park_place` od automatu `Car`. Tím začíná hledat parkovací místo. Vysílá signál `move`, kterým dává pokyn k pohybu. Poté, co je pohyb vykonán, přichází signál `move_done`. Pokud je v tu chvíli orientace vozidla `auto_orientation.degrees` taková, že projíždí kolem parkovacích míst (tedy 0 nebo 180 - dáno typem parkoviště), snaží se pomocí funkce `side_sensoring()` najít bočními senzory určené parkovací místo, které je označeno hodnotou 10. Pokud senzory místo nedetekují nebo vůbec nejsou spuštěny, když neprojíždí kolem parkovacích míst, vrací se zpátky do prvního žlutého stavu, ze kterého se pak opět vysílá signál `move`.

V případě, že senzory detekují parkovací místo, dostává se automat do růžového stavu a následně vysílá signál `park`, kterým spouští automat `Parker`. Tím se automat dostal opět do počátečního stavu, ve kterém čeká, dokud nezachytí signál `find_exit`, ze kterého vyplývá, že vozidlo již vyparkovalo a hledá cestu ven z parkoviště.

Následuje podobný cyklus jako při posouvání se k parkovacímu místu, ve kterém se signálem `move` dává pokyn k pohybu. Zda v cyklu pokračovat, se zde určuje pomocí funkce `front_sensoring()`, která snímá prostor před vozidlem. Pokud je detekovaná hodnota rovna 11, což značí výjezd, dostane se automat do stavu `exit_detected`, ze kterého se vysílá signál `outside`. Jestliže výjezd detekován nebyl, vrací se automat na začátek cyklu.

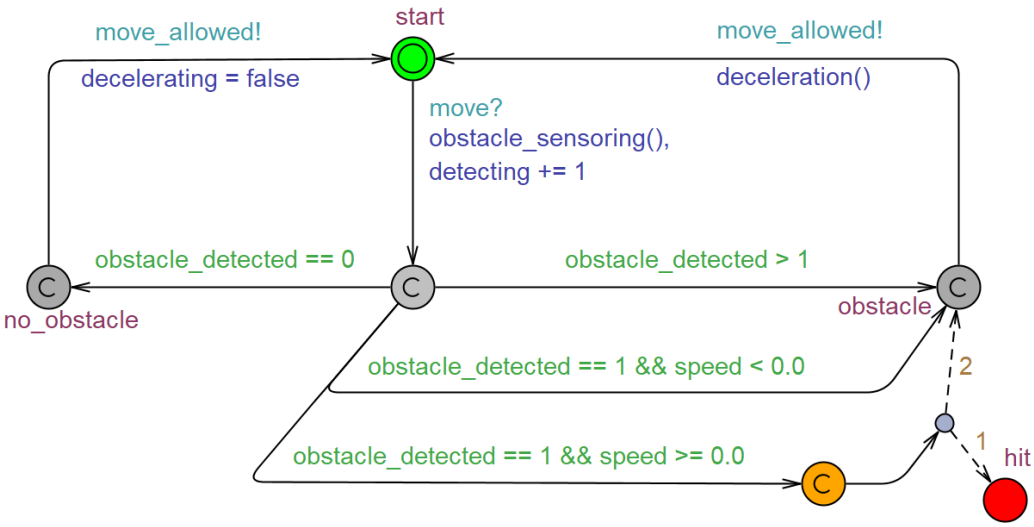


Obrázek 4.5: Pathfinder

Detekce překážek

Automat `ObstacleSensor`, který lze vidět na obrázku 4.6, se stará o detekci překážek a reakce na ně. Začíná přijetím signálu `move` od automatu `Pathfinder` a voláním funkce `obstacle_sensoring()`, která snímá prostor před vozidlem v šířce 3 metry, tedy šířce vozovky, a do dálky až 3 metrů v pěti úrovních. Pokud se žádná překážka nezaznamená, dostane se automat do stavu `no_obstacle`. Z něj se poté vyšle se signál `move_allowed`, nastaví se ukazatel zrychlení `decelerating` na hodnotu `false` a vrací se zpátky do počátečního stavu.

Pokud je překážka detekována, dostává se automat do stavu `obstacle`. Je-li překážka detekována těsně před vozidlem, tzn. do 60 cm, ale rychlost vozidla je nulová, dostává se také do stavu `obstacle`. Jestliže ale rychlost při těsném kontaktu nulová není, je pravděpodobnost 1:2, že nastane srážka, která simulaci končí s nezdarem. Pokud se srážce podaří zabránit, dostane se automat opět do stavu `obstacle`. Z něj se následně vysílá opět signál `move_allowed`, ale sníží rychlost pomocí funkce `deceleration()` v závislosti na vzdálenosti překážky.



Obrázek 4.6: ObstacleSensor

Výběr pohybu

Automat `MoveConductor`, který lze vidět na obrázku 4.7, začíná přijetím signálu `move_allowed` od automatu `ObstacleSensor`. Tím se dostává do bílého rozhodovacího stavu. Rozhoduje se podle proměnné `position_value`, ve které je obsažena hodnota políčka na mapě, na kterém je aktuálně střed vozidla. Vztah mezi hodnotou proměnné a následným stavem je vyjádřen v tabulce 4.2.

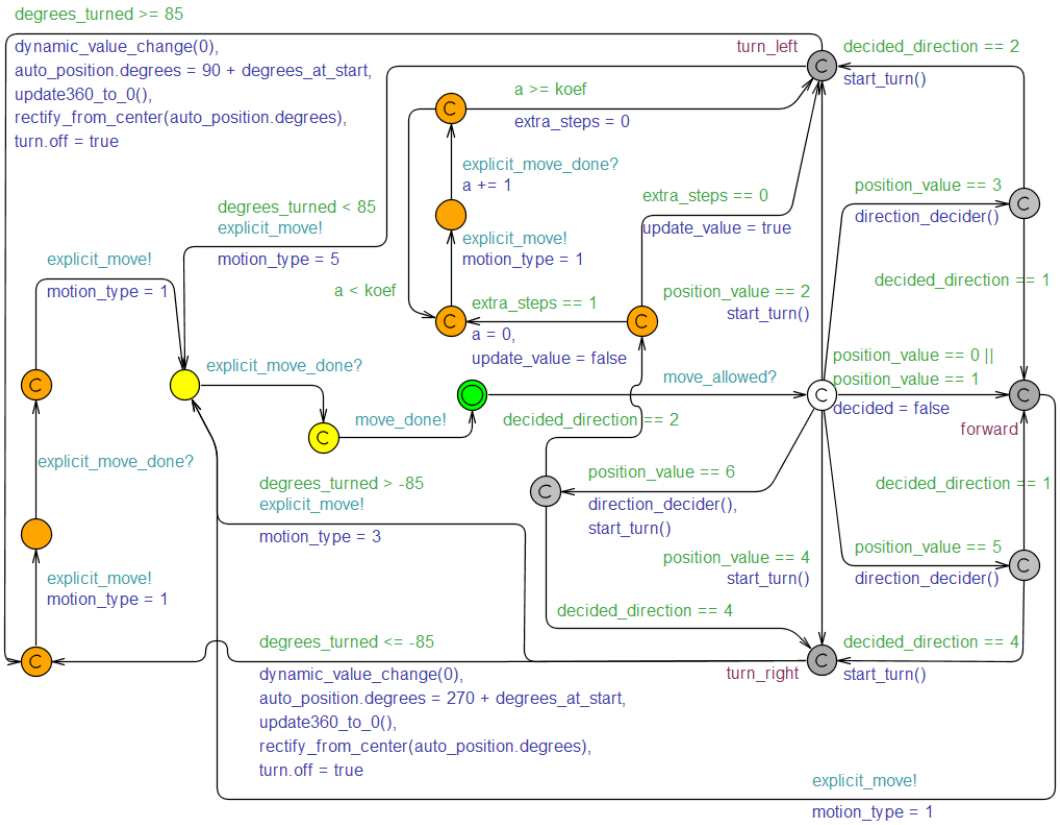
Při hodnotách 3, 5 a 6 jsou následující stavy světle šedé. V nich je nutno směr rozhodnout pomocí funkce `direction_decider()`, protože se jedná o křižovatku. Podle rozhodutí se pak automat dostává do některého z tmavě šedých stavů `turn_left`, `turn_right` nebo `forward`. Výjimkou je odbočení na křižovatce doleva, kde je potřeba provést mezitím ještě dokročení před zahájením odbočení. To se děje v oranžových stavech nacházejících se zhruba uprostřed automatu.

position_value	význam	následující stav
1	dopředu	forward
2	doleva	turn_left
4	doprava	turn_right
3	dopředu-doleva	
5	dopředu-doprava	
6	doleva-doprava	

Tabulka 4.2: Tabulka hodnot proměnné `position_value`

Automat se tedy nachází v jednom z tmavě šedých stavů podle vybraného směru. Pokud není vozidlo už na konci zatáčky, tak ze všech tří stavů následně přechází do levého žlutého stavu, přičemž vydává signál `explicit_move` a nastavuje hodnotu proměnné `motion_type`. Tím předává řízení automatu `Carmover`. Pokud je pohyb vykonán, přijme signál `explicit_move_done` a vzápětí vysílá signál `move_done`, kterým se řízení vrací zpátky automatu `Pathfinder`.

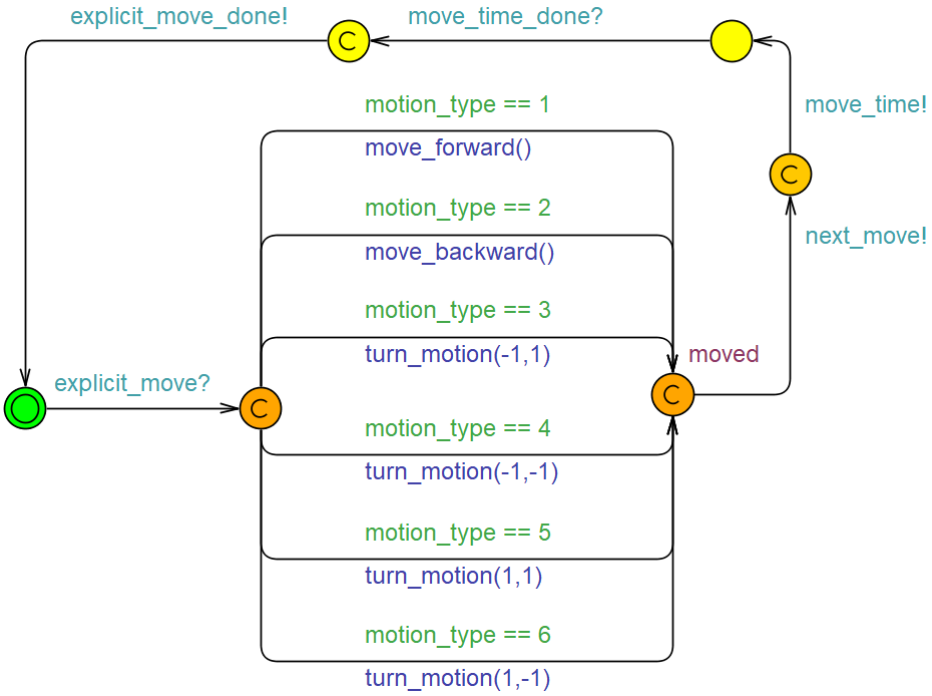
Pokud je vozidlo na konci zatáčky, tedy hodnota `degrees_turned` je větší než 85 při zatáčení doleva nebo menší než -85 při zatáčení doprava, aktualizují se údaje a automat se dostane do spodního z oranžových stavů v levé části obrázku automatu. Následuje dokročení po zatáčce a pak už shodně jako v ostatních případech se dostane do prvního ze žlutých stavů, odkud odkomunikuje poslední pohyb.



Obrázek 4.7: MoveConductor

Vykonání zvoleného pohybu

Automat `Carmover`, který je zobrazen na obrázku 4.8, je spouštěn signálem `explicit_move`. Následně podle hodnoty proměnné `motion_type` vykoná příslušnou funkci, která provede pohyb vozidla tak, že upraví souřadnice vozidla a podle toho aktualizuje hodnoty na mapě. Tím se automat dostane do stavu `moved`, ze kterého vysílá signál `next_move` a poté signál `move_time`. Tím zahajuje posun času v hlavním automatu `Car`. Jakmile je čas posunut, lze se po přijetí signálu `move_time_done` posunout do dalšího stavu, ze kterého se potom vysílá signál `explicit_move_done`, čímž se vrací režie předešlému automatu, který tento automat na začátku volal.



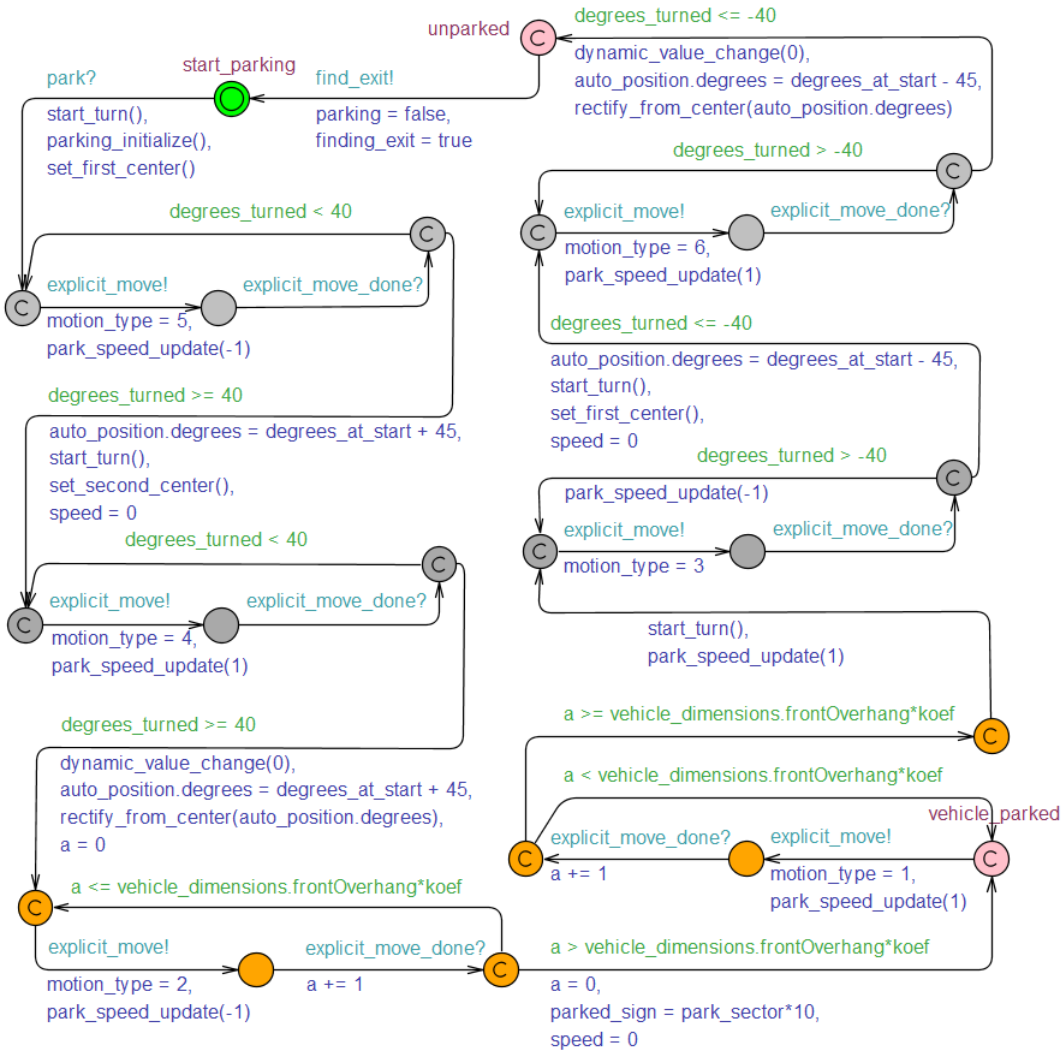
Obrázek 4.8: Carmover

Parkování

O parkování se stará automat `Parker`, který je vyobrazen na obrázku 4.9. Vozidlo parkuje najetím pozadu podobně, jako je naznačeno na obrázku 2.10 c). Trajektorie pohybu je složena ze zatočení doleva dopředu, couvání doprava a následným docouváním rovně na parkovací místo. a po stejné trajektorii i vyparkovává.

Automat je spuštěn signálem `park`, který je vyslán automatem `Pathfinder`. Zároveň je volána funkce `start_turn()`, která se volá před každým začátkem zatáčení, aby vynulovala hodnoty `degrees_turned`, tedy o kolik stupňů se již vozidlo otočilo, a nastavila hodnotu `degrees_at_start`, aby byla uložena hodnota velikosti natočení před otáčením. Inicializují se také hodnoty potřebné celkově k parkovacímu manévru funkcí `parking_initialize` a prostřednictvím funkce `set_first_center()` se nastaví centrum otáčení prvního zatáčení.

Tím se automat dostane na začátek cyklu složeného ze světle šedých stavů. Vysílá se signál `explicit_move` pro vykonání pohybu a funkcí `park_speed_update(-1)` začíná snižovat rychlost. Po přijmutí signálu `explicit_move_done`, kterým je potvrzeno vykonání



Obrázek 4.9: Parker

pohybu, se rozhoduje, zda se bude v otáčení pokračovat - tedy se vozidlo otočilo o méně než 40°, nebo přejde k další fázi parkování.

Pokud se vozidlo otočilo už o více než 40° (respektive 45°, ale kvůli zaokrouhlování je hranice mírně snížena), přechází do tmavě šedého stavu, kterým začíná cyklus couvání doprava. Přepočítává se velikost natočení auta ve stupních kvůli zaokrouhlování, funkcí `start_turn` se zahajuje další zatáčení a za pomoci funkce `set_second_center()` se nastavuje pozice nového centra otáčení. Rychlost klesla na nulu, protože vozidlo mění směr jízdy. Opět komunikuje s automatem `Carmover` přes kanály `explicit_move` a `explicit_move_done` pro vykonání pohybu. Pomocí funkce `park_speed_update(1)` opět navyšuje rychlost, protože se rozjíždí z nuly.

Po dokončení couvání doprava (opět o 45°) se automat dostává do prvního z oranžových stavů, které značí dojetí rovně na parkovací místo. Pohyb probíhá opět v cyklu za pomoci stejných signálů a rychlost se znovu snižuje až k nule. Poté, co auto zacouvá tak, že je již celé na zvoleném parkovacím místě, přechází automat do růžového stavu `vehicle_parked`.

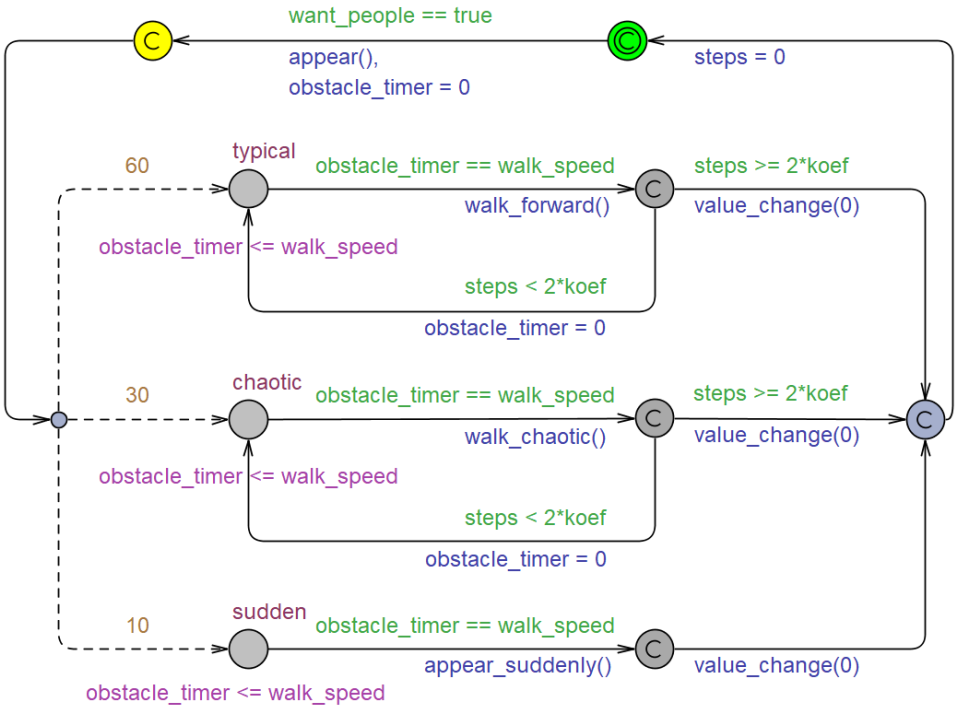
Následuje vyparkování, které je složeno z úplně stejných pohybů jako zaparkování, jen v opačném směru. Jednotlivé pohyby jsou označeny na obrázku automatu 4.9 stejnou bar-

vou, lze tedy vidět, že pravá strana, na které se provádí vyparkování je téměř zrcadlová se stranou levou, ve které auto parkuje.

Jakmile vozidlo vyparkuje, dostane se do růžového stavu `unparked`. Následně se vysílá signál `find_exit`, který dává automatu `Pathfinder` vědět, že skončilo parkování a začíná fáze hledání výjezdu.

Generátor pohybujících se překážek

V modelu parkoviště jsou vytvořeny pohybující se překážky, které reprezentují primárně různé typy chodců. Automat `Peoplemaker`, který lze vidět na obrázku 4.10, dokáže řídit vždy jednu překážku v daném momentu. Pro větší hustotu provozu na parkovišti je proto nutné přidat do modelu tento automat několikrát. Počet lze nastavit nastavením konstanty N v globálních deklaracích.



Obrázek 4.10: Peoplemaker

Automat začne v momentě, kdy je přepnuta hodnota proměnné `want_people` z `false` na `true`, což dělá hlavní automat `Car`. V tu chvíli se přechází do žlutého stavu a prostřednictvím funkce `appear()` se na mapě objeví překážka. Místa objevení jsou na krajích parkovacích a zastavěných míst, kde je hned vedle vozovka. Překážky tak mohou rovnou vstupovat do vozovky a blokovat provoz.

Na přechodu ze žlutého stavu se vybírá typ chování překážky. S 60% šancí se automat dostane do stavu `typical`, který značí větev, ve které se překážka chová jako typický chodec, který se snaží přejít cestu. Podobným způsobem se může chovat také auto, které se snaží vyparkovat. V cyklu tak vykoná sadu kroků pomocí funkce `walk_forward`, která aktualizuje souřadnice chodce a zanáší je na mapu. Cyklus je opakován přesně tolikrát, jaká je celková šířka vozovky, a délka jednoho průchodu se řídí proměnnou `walk_speed`. Poté se automat vrátí do počátečního stavu.

S 30% pravděpodobnostní bude překážka vykonávat pohyb chaoticky a dostane se do větve počínající stavem `chaotic`. Může to být například chodec, který se motá kolem kufru svého auta při nakládání věcí, zmatený pes nebo dítě. S šancí 10 % se pak překážka zjevuje ve vozovce náhle - demostruje nepozorné lidi, děti, zvířata či jakékoli další nepředvídatelné překážky. V tomto případě je ale modelováno zmizení stejně rychlé jako objevení, a mezitím se nevykonává v cyklu žádný další pohyb, což by mělo simulovat například vběhnutí do vozovky a rychle uskočení před jednoucím vozidlem v případě, že se objekty navzájem nesrazí.

Kapitola 5

Zhodnocení řešení

Tato kapitola se zabývá zkoumáním vybraných vlastností vytvořeného modelu. Samočině parkující vozidlo v simulacích začíná vždy ve vjezdu na parkoviště. Ve všech simulacích je cílem zaparkovat a po zaparkování vyjet z parkoviště ven.

5.1 Základní scénáře a vlastnosti

V následující sekci se budou pomocí simulací ověřovat klíčové vlastnosti modelu. Ověří se koncové stavy modelu v části 5.1.1, zda vozidlo dokáže najít cestu a zaparkovat na vybrané parkovací místo v části 5.1.2, ověření jednotlivých typů pohybů překážek v části 5.1.3 a v části 5.1.4 se demonstruje vliv okolí na rychlost vozidla.

5.1.1 Ověření koncových stavů

Jako první je potřeba ověřit koncové stavy modelu. Jako úspěšný konec simulace se považuje ten, kdy vozidlo vyjede z parkoviště. Za neúspěšný konec simulace lze považovat případ, kdy dojde ke srážce s překážkou, protože vozidlo v řízení poté již nepokračuje a k výjezdu se tak samo nedostane. Dále je zapotřebí ověřit, zda se vozidlo v průběhu simulace pohybuje pouze po parkovišti a nevyjíždí nikam mimo. Oba tyto scénáře se budou ověřovat dvakrát. Poprvé bez přítomnosti překážek a podruhé se středním provozem překážek, tzn. s 10 pohybujícími se překážkami v jeden čas.

Za nepřítomnosti překážek

Pro zjištění dosažitelnosti úspěšného koncového stavu simulace `exit.done` byla simulace spuštěna s dotazem 5.1.

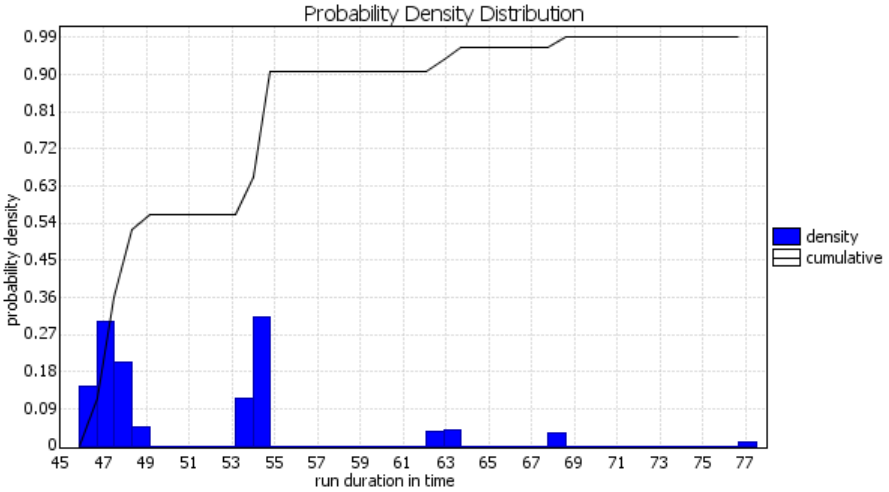
$$Pr[<= 1000]\ (<>\ car.exit\_done) \tag{5.1}$$

Výsledek lze vidět na obrázku 5.2. Pokud se na parkovišti v době průjezdu nevyskytují žádné překážky, dosáhne distribuční funkce hodnoty 1, čímž je ověřeno, že vozidlo vždy najde cestu ven a neztratí se po cestě.

Pomocí dotazu 5.2 se zjistí, v jakém rozmezí souřadnic se vozidlo pohybuje.

$$simulate\ 100\ [<= 200]\ \{auto\_position.center.y,\ auto\_position.center.x\} \tag{5.2}$$

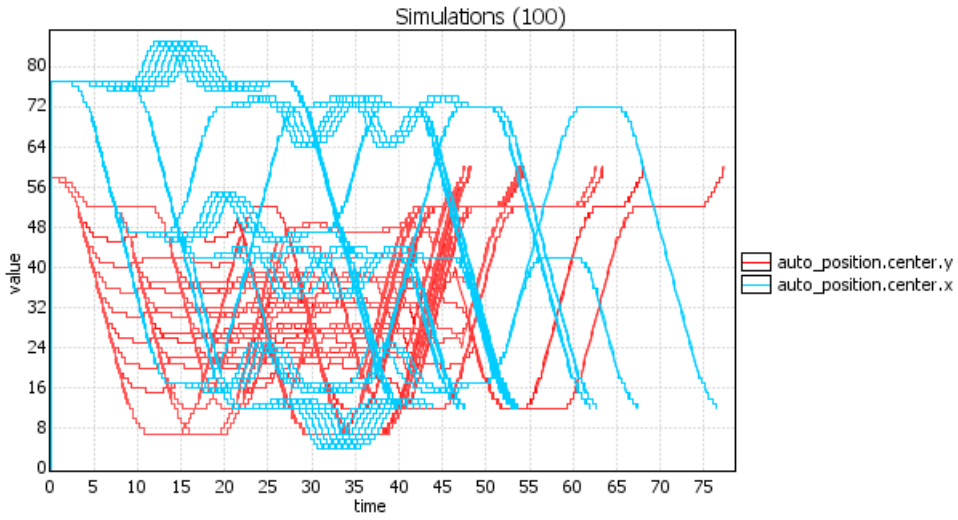
Vygenerovaný graf lze vidět na obrázku 5.2. Souřadnice x, vyznačená modře, se pohybuje v rozmezí $\langle 5, 85\rangle$ a souřadnice y, která je vyznačena červenou barvou, v intervalu $\langle 7, 60\rangle$.



Obrázek 5.1: Graf dotazu 5.1 za nepřítomnosti překážek

Obě centrové souřadnice vozidla se tak ve všech časech drží v rozmezí souřadnic mapy parkoviště, čímž se ověřuje, že vozidlo v průběhu nevykonává část trasy mimo mapu.

Z grafu lze také vyčíst časy výjezdů kolem 47, 54, 64, 68 a 77 sekund, které jsou znázorněny koncem křivek, které reprezentují jednotlivé souřadnice v čase. Časy výjezdů odpovídají i rozložení pravděpodobnosti dosažení koncového stavu simulace v grafu 5.1.



Obrázek 5.2: 100 běhů simulace souřadnic vozidla `x` a `y` za nepřítomnosti překážek

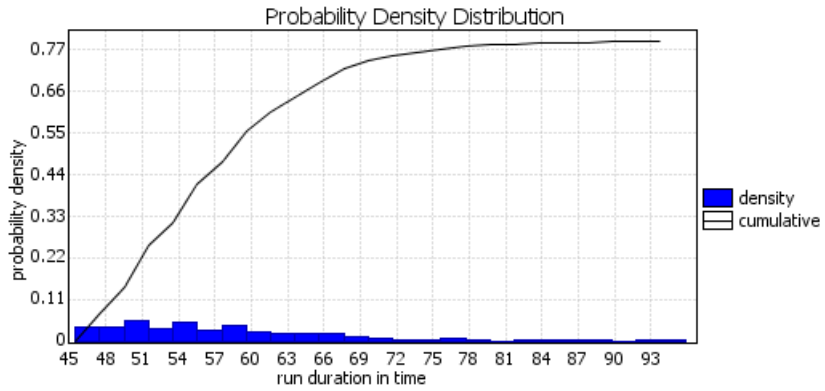
Za přítomnosti překážek

V následujících simulacích je nastavena konstanta N, která značí souběžný počet překážek v čase, na hodnotu 10. Z grafu na obrázku 5.3 lze vidět, že kumulativní pravděpodobnost dosahuje pouze hodnoty zhruba 0.8, ne všechna vozidla tak dojedou zdárně do cíle.

Dotazem 5.3 lze zjistit pravděpodobnost neúspěšného konce simulace v podobě srážky.

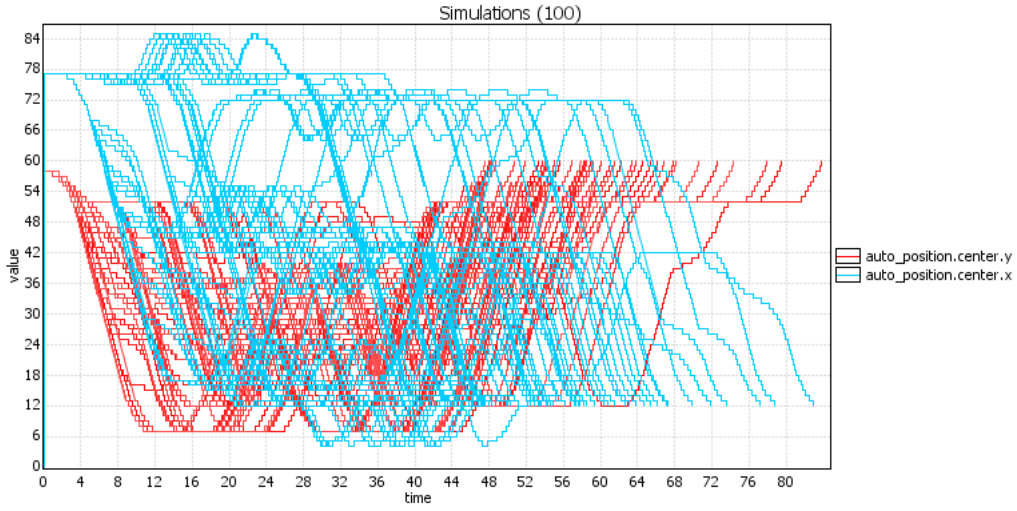
$$Pr[<= 1000]\ (<>\ obstaclesensor.hit) \tag{5.3}$$

Výsledkem je rozmezí $\langle 0.183243, 0.243217 \rangle$ s průměrnou hodnotou zhruba 21 %. Součet pravděpodobnosti srážky a úspěšného výjezdu tedy dává dohromady zhruba 100 %, čímž můžeme považovat všechny možné konce simulací za ověřené.



Obrázek 5.3: Graf dotazu 5.1 za přítomnosti překážek

Na obrázku 5.4 lze vidět, že souřadnice `x` a `y` zůstávají ve stejném rozmezí, jako na obrázku 5.2, auto tedy nevyjíždí ven z parkoviště ani za přítomnosti překážek. Kvůli přítomnosti chodců jsou ale trasy v čase více či méně posunuté.



Obrázek 5.4: 100 běhů simulace souřadnic vozidla `x` a `y` za přítomnosti překážek

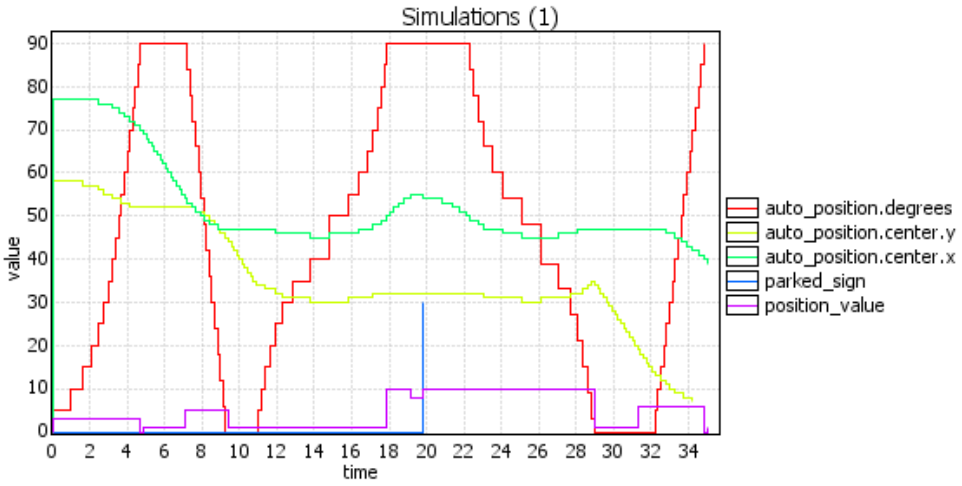
5.1.2 Parkování na určené místo

V této části se bude ověřovat, zda vozidlo dokáže najít cestu k parkovacímu místu a zdárně zaparkovat. Do sektorů 1, 3 a 5 se parkuje z celkového pohledu směrem doprava (auto najíždí na mapě zespodu) a do sektorů 2, 4 a 6 směrem doleva (auto najíždí seshora). Dotazem 5.4 získáme graf, který je vyobrazen na obrázku 5.5 a znázorňuje ukázku parkování do sektoru 3. Pomocí dotazu 5.5 se vygeneruje graf, který je na obrázku 5.6 a zobrazuje parkování do sektoru 2. Číslo sektoru lze z grafů vyčíst pomocí hodnoty proměnné `parked_sign` (modře), která v čase zaparkování změní hodnotu z 0 na právě na číslo sektoru ($\times 10$ kvůli viditelnosti v grafu). Ve výše zmíněných případech se tedy zvýší na hodnotu 30, respektive 20.

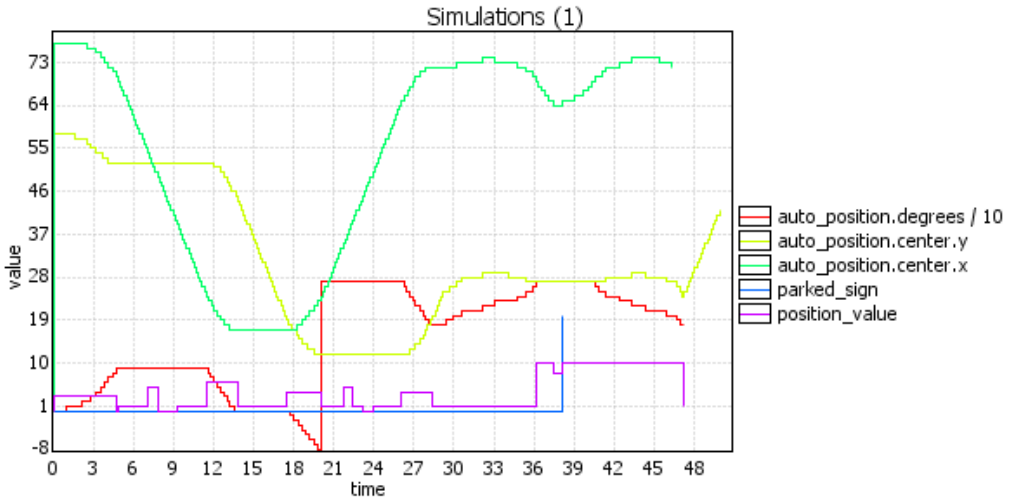
$$simulate\ 1\ [<= 35]\ \{auto\_position.degrees,\ auto\_position.center.y,\ auto\_position.center.x,\ parked\_sign,\ position\_value\} \tag{5.4}$$

$$simulate\ 1\ [<= 50]\ \{auto\_position.degrees/10,\ auto\_position.center.y,\ auto\_position.center.x,\ parked\_sign,\ position\_value\} \tag{5.5}$$

V obou grafech lze vidět, že v čase zaparkování je hodnota `position_value` (fialově) na čísle 10, které značí vybrané parkovací místo. Trvající hodnota 10 v čase vyparkování a tedy nesouměrnost s fázi parkování je dána tím, že v průběhu samotného zatáčení se hodnota `position_value` neaktualizuje.



Obrázek 5.5: Parkování v sektoru 3



Obrázek 5.6: Parkování v sektoru 2

Z obou grafů 5.5 a 5.6 se dají také interpretovat hodnoty souřadnic a natočení vozidla. Souřadnice `y` (žlutě) se v čase parkování příliš nemění, protože parkování probíhá více

v horizontální rovině. Pohyb v této rovině lze vidět na souřadnici `x`, která bud znatelně roste (5.5) nebo klesá (5.6) v závislosti na parkovacím sektoru a tedy směru parkování. Směr natočení `degrees` (červeně) lze přímo v grafu vidět také. V případě parkování do sektoru 3 je vozidlo natočeno směrem doprava, což odpovídá hodnotě 90, při parkování do sektoru 2 je pak hodnota 27, respektive 270, která značí orientaci doleva.

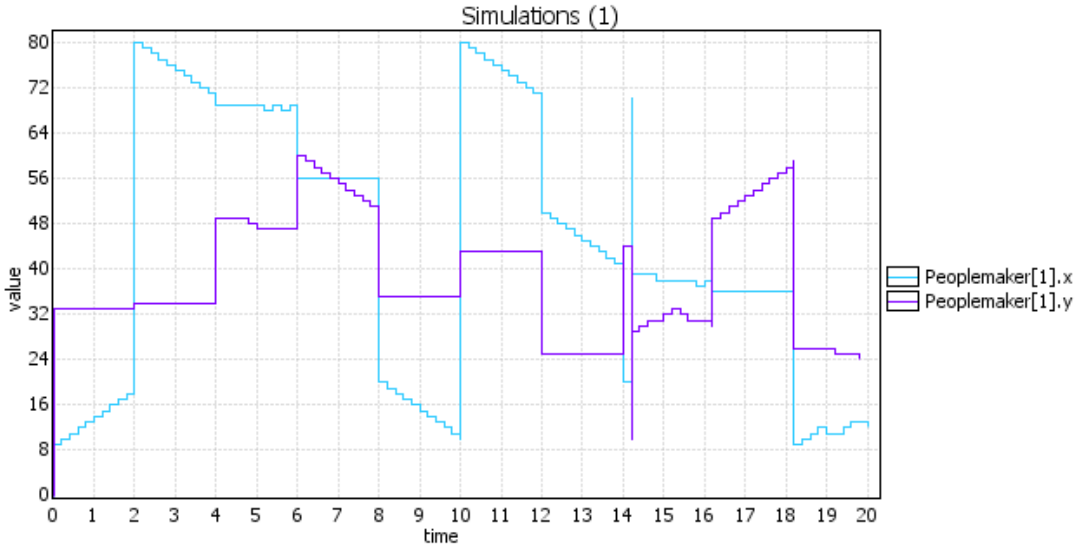
5.1.3 Chování pohyblivých překážek

Dotazem 5.6 se ověří chování pohybujících se překážek po parkovišti.

$$simulate\ 1\ [<= 20]\ \{Peoplemaker(0).x,\ Peoplemaker(0).y,\\ Peoplemaker(1).x,\ Peoplemaker(1).y\} \tag{5.6}$$

Na obrázku vygenerovaného grafu 5.7 lze vidět pohyb jedné překážky v průběhu 20 sekund. Jednotlivé typy pohybů jsou vykonávány zpravidla 2 sekundy, poté je vygenerována nová překážka na jiném místě. Podle změny souřadnic lze rozlišit 3 typy chování.

Pokud se mění pouze jedna souřadnice, jako například v časových intervalech $(0, 2)$, $(2, 4)$, $(8, 10)$ atd., jedná se o první typ chování znázorňující chodce, který přechází cestu. Pokud se mění obě souřadnice jako v úseku $(4, 6)$, $(14, 16)$ a zhruba $(18, 20)$, jde o druhý typ chování, které je chaotické. Třetí typ chování, které je náhodným skokem do vozovky, lze vidět v čase kolem 14 sekund.



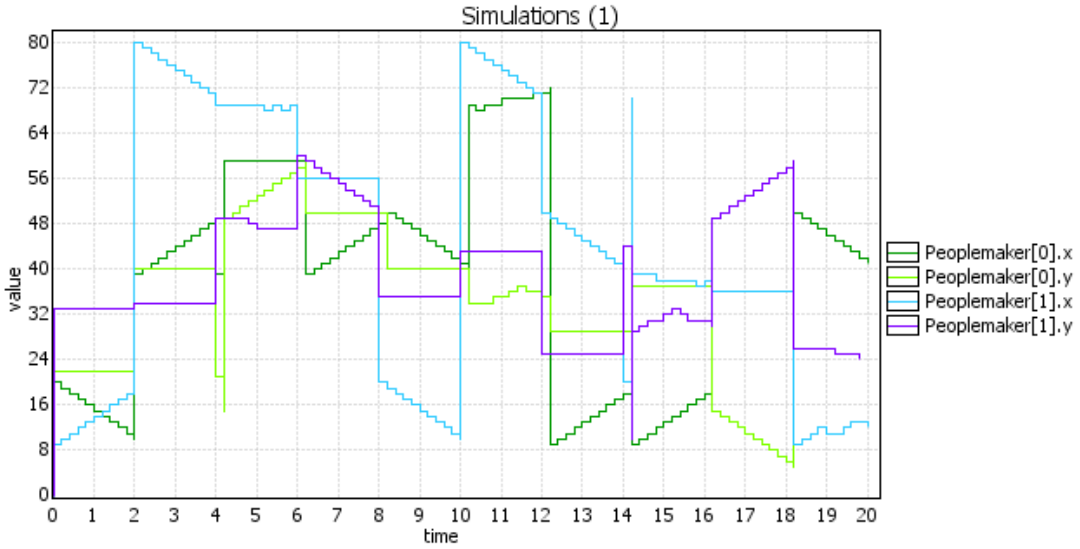
Obrázek 5.7: Chování pohyblivé překážky

Na obrázku 5.8 jsou vidět obě překážky z dotazu 5.6. Zde se ověřuje, zda jsou při spuštění více automatů `Peoplemaker` na sobě navzájem nezávislé pohyby, které se vykonávají ve stejném čase. Zelené křivky souřadnic první překážky a modré křivky druhé překážky nevykazují v grafu žádné viditelné známky korelace, čímž je nezávislost ověřena.

5.1.4 Změna rychlosti při reakci na okolí

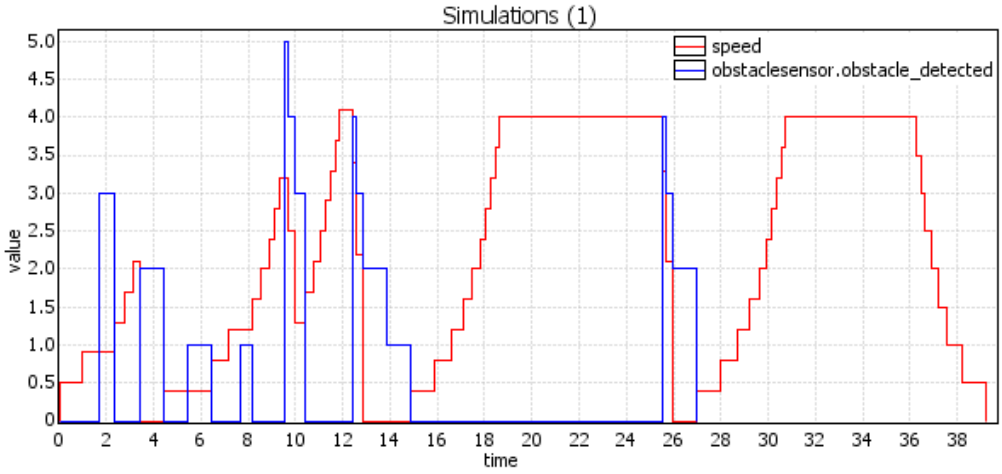
Tato část je zaměřena na ověřování změny rychlosti jako reakci na okolí. Jako první se dotazem 5.7 vygeneruje graf rychlosti v závislosti na vzdálenosti detekovaných překážek.

$$simulate\ 1\ [<= 40]\ \{speed, obstaclesensor.obstacle\_detected\} \tag{5.7}$$



Obrázek 5.8: Chování více pohyblivých překážek zároveň

Graf je vidět na obrázku 5.9. Vzdálenost překážky od auta `obstacle_detected` je vyznačena modře, přičemž může nabývat hodnot $\{0, 1, 2, 3, 4, 5\}$. Maximální detekovatelná vzdálenost je tři metry, rozdíl mezi jednotlivými úrovněmi je tedy vždy 0,6 metru. Čím nižší hodnota je, tím blíže se překážka vyskytuje. V grafu lze vidět detekce překážky ve všech možných úrovních. Ve většině případů detekce překážky vozidlo svou rychlost `speed` (červeně) sníží, pokud je v daný moment rychlost dostatečně nízká, dál se nesnižuje. Skutečnost, že se vozidlo při potkávání překážek s žádnou přímo nesrazilo značí to, že dále pokračuje v jízdě, což je znázorněno měnící se rychlostí v průběhu celého času simulace.



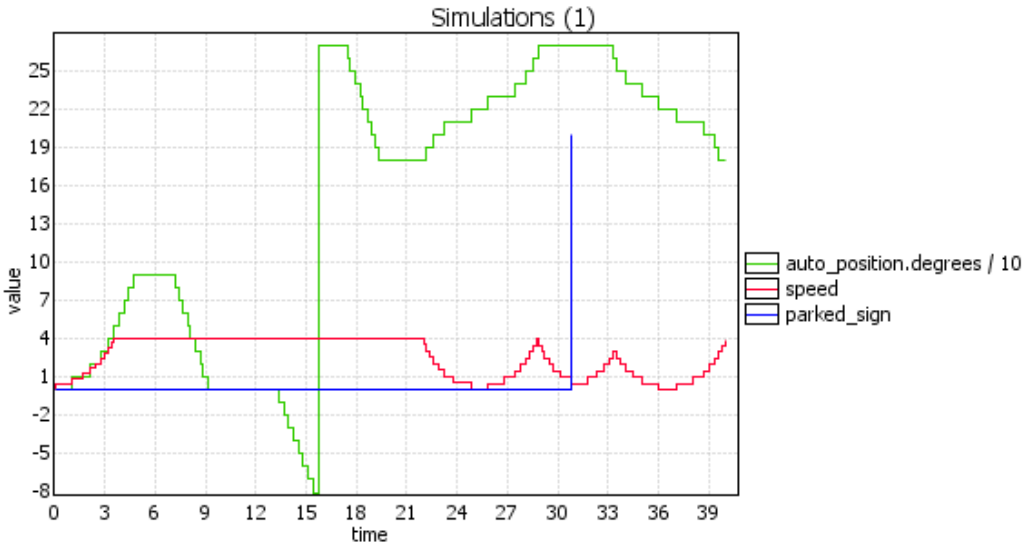
Obrázek 5.9: Změna rychlosti jako reakce na překážku

Rychlost by se měla měnit nejen při detekci překážek, ale také při parkování. To lze ověřit pomocí dotazu 5.8.

$$simulate\ 1\ [<= 40]\ \{autoposition/10,\ speed,\ parked\_sign\} \tag{5.8}$$

Výsledek je možné vidět na obrázku 5.10. Čas zaparkování zde značí opět proměnná `parked_sign` (modře), která v okamžiku zaparkování nabyde hodnoty sektoru ×10. Rychlost `speed` (červeně) začíná klesat v okamžiku zahájení parkovacího úkonu. To je v tomto případě zhruba v čase 22 sekund. Začátak parkování lze vidět i na křivce orientace vozidla `degrees` (zeleně), kdy se její hodnota začíná zvyšovat a auto tak začalo manévr zatáčení doleva. Tato první část parkovací sekvence končí zhruba v čase 25 sekund, kdy se rychlost snížila na 0, protože vozidlo mění směr a začíná couvat. Rychlost opět postupně vzroste a následně klesne na 0, čímž je auto zaparkováno. Vyparkování probíhá analogicky, jen v opačném směru, a lze vidět, že rychlost se mění téměř zrcadlově.

Na grafu lze také dobře vidět třetí situace, kdy se mění rychlost vozidla, a tou je rozjezd v samém začátku simulace. Pokud není důvod zpomalovat, vozidlo si drží rychlost na úrovni 4 m/s (zhruba 15 km/h), což je maximální povolená rychlost na parkovišti. Stejně tak lze na obou grafech rychlosti 5.9 a 5.10 vidět, že rychlost není nikdy záporná.



Obrázek 5.10: Změna rychlosti při parkování

5.2 Vliv hustoty provozu

V této sekci se bude zkoumat vliv hustoty provozu na čas strávený na parkovišti a pravděpodobnost srážky s překážkou, potažmo tedy míru rizikovosti autonomního parkování.

Nejprve se pomocí dotazu 5.9 bude zjišťovat celkový čas strávený na parkovišti od vjezdu, přes parkování až po výjezd. V žádném z experimentů není zahrnut čas stání na parkovacím místě.

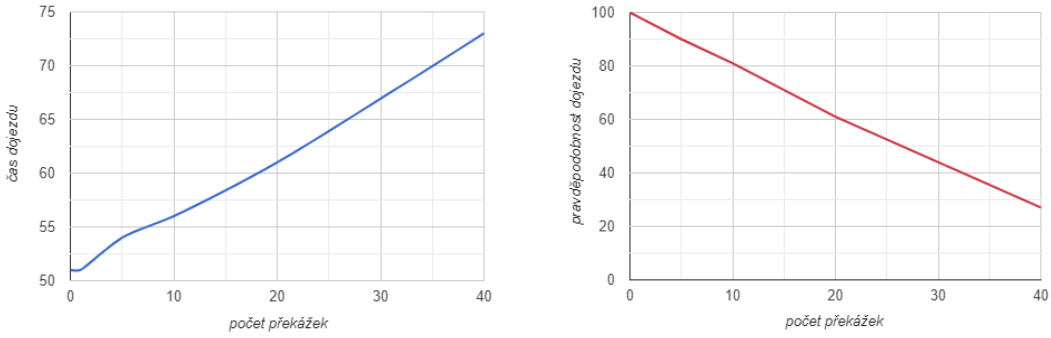
$$Pr[<= 1000]\ (<>\ car.exit\_done) \tag{5.9}$$

Výsledky rozdělení pravděpodobnosti pro různou hustoty provozu, a tedy počet překážek, jsou zobrazeny na obrázku A.1 v příloze A. Z dotazu lze zjistit také průměrná hodnota celkového času stráveného na parkovišti a pravděpodobnost, že se vozidlo k výjezdu dostane beze srážky. Zaokrouhlené hodnoty jsou shrnuty v následující tabulce 5.1.

V grafu 5.11 z hodnot v tabulce 5.1 lze vidět, že průměrný čas dojezdu roste vzhledem k počtu překážek zhruba lineárně a relativně mírně - při 40 souběžně pohybujících se překážkách vzroste čas o necelou polovinu vůči situaci bez provozu. Pravděpodobnost úspěšného

počet překážek	průměrný čas dojezdu	pravděpodobnost dojezdu
0	51,3	100%
1	51,8	98%
5	54	90%
10	56,6	81%
20	61	61%
40	73,5	27%

Tabulka 5.1: Tabulka hodnot ze spuštěných simulací



(a) Čas dojezdu

(b) Pravděpodobnost dojezdu

Obrázek 5.11: Grafy hodnot z tabulky v závislosti na počtu překážek

dojezdu naopak lineárně klesá, a to poměrně značně. Vzhledem k lineární povaze funkce lze předpokládat, že při počtu zhruba 55 překážek bude pravděpodobnost úspěšného projetí parkovištěm téměř nulová.

S narůstající intenzitou provozu je tak mnohem větší šance srážky, přitom rychlost pohybu po parkovišti se nezvyšuje nijak markantně. Z toho lze usoudit, že chování překážek v modelu je rizikové v poměrně velké míře.

Zmenšená rizikovost překážek

Tento experiment se bude snažit snížit procento srážek při průjezdu parkovištěm. V automatu `ObstacleSensor` (obrázek 4.6) se upraví poměr hodnot pravděpodobností na hranách z oranžového stavu. Nově se dostane vozidlo do stavu `hit`, který značí srážku jen v poměru 1:9. Tím se simulují situace, kdy je překážka detekována na kraji vozovky, ale nedojde ke srážce, protože překážka není přímo před vozidlem, které je užší než detekovávaná výseč.

Opět pomocí dotazu 5.9 se zjistí průměrné hodnoty času výjezdu z parkoviště a pravděpodobnost dojetí k výjezdu beze srážky. Výsledné hodnoty jsou shrnuty v tabulce 5.2 v porovnání s původním modelem.

Podle očekávání se pravděpodobnost dojezdu velkou měrou zvýšila, zejména při větším počtu překážek. Teoretická nulová průjezdnost, vyplývající z lineární povahy grafu, by zde nastala zhruba někde mezi 120 a 130 překážkami, což je vzhledem k velikosti parkoviště velmi málo pravděpodobné. Průměrný čas dojezdu zůstává téměř shodný, jen při velkém počtu překážek je nepatrně vyšší. Zvýšení ale není o moc větší, než rozmezí statistické chyby.

počet překážek	průměrný čas dojezdu		pravděpodobnost dojezdu	
poměr	1:2	1:9	1:2	1:9
0	51,3	51,3	100%	100%
1	51,8	51,7	98%	99%
5	54	54	90%	96%
10	56,6	57	81%	93%
20	61	63,2	61%	83%
40	73,5	75,2	27%	68%

Tabulka 5.2: Tabulka hodnot ze spuštěných simulací

Výchozí model, kde byl provoz značně rizikový, je v reálném světě pravděpodobně docela ojedinělý, vzhledem ke skutečnosti, že na parkovištích se taková míra nehod neděje. Upravený model oproti tomu vykazuje mnohem méně rizikové chování překážek, které lze očekávat na větším procentu parkovišť.

Kapitola 6

Závěr

Práce zdokumentovala technologie, pojmy a principy související s činností samočinně parkujícího vozidla a popsala dostupné prostředky v oblasti výpočetního modelování systémů a analýzy jejich vlastností. Pomocí nástroje UPPAAL SMC byl vytvořen model samočinně parkujícího vozidla v prostředí inteligentního parkoviště. V modelu umí vozidlo bez pomoci řidiče najít a dojet k vybranému parkovacímu místu, zaparkovat, posléze vyparkovat a dojet k výjezdu z parkoviště. Pomocí simulací byly vlastnosti modelu ověřeny v několika základních scénářích. V modelu se vyskytují také pohyblivé překážky, které umí vozidlo detekovat a snaží se zabránit srážkám. Podle rizikovosti chování překážek se to vozidlo daří určitou měrou. Při vysoké rizikovosti s počtem součinně pohybujících se překážek nad 55 jednotek se stává parkoviště neprůjezdným, protože je téměř 100% šance, že se vozidlo s překážou srazí. Při upravení chování překážek na menší míru rizikovosti se průjezdnost parkoviště o velké procento zvětšila.

Průjezd parkovištěm bez překážek bez započtení času stání na parkovacím místě podle simulací trvá zhruba 51 sekund. Při pohybu 20 překážek po parkovišti stoupne doba průjezdu pouze zhruba o 15 sekund. V tomto modelu je parkovací místo vybíráno náhodně, stojí ovšem za zvážení, zda v některých případech nevybírat místo například blízko výjezdu, aby doba příjezdu auta z parkovacího místa po zavolání například pomocí mobilní aplikace byla co nejmenší.

Práce by mohla být dále rozšířena o reakce na nepohyblivé překážky, kdy by je vozidlo musel objíždět, přidat možnost poruchy senzorů nebo rozšířit model o další typy parkovacích manévrů. Další možností by bylo neorientovat se na parkovišti pomocí statických hodnot na mapě, ale použít například algoritmus RRT (Rapidly-exploring Random Tree – rychle rostoucí náhodný strom).

Literatura

[1] *About | UPPAAL* [URL https://uppaal.org/features]. Online, navštíveno 17. 1. 2021.

[2] *Automated Valet Parking*
[URL https://www.bosch.com/stories/automated-valet-parking/]. Online, navštíveno 17. 1. 2021.

[3] *Nissan Unveils a Bold New Vision for the London "Black Cab": The Nissan NV200 London Taxi*
[URL https://usa.nissannews.com/en-US/releases/nissan-unveils-a-bold-new-vision-for-the-london-black-cab-the-nissan-nv200-london-taxi].

[4] *Proxel* [URL https://www.proxel.com/en/]. Online, navštíveno 2. 4. 2021.

[5] *Srovnání poloměru otáčení Škody Fabia s ostatními velkými hatchbacky*
[URL https://www.autohled.cz/a/skoda/fabia/t/parametry/polomer-otaceni].

[6] *Ultrasonic sensor* [URL https://www.bosch-mobility-solutions.com/en/solutions/sensors/ultrasonic-sensor/]. Online, navštíveno 2. 4. 2021.

[7] *Volkswagen Golf dimensions* [URL https://www.buyacar.co.uk/volkswagen/golf/golf-hatchback/212/volkswagen-golf-dimensions].

[8] Four Wheels on Jacks Park Car. *Popular Science Monthly*. Sep 1934, s. 58. Dostupné z: https://books.google.cz/books?id=HCgDAAAAMBAJ&printsec=frontcover.

[9] *The Geometry of Perfect Parking*
[URL https://personal.rhul.ac.uk/uhah/058/perfect_parking.pdf]. November 2009. Online, navštíveno 16. 1. 2021.

[10] *ČSN 73 6056 - Odstavné a parkovací plochy silničních vozidel*. March 2011.

[11] *Brooks Walker - The man who eased the pain of parking*
[URL https://autoportal.com/articles/brooks-walker-the-man-who-eased-the-pain-of-parking-3341.html]. March 2015. Online, navštíveno 10. 12. 2020.

[12] *AN5322 - TPMS wheel location introduction and main concepts*
[URL https://www.nxp.com/docs/en/application-note/AN5322.pdf]. Leden 2017. Online, navštíveno 18. 3. 2021.

[13] *Parking Sensors Differences Explained*
[URL https://www.nationwidevehiclecontracts.co.uk/blog/parking-sensors-differences-explained]. August 2017. Online, navštíveno 10. 12. 2020.

[14] *ÚAMK změřil šířku 154 modelů vozů* [URL https://www.uamk.cz/aktuality/1945-uamk-zmeril-sirku-150-modelu-vozu]. August 2017. Online, navštíveno 15. 1. 2021.

[15] *How to Park a Car Perfectly? Easy Guide to Parallel Parking* [URL https://www.acko.com/car-guide/how-to-park-a-car-perfectly-easy-guide-to-parallel-parking/]. June 2020. Online, navštíveno 2. 4. 2021.

[16] Andreev, N. *A brief history of car parking technology* [URL https://www.confused.com/on-the-road/gadgets-tech/parking-technology-brief-history]. June 2018. Online, navštíveno 10. 12. 2020.

[17] Ashish. *Why Is It Easier To Parallel Park Your Car If You Reverse Into The Space Rather Than Drive Straight In?* [URL https://www.scienceabc.com/eyeopeners/why-easier-parallel-park-car-if-reverse-into-space-than-drive-straight-in.html]. November 2019. Online, navštíveno 16. 1. 2021.

[18] Baier, C. a Katoen, J.-P. *Principles of Model Checking*. Leden 2008. ISBN 978-0-262-02649-9.

[19] Behrmann, G., David, A. a Larsen, K. A Tutorial on Uppaal. In:. Leden 2004, sv. 3185, s. 200–236. DOI: 10.1007/978-3-540-30080-9_7. ISBN 978-3-540-23068-7.

[20] Brazil, M., Grossman, P., Rubinstein, H. a Thomas, D. Demonstrating efficiency gains from installing truck turntables at crushers. Listopad 2015.

[21] Brooks, W. *Motor vehicle*. Dec 1938. US Patent 2,139,341.

[22] Košnar, K. *Mobilní robotika* [URL http://old.roznovskastredni.cz/dwnl/pel2007/06/Kosnar.pdf]. Online, navštíveno 21. 3. 2021.

[23] LaValle, S. M. *Planning Algorithms*. Cambridge University Press, 2006. 880 – 886 s. ISBN 0521862051.

[24] Paromtchik, I. E. a Laugier, C. Motion generation and control for parking an autonomous vehicle. In: *Proceedings of IEEE International Conference on Robotics and Automation*. 1996, sv. 4, s. 3117–3122 vol.4. DOI: 10.1109/ROBOT.1996.509186.

[25] Peringer, P. *SIMLIB Home Page* [URL https://www.fit.vutbr.cz/~peringer/SIMLIB/]. Online, navštíveno 15. 1. 2021.

[26] Peringer, P. *Modelování a simulace - studijní opora*. FIT VUT, 2012.

[27] Reeds, J. A. a Shepp, L. A. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*. Pacific Journal of Mathematics, A Non-profit Corporation. 1990, sv. 145, č. 2, s. 367 – 393. DOI: pjm/1102645450. Dostupné z: https://projecteuclid.org:443/euclid.pjm/1102645450.

[28] RV, T. *Behind the Pivot Point: Understanding the Turning Radius of Your RV* [URL https://togorv.com/rv-living/understanding-the-turning-radius-of-your-rv/]. Červen 2019. Online, navštíveno 19. 3. 2021.

[29] Sajdl, J. *Ackermannova podmínka* [URL https://www.autolexicon.net/cs/articles/ackermannova-podminka/]. Online, navštíveno 17. 3. 2021.

[30] Sugeno, M. a Murakami, K. Fuzzy parking control of model car. In: *The 23rd IEEE Conference on Decision and Control*. 1984, s. 902–903. DOI: 10.1109/CDC.1984.272144.

[31] Sussmann, H. J. a Tang, G. Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control. *Rutgers Univ., Tech. Rep. SYNCON*. 1991.

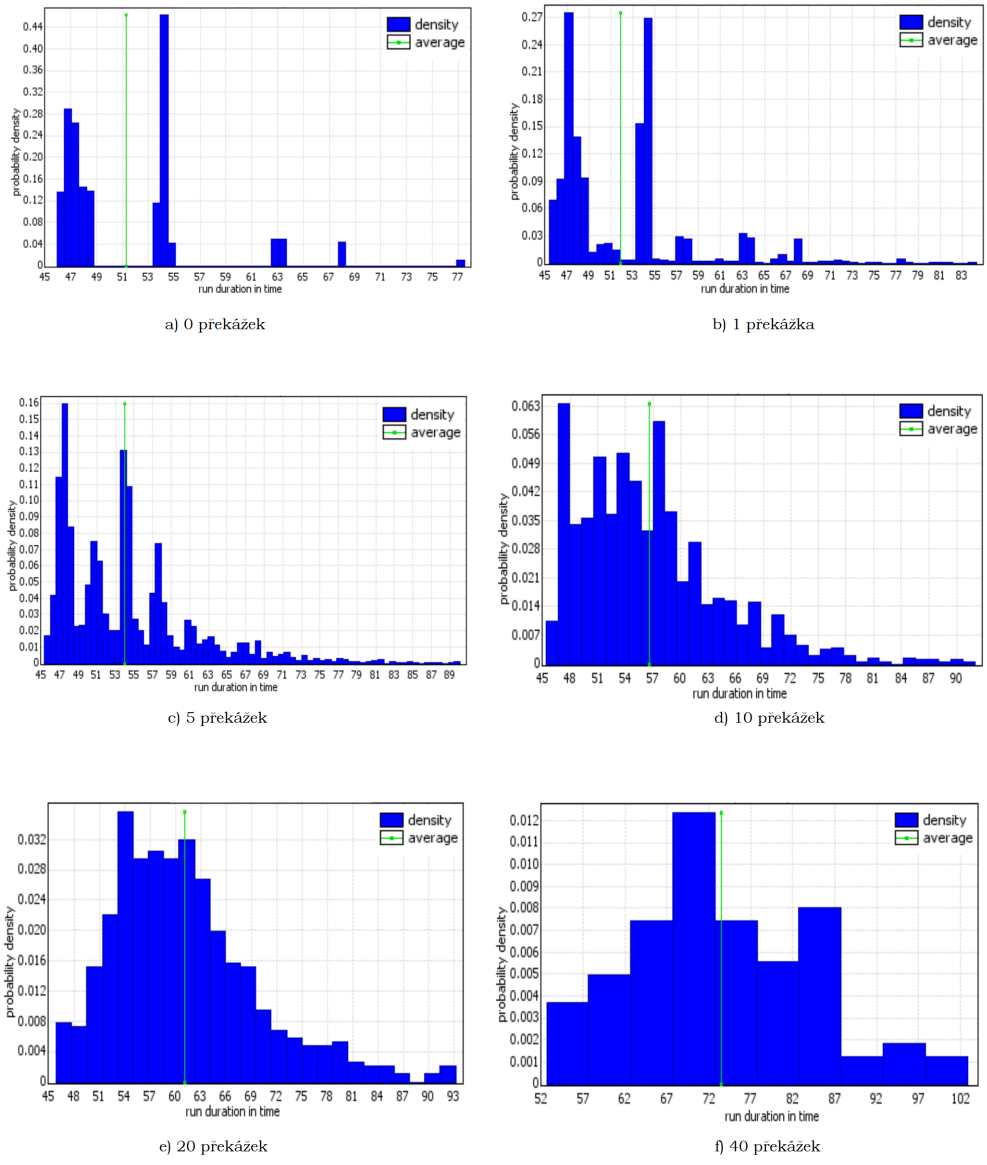
[32] Systèmes, D. *What is Dymola?* [URL https://www.3ds.com/fileadmin/PRODUCTS/CATIA/DYMOLA/PDF/What-is-Dymola-2020x.pdf]. 2019. Online, navštíveno 14. 1. 2021.

[33] The MathWorks, I. *Simulink Product Description* [URL https://www.mathworks.com/help/simulink/gs/product-description.html]. Online, navštíveno 14. 1. 2021.

[34] Zelinka, J. *Automatické parkování – jak funguje Park Assist?* [URL https://www.autohled.cz/magazin/automaticke-parkovani-ndash-jak-funguje-park-assist/1536]. March 2020. Online, navštíveno 5. 4. 2020.

Příloha A

Grafy rozdělení pravděpodobnosti



a) 0 překážek

b) 1 překážka

c) 5 překážek

d) 10 překážek

e) 20 překážek

f) 40 překážek

Obrázek A.1: Grafy rozdělení pravděpodobnosti

Příloha B

Obsah přiloženého CD

CD disk přiložený k práci obsahuje následující soubory:

- doc/ - uchovává všechny potřebné dokumenty k vytvoření této zprávy
- src/ - obsahuje zdrojový soubor modelu ve formátu .xml
- tool/ - obsahuje UPPAAL Stratego verze 4.1.20-7 s rozšířením SMC
- tz-xkruci00.pdf - soubor obsahující tuto zprávu
- README.txt - obsahuje návod ke spuštění a informace o dokumentaci