Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Optimalizace testovacích algoritmů nástroje Taster

Matěj Štula

Vedoucí: Ing. Jan Sobotka, Ph.D.
Obor: Kybernetika a robotika
Srpen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Štula** Jméno: **Matěj** Osobní číslo: **474601**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Optimalizace testovacích algoritmů nástroje Taster

Název bakalářské práce anglicky:

Taster tool algorithms optimization

Pokyny pro vypracování:

1. Seznamte se s problematikou HiL integračního testování metodou Model-Based Testing a software Taster.
2. Prostudujte použité grafové algoritmy procházení modelu použité softwarem Taster.
3. Identifikujte možné nedostatky a navrhněte optimalizaci algoritmů pro generování integračních testů.
4. Navrženou optimalizaci implementujte.
5. Pro urychlení experimentů naprogramujte simulační mód, který nebude respektovat zadaný časový krok.
6. Demonstrujte výsledky práce na vhodných modelech.

Seznam doporučené literatury:

[1] West, Douglas Brent, et al.: Introduction to graph theory. Upper Saddle River, NJ: Prentice Hall, 1996.
[2] J. Zander, I. Schieferdecker, and P.J. Mosterman: Model-Based Testing for Embedded Systems. Taylor & Francis, 2011.
[3] Veselka, Michal: Integrační testování metodou Model-BasedTesting - případová studie. Diplomová práce ČVUT FEL, Praha 2019.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Sobotka, Ph.D., katedra měření FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **12.02.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce:
do konce letního semestru 2020/2021

Ing. Jan Sobotka, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Podpis studenta

Datum převzetí zadání

Poděkování

Děkuji svému vedoucímu Ing. Janu Sobotkovi, Ph.D. za pomoc, rady a odborné vedení při psaní bakalářské práce. A taky díky za všechny ryby.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Liberec, 13. srpna 2020

. . . . . .

Abstrakt

V moderních automobilech je stále více elektroniky a řídících jednotek. Aby vše fungovalo tak, jak má, je zapotřebí, aby spolu všechny součásti spolupracovaly. Pro ověření, že vše je v pořádku, se provádí testy. Možných situací je teoreticky nekonečně mnoho a pro testy se vybírá pouze část situací.

Generování testovacích sad bývalo doménou testovacích inženýrů. V dnešní době se již zapojují počítače. Nástroj Taster je jedním z programů určených na generování HiL integračních testů. Ale stejně jako inženýři, i tento program se musí ve své činnosti zdokonalovat. Z tohoto důvodu se tato práce zabývá popisem jeho strategií pro generování testů, odhalením možných nedostatků a na základě těchto poznatků návrhem strategií nových.

Klíčová slova: Hil, Taster, procházení stavového prostoru, grafy, testování, elektronika automobilu, časovaný automat

Abstract

There are more and more electronics and control units in modern cars. For everything to work as it should, all the components need to work together correctly. To ensure everything works properly, tests are performed. Theoretically, there is an infinite number of possible situations and only some of them are selected for the tests.

Generating tests used to be the domain of testing engineers. Nowadays, computers are involved. Taster is one of the programs designed to generate the HIL integration tests. Similarly to engineers, this program must keep improving. For this reason, this thesis deals with the description of its strategies for generating tests, revealing possible shortcomings and designing new strategies based on this knowledge.

Keywords: HiL, Taster, state space traversing, graphs, testing, car electronics, timed automata

Title translation: Taster tool algorithms optimization

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Uživatelé moderních automobilů používají spousty funkcí, které jsou řízeny počítači. Počínaje prvky důležitými pro samotnou činnost vozu přes bezpečnostní po komfortní služby. Nikdo by jistě nechtěl vůz, který by místo ztlumení rádia zapnul stěrače a místo zapnutí dálkových světel by otevíral kufr. Aby k těmto situacím nedocházelo, jednotlivé prvky i celý systém jsou důkladně testovány.

Jedna z metod testování se nazývá Hardware-in-the-Loop (HiL). Pro urychlení, usnadnění a zkvalitnění vývoje se elektromechanické prostředí ovládané řídící jednotkou nahradí simulátorem. V takové konfiguraci dostává řídící jednotka stejné signály, jako by dostávala z reálných senzorů. Příkladem může být řízení osvětlení. Řídící jednotka dostane signál o stisku tlačítka, které rozsvěcí světla, a výstup nastaví tak, aby se žárovka rozsvítila. Simulátor na řídící jednotku připojuje umělou zátěž. [1]

Pro testování se používá model předpokládaného chování systému. Na základě takového modelu lze vygenerovat testovací sadu, respektive seznam vstupů do řídící jednotky a očekávané výstupy z ní. [2] Taster je jedním z nástrojů, které lze použít pro generování těchto testů na základě modelu chování. Tento nástroj využívá čtyři různé strategie pro generování testů.

Po seznámení se s nástrojem Taster a typem používaných modelů je vytvořen simulační mód, který umožňuje rychlejší procházení systému. Posléze jsou popsány původní strategie procházení včetně jejich nedostatků. Na konci jsou vytvořeny nové strategie a všechny jsou vzájemně porovnány na dvou modelech.

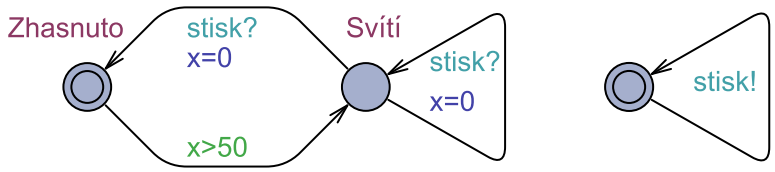
Kapitola 2

Model systému

Systémy jsou modelovány jako síť časovaných automatů. Tyto automaty jsou dále rozšířeny o proměnné, které jsou součástí stavů. Stav systému je definován aktivními uzly všech automatů, stavem hodin a hodnotami proměnných. Hodiny běží synchronně a stejnou rychlostí. Hrany mohou obsahovat podmínky pro projití. [3, 4]

Podmínky na hranách lze vyjádřit jako $x \sim n$ nebo x - $y \sim$ n, kde $x, y \in C$ jsou proměnné, $n \in N$ je přirozené číslo a $\sim \in \{\leq, <, =, >, \geq\}$ jsou operátory. Na jedné hraně může být podmínek více. V takovém případě musejí být pro výběr takové hrany splněny všechny podmínky. [4, 5]

Na obrázku 2.1 je jednoduchá síť časovaných automatů modelující schodišťové osvětlení. Světelný automat zobrazený na obrázku 2.1a obsahuje dva uzly, v nichž se může nacházet. Světlo je buď rozsvícené, či zhasnuté. Dobu rozsvícení hlídá časovač `x`. Když si chce někdo rozsvítit světlo nad schody, tak může stisknout vypínač. Tento stisk se do tohoto automatu dostane pomocí kanálu `stisk?`. Po stisknutí vypínače budou světla svítit po dobu 50 s. Pokud během této doby někdo opět stiskne vypínač, tak se časovač resetuje a začne počítat od začátku. Pokud nikdo nezmáčkne vypínač po dobu 50 s, světla se vypnou. Obrázek 2.1b zobrazuje model vypínače. Ten obsahuje jedinou hranu se synchronizačním kanálem `stisk!`. Uživatel může vypínač kdykoli stisknout, přičemž dojde k vyvolání synchronizace `stisk!`.



(a) : Světlo. **(b) :** Vypínač.

Obrázek 2.1: Model časovače světla.

Zde používané modely jsou vytvářeny v programu UPPAAL vyvíjeném na švédské univerzitě Uppsala a dánské univerzitě Aalborg. [6] Tento program umožňuje reprezentovat model jako síť časovaných automatů rozšířených o proměnné. Jednotlivé přechody, respektive hrany v systémech mohou obsahovat údaje o synchronizaci mezi systémy, podmínku pro vybrání přechodu,

informaci o relevanci dané hrany a také mohou obsahovat přiřazovací akce.

Synchronizace je potřebná v situacích, kdy jsou jednotlivé systémy na sobě navzájem závislé. Synchronizační řetězec je sestaven z názvu a identifikačního symbolu, který určuje, zda se jedná o hranu vyvolávající synchronizaci, nebo o hranu která je podmíněná jinou hranou. První zmíněná hrana má za názvem symbol „!“, druhá symbol „?“. V případě, že má být vybrána hrana se synchronizací, vždy musí zároveň dojít i k výběru odpovídající hrany s opačným symbolem. Pokud to není možné, tak nemůže být vybrána ani první zmíněná hrana.

Podmínky průchodu mohou kontrolovat stavové proměnné systému nebo globální proměnné celého modelu. Také mohou kontrolovat časové závislosti událostí. Tyto podmínky vedou k tomu, že nemusí být k dispozici v daném čase žádná možná hrana. Pomocí přiřazení mohou být při výběru hrany změněny hodnoty stavových proměnných či mohou být zresetovány některé časovače. Dále lze vyvolat akce na HiLu. Pomocí údajů o relevanci může autor modelu přidat hranám prioritu. V některých algoritmech procházení je tento údaj implementován.

Dodatečné informace nemají jen hrany, ale i jednotlivé uzly. Při příchodu do uzlu může proběhnout kontrola, zda je model nebo HiL v předpokládaném stavu. Opět jsou k dispozici proměnné systému i celého modelu. Také je kontrolován výstup z HiLu, jestli odpovídá předpokládanému stavu. Pokud tak není, test selhal a byla nalezena chyba.

2.1 Slovník

- Systém - Síť časovaných automatů. Celý model, který je procházen.
- Automat - Časovaný automat, odpovídá části systému.
- Uzel - Stav automatu.
- Hrana - Orientovaný přechod mezi dvěma uzly.

  Hrana s „!“ - Hrana vyvolávající synchronizaci, může být vybrána pouze pokud je k dispozici odpovídající hrana s „?“.

  Hrana s „?“ - Synchronizovaná hrana, která je podmíněna hranou s „!“.

Kapitola 3

Taster

Program Taster je vyvíjen na Fakultě elektrotechnické ČVUT. Je určen na generování testů pro HiL integrační testování a po připojení adaptérem k HiL testovacímu stavu i na samotné testování. Pracuje s modely vytvořenými v programu UPPAAL. Po načtení modelu nabízí výběr připojení pomocí adaptéru k HiLu či offline simulaci. Pro procházení modelu jsou k dispozici různé strategie průchodu s možností dalších nastavení. Lze uložit prošlé trasy, prohlížet je či opětovně procházet.

Procházení modelu probíhá v diskrétních krocích a lze nastavit jeho rychlost. V každém kroku je vybrána žádná, jedna nebo dvě hrany. Dvě hrany jsou vybrány právě tehdy, když dochází k synchronizaci.

V každém kroku simulace jsou hrany, které mají splněné podmínky pro vybrání a vedou z aktivních uzlů, rozděleny do dvou seznamů. První seznam, `AvailableEdges` obsahuje hrany bez synchronizace a hrany, které ji vyvolávají. Do druhého seznamu, `SyncableEdges` jsou přesunuty hrany podléhající synchronizaci. Pokud hrany se synchronizací nemají k dispozici svůj protějšek v druhém seznamu, jsou odstraněny. Tvorba těchto dvou seznamů je popsána v algoritmu 1. Ze seznamu `AvailableEdges` je poté vybrána jedna hrana podle zvolené strategie procházení modelu. Pokud vybraná hrana vyvolává synchronizaci, tak je z druhého seznamu podle dané strategie vybrána hrana splňující podmínku synchronizace.

Pro testování založené na modelu je záměr pokrýt co nejvíce různých scénářů neboli průchodů modelem a zároveň pokrýt pokud možno co nejširší rozsah různých situací. Nové scénáře lze ze stejného modelu vytvořit pomocí nové strategie. Také lze použít nedeterministickou strategii a pustit ji vícekrát. Nicméně takto získané průchody budou v jistém ohledu podobné. Důležité je také pořadí, ve kterém byly hrany a uzly navštíveny.

V rámci úprav nástroje Taster byly vytvořeny nové strategie průchodu. Pro potřeby simulací a generování testovacích scénářů přibyla možnost eliminovat časovou prodlevu mezi jednotlivými kroky.

Algorithm 1 Vytvoření seznamu dostupných hran a seznamu hran podmíněných synchronizací

```
1:  for active node N in model do
2:    for edge E outgoing from N do
3:      if E.Guard is fulfilled then
4:        if E.Sync.Contains(„?“) then
5:          SyncableEdges.Append(E)
6:        else
7:          AvailableEdges.Append(E)
8:        end if
9:      end if
10:   end for
11: end for
12: for edge E in AvailableEdges do
13:   if E.ContainsSync then
14:     if all edges in SyncableEdges not syncs with E then
15:       AvailableEdges.Remove(E)
16:     end if
17:   end if
18: end for
19: for edge E in SyncableEdges do
20:   if all edges in AvailabeleEdges not syncs with E then
21:     SyncableEdges.Remove(E)
22:   end if
23: end for
```

Kapitola 4

Urychlení simulace

Pro simulaci na rozdíl od testu s připojeným hardwarem není potřeba, aby byl dodržován pevný časový krok. Naopak je vhodné, aby byla pokud možno co nejrychlejší. V této kapitole jsou popsány okolnosti, které simulaci zpomalují, a poté způsoby, kterými lze simulace urychlit.

4.1 Faktory omezující rychlost simulace

Byly identifikovány tři faktory, které mají vliv na rychlost simulace. První je nastavený minimální časový krok diskrétní simulace. Při automatickém procházení je dodržován stejný interval mezi kroky a jednotlivé kroky jsou provedeny po splnění podmínky

$$T > s \cdot t_s, \tag{4.1}$$

kde T je doba běhu simulace, s je pořadové číslo kroku a t_s je pevný časový interval mezi kroky. Rychlost simulace lze nastavit v rozmezí 10 kroků za vteřinu po jeden krok za dvě vteřiny. Prodleva mezi kroky může tedy nabývat hodnot od 100 ms do 2 s. Tento čas odpovídá běhu reálného času, nikoliv časování v simulaci.

Druhým zpomalením jsou některé podmínky na hranách. Zmiňované hrany jsou podmíněné uplynutím času, respektive projitím daného počtu kroků v simulaci. V některých modelech z tohoto důvodu často není k dispozici žádná platná hrana k výběru a v daném kroku nedojde k vybrání hrany. Dle tabulky 4.1 může počet kroků, kdy není vybrána hrana, dosáhnout více než 90 % ze všech kroků v simulaci. Vynechání těchto podmínek má nicméně vliv na finální průchod systémem. Je tedy vhodné pouze na testování strategií, ale již ne na generování testů.

Poslední faktor je výkon počítače, na němž simulace běží, a výpočetní složitost programu. Toto se projeví vždy, když výpočet jednoho kroku bude trvat déle než je interval mezi kroky. Při rychlé simulaci není nastaven minimální doba mezi kroky, takže rychlost simulace je přímo závislá na náročnosti programu. Ta je ovlivněna konkrétní vybranou strategií a také množstvím vykreslovaných údajů. V každém kroku jsou totiž všechny systémy modelu překreslovány v grafickém prostředí nástroje Taster do aktuálního podoby.

	Počet	Podíl
Prázdné kroky	50635	93,6 %
Reálné kroky	3476	6,4 %
Celkem	54111	100 %

Tabulka 4.1: Počty kroků v průchodu systémem ovládání pátých dveří automobilu náhodnou strategií.

4.2 Opatření pro zrychlení

Zásadní vliv na rychlost simulace má nastavený časový krok. Pokud tato podmínka zůstane aktivní, potom i při vylepšení zbylých faktorů nedojde ke zrychlení. Po lokalizaci podmínky popsané v rovnici (4.1) ve zdrojovém kódu byla přidána alternativa, během které je další krok spuštěn ihned po ukončení předchozího kroku.

V určitých systémech ve většině kroků není vybrána žádná hrana, což je způsobeno hranovými podmínkami vztahujícími se k proměnným typu `clock`. Při analýze strategií nejsou podstatné kroky, ve kterých není vybírána hrana. Taková situace nastane, pokud jsou všechny dostupné hrany takto podmíněny. Tyto kroky byly eliminovány tak, že podmínky vázané na proměnné typu `clock` jsou vždy vyhodnocovány jako pravdivé. Tato úprava může způsobovat problémy při generování testů.

Třetím a posledním omezením je výkon počítače, jehož změny nejsou obsahem této práce. Je ale možné ovlivnit výpočetní náročnost programu, která je pro každou strategii jiná. Společným prvkem je pravidelné překreslování systému do aktuální podoby. Poslední úprava dovoluje toto vykreslování pozastavit.

Do nastavení nástroje Taster byla přidána záložka `Acceleration`, v níž lze nastavit jednotlivé prvky mající vliv na rychlost simulace. Při normálním běhu nástroje Taster jsou všechny aktivní a simulace není urychlována.

Po vypnutí položky `Display templates` se přestane model překreslovat do aktuální podoby. Bez položky `Use time guards` nejsou uvažovány časové podmínky na hranách. A po vypnutí poslední položky `Use timed step` se přestane dodržovat pevný interval mezi kroky.

4.3 Porovnání rychlostí

Rychlosti jednotlivých možností simulace byly porovnány v modelu ovládání pátých dveří automobilu za použití systematické strategie procházení s dobou běhu 100 s. Tato strategie byla vybrána pro svou vlastnost, že prochází vždy stejnou trasou. Jediný rozdíl mezi jednotlivými průchody bude v jejich délce. Použitý model je popsán v podkapitole 8.1.

V původním nastavení s nejmenším časovým krokem, tedy 100 ms, proběhl 1001 krok, z nichž pouze 70 bylo reálných. Při vypnutí časových podmínek bylo

reálných kroků 1000. V této variantě nezáleží, zda jsou modely vykreslovány, protože mezi jednotlivými kroky jsou dostatečné časové prodlevy.

V případě, že nebyly uvažovány časové podmínky ani pevný interval mezi kroky, tak bez vykreslování proběhlo 1547 reálných kroků. Pokud bylo ponecháno zapnuté vykreslování, tak proběhly 1022 reálné kroky. Jelikož nebyly uvažovány časové podmínky, nevznikly prázdné kroky.

Pokud zůstaly aktivní pouze časové podmínky, proběhlo 8219 kroků, z nichž reálných bylo pouze 545. V případě, že bylo ponecháno zapnuté také vykreslování, tak proběhly 8873 kroky a z toho 588 kroků bylo reálných.

Vypnutí vykreslování může procházení urychlit o 50 %, ale také mírně zpomalit. Jeho vliv je nejistý a zda jej použít je zapotřebí zvážit pro každou konfiguraci zvlášť. Bez pevného intervalu mezi kroky je možné dosáhnout osminásobného urychlení procházení. Při neuvažování časových podmínek lze docílit 22násobku počtu reálných kroků oproti základnímu průběhu.

Kapitola 5

Původní grafové algoritmy

Na začátku této práce byly v programu Taster implementovány 4 strategie průchodu, které vycházejí z grafových algoritmů, a dvě využívající strojového učení (ty nejsou zkoumány v této práci). Původní metody výběru hran jsou náhodná, systematická, vážená a heuristická.

Zde používaný stavový prostor se liší od běžné interpretace, jelikož některé hrany obsahují podmínky, které musí být splněny, aby dané hrany mohly být vybrány. Z tohoto důvodu se stavový prostor po každém průchodu hranou může změnit. Některé změny jsou závislé pouze na čase a ne na průchodech hranami.

Všechny zkoumané strategie používají seznamy `AvailableEdges` a `SyncableEdges`, které byly popsány v kapitole 3 a v algoritmu 1.

5.1 Systematická strategie

Tato strategie se řídí pouze počtem průchodů jednotlivých hran a hran, které mohou být vybrány díky synchronizaci s první zmíněnou hranou. Při rozhodování vybírá ze seznamu použitelných hran tu, která umožní vybrání hrany s nejnižším počtem průchodů. Při rovnosti vybírá hranu, která byla v seznamu výše. Pokud je ze seznamu `AvailableEdges` vybrána hrana, která vyvolává synchronizaci, tak je poté ze seznamu `SyncableEdges` vybrána odpovídající hrana s nejmenším počtem průchodů. Strategie je popsána algoritmem 2.

V této strategii není žádný zdroj náhodnosti. Při stejných počátečních podmínkách budou průchody vždy stejné. Toto je nevýhoda, jelikož strategie bez dalších úprav vrací pouze jeden průchod.

5.2 Vážená strategie

Vážená, též relevantní, strategie rozšíří seznam `AvailableEdges` tak, aby se v něm hrany vyskytovaly v počtu odpovídajícím jejich relevanci. V případě nulového údaje je taková hrana v seznamu jednou. Z takto upraveného seznamu je poté vybrána náhodně jedna hrana. U hran, které mají vyšší relevanci je tedy vyšší pravděpodobnost vybrání. Pokud byla vybrána hrana

Algorithm 2 Systematická strategie

```
minCount = MAX
for edge E1 in AvailableEdges do
  if E1.ContainsSync then
    for edge E2 in SyncableEdges do
      if E2.SyncWith(E1) and E2.Count < minCount then
        minCount = E2.Count
        Edge = E1
      end if
    end for
  end if
  if E1.Count < minCount then
    minCount = E1.Count
    Edge = E1
  end if
end for
if Edge.ContainsSync then
  minCount = MAX
  for edge E2 in SyncableEdges do
    if E2.SyncWith(Edge) and E2.Count < minCount then
      minCount = E2.Count
      Edge2 = E2
    end if
  end for
  return Edge, Edge2
else
  return Edge
end if
```

vyvolávající synchronizaci, tak se ze seznamu `SyncableEdges` náhodně vybere odpovídající hrana. Relevance se vyskytuje jen u hran vyvolávajících synchronizaci či hran bez synchronizace. Není tedy potřeba uvažovat zvláštní vybírání synchronizovaných hran. Tato strategie je podrobně popsána v algoritmu 3.

5.3 Náhodná strategie

Jedná se o základní způsob procházení. Ze seznamu `AvailableEdges` je hrana vybrána náhodně. Pokud vybraná hrana obsahuje synchronizaci, tak z odpovídajících hran v seznamu `SyncableEdges` je vybrána druhá hrana opět náhodně. Tento postup je znázorněn v algoritmu 4.

Algorithm 3 Vážená strategie

```
minCount = MAX
for edge E1 in AvailableEdges do
  if E1.Relevance then
    for E1.Relevance do
      AvailableEdges.Append(E1)
    end for
  end if
end for
Edge = AvailableEdges.Random
if Edge.ContainsSync then
  Edge2 = SyncableEdges.SyncWith(Edge).Random
  return Edge, Edge2
else
  return Edge
end if
```

Algorithm 4 Náhodná strategie

```
Edge = AvailableEdges.Random
if Edge.ContainsSync then
  Edge2 = SyncableEdges.SyncWith(Edge).Random
  return Edge, Edge2
else
  return Edge
end if
```

5.4 Heuristická strategie

Tato strategie pro výběr první hrany vytvoří seznam hran, které mají aktuálně nejmenší počet průchodů, popřípadě synchronizací související hrana má daný počet průchodů. Z tohoto seznamu poté vybere hranu, která má největší relevanci. Pokud takto byla vybrána hrana se synchronizací, tak z odpovídajících hran je poté vybrána ta, která byla vzata v nejméně případech a má největší relevanci.

Heuristická strategie je také popsána v algoritmu 5. Stejně jako v systematické strategii zde není žádný zdroj náhodnosti. Při stejných počátečních podmínkách budou průchody vždy stejné. Toto je nevýhoda, jelikož strategie bez dalších úprav vrací pouze jeden průchod.

Algorithm 5 Heuristická strategie

```
minCount = MAX
for edge E1 in (AvailableEdges or SyncableEdges) do
  if E1.Count < minCount then
    minCount = E1.Count
  end if
end for
for edge E1 in AvailableEdges do
  if E1.Count == minCount then
    SelectionEdges.Append(E1)
  else
    for edge E2 in SyncableEdges do
      if E2.SyncWith(E1) and E2.Count == minCount then
        SelectionEdges.Append(E1)
      end if
    end for
  end if
end for
SelectionEdges.SortByRelevances.Descending
Edge = SelectionEdges.First
if Edge.ContainsSync then
  minCount = MAX
  for edge E2 in SyncableEdges do
    if E2.SyncWith(Edge) and E2.Count < minCount then
      minCount = E2.Count
      Edge2 = E2
    end if
  end for
  return Edge, Edge2
else
  return Edge
end if
```

Kapitola 6

Nedostatky původních algoritmů

V této kapitole budou popsány obecné nedostatky, které byly nalezeny u původních strategií. U náhodné strategie lze vytknout právě to, že je zcela nedeterministická. Nelze u ní nijak předpovědět postup procházení. Vážená strategie je vylepšená verze náhodné. Při návrhu systému lze chování ovlivnit zvolenými relevancemi.

Dále budou popsány nedostatky u systematické a heuristické strategie. Za prvé, pokud jsou v modelu paralelní hrany, tak může nastat situace, kdy by některé části systému byly preferované před jinými. Pokrytí by poté nebylo rovnoměrné. Za druhé, průchody poloviny strategií závisely na pořadí, v jakém byly jednotlivé automaty, hrany a uzly definovány. V závislosti na pořadí mohly být některé části systému nedosažitelné. A za třetí, všechny strategie se rozhodovaly pouze na základě informací z daného aktuálního stavu.

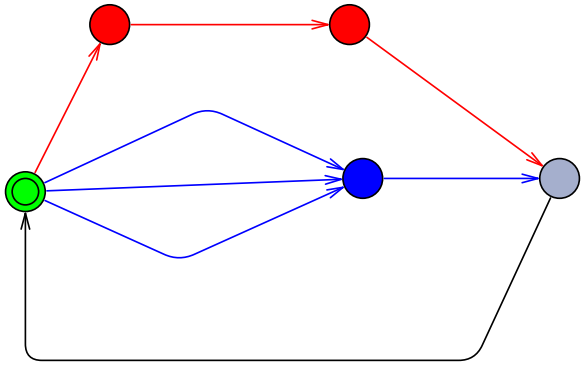
6.1 Paralelní hrany

V jednotlivých automatech se mohou vyskytovat paralelní hrany. Tedy situace, kdy více hran má stejný počáteční i koncový uzel. Takto lze modelovat například více tlačítek, která ovládají stejnou funkci.

Zmíněné dvě strategie vybíraly podle nějakého klíče hrany, které měly nejmenší počet průchodů. V každém uzlu se tedy snažily docílit toho, že všechny odchozí hrany budou vybrány stejně často. Toto mělo za následek, že část systému navazující na paralelní hrany byla vybrána častěji než část, ke které vedlo méně hran.

Tento nedostatek lze ilustrovat na obrázku 6.1. Při rozhodování v zeleném uzlu jsou k dispozici čtyři hrany, tři paralelní modré a jedna červená. Tyto barvy značí různé cesty či části systému, do kterých se lze dostat, pokud bude vybrána hrana dané barvy.

Vždy, když se zmíněné dvě strategie rozhodují v zeleném uzlu, tak s trojnásobnou frekvencí budou vybírat modré hrany oproti červené hraně. Jednotlivé větve mohou skrývat libovolně velkou část systému. Pokud by větší část byla v modré větvi, tak to mohlo být bráno jako výhoda. Lepší výběr by byl, pokud by strategie prvně vybrala cílový uzel na základě součtu průchodů odpovídajících si hran a následně z případných paralelních hran vybrala tu s nejmenším počtem.

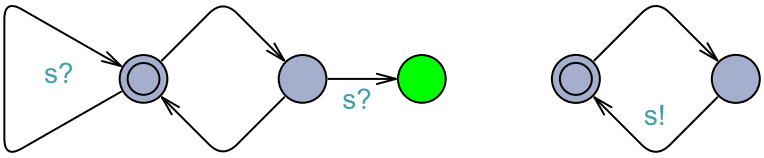


Obrázek 6.1: Situace, ve které by měly být uvažovány paralelní hrany.

6.2 Pořadí definic

Obě strategie používají pro výběr ideální hrany nějakou hodnotící funkci. V ideálním případě by pro každou hranu dávala jiný výsledek a bylo by jasné, kterou hranu vybrat. Při procházení systémů ale dochází k situacím, kdy stejný výsledek má více hran. Pokud bude v jednom stavu existovat více hran se stejným výsledkem, který bude pro danou strategii optimální, tak bude vybrána první hrana s tímto výsledkem. Pořadí, v jakém jsou hrany kontrolovány je dáno definicí jednotlivých automatů a celého systému, a po celou dobu je neměnné.

Systém, který je jednoduše dosažitelný, se může stát nepřístupným. Záleží pouze na pořadí, v jakém byly automaty a hrany definovány. Na obrázku 6.2 jsou dva automaty ze systému, ve kterém záleží na pořadí definicí. První automat 6.2a obsahuje zeleně zvýrazněný cílový uzel. Druhý 6.2b pouze vyvolává synchronizaci s prvním.



(a) : Automat s cílovým stavem. **(b) :** Synchronizační automat.

Obrázek 6.2: Systém, u kterého záleží na pořadí definic.

Problém se objeví, když bude cílový automat definovaný před synchronizačním. V takovém případě při stejném ohodnocení vícero hran budou vždy preferovány hrany z hlavního automatu. Pokud nastane situace, že všechny hrany bez té do cíle budou mít stejný počet projití a aktuálními uzly budou dva zvýrazněné, tak se strategie nebude schopná dostat do cílového uzlu.

V prvním kroku jsou možné horní hrany z obou automatů. Vzhledem k pořadí definic bude vybrána hrana z cílového automatu. Nyní by pro cestu do cíle stačilo, aby synchronizační automat umožnil synchronizaci. Před hranou vedoucí k synchronizaci je ale opět preferována spodní hrana z prvního

automatu. V tuto chvíli již všechny možné hrany z cílového automatu mají více průchodů, než hrany v synchronizačním. Dojde tedy k synchronizaci a systém se dostal do obdobného stavu jako na začátku. Pokud by automaty byly definované v opačném pořadí, tak by dosažení cíle nebylo problém.

Tento problém by bylo možné vyřešit dvěma způsoby. Úpravou vyhodnocující funkce tak, aby vždy určila právě jednu hranu. Druhá možnost spočívá v tom, že výsledná hrana bude vybrána náhodně z hran, které všechny mají nejlepší výsledek vyhodnocení.

6.3 Nízká informovanost

Původní strategie byly málo informované. Chyběly jim veškeré informace o kontextu aktuálního stavu a rozhodovaly se pouze na základě znalostí o možných hranách. První informace, která jim chyběla již byla popsána, jednalo se o paralelní hrany. Další zdroj informací je cesta, po které strategie dostala do současného stavu. Také lze odvodit, do jakého stavu se systém dostane, pokud bude vybrána jistá hrana.

Kapitola 7

Nové strategie

Na základě zjištěných nedostatků byly vytvořeny dvě nové strategie procházení modelu. Stejně jako původní strategie, i nové používají seznamy dostupných hran, které jsou popsány v kapitole 3.

7.1 Strategie zohledňující předchozí hrany

Při testování záleží na pořadí, v jakém jsou uzly procházeny. První přidaná strategie systematicky prochází systém na základě informace o nejmenším počtu průchodů ne jednotlivých hran, ale trojice hran.

První přidaná strategie se od systematické se liší tím, že nezkoumá počet průchodů jednou hranou, ale celkem trojicí navazujících hran. Trojice obsahuje dvě hrany, které předcházely příchodu do uzlu, a jednu hranu, která bude vybrána. Všechny uvažované hrany vždy patří do stejného automatu. Počty se určují pro jednotlivé trojice, takže pokud daným uzlem prošla po jiné hraně, tak se toto neprojeví.

Strategie dovoluje vybrat hranu, která neodpovídá nejvýhodnější trojici, pokud tato hrana vyvolává synchronizaci s hranou, která má menší počet průchodů než nejvýhodnější trojice. Z hran s odpovídající synchronizací je poté vybrána hrana s nejmenším počtem průchodů. Celá strategie je také popsána v algoritmu 6.

7.2 Strategie prohledávacího stromu

V aktuálním uzlu nemusí být zřejmé, která hrana vede k cílovému stavu modelu, který ještě nebyl otestován. Cílový stav se může objevit až po několika krocích, během kterých ale strategie může zvolit jinou hranu a tento stav mine.

Procházení modelu lze brát jako stavový prostor. Stav je poté seznam aktivních uzlů včetně hodnot všech proměnných. Akce či přechody jsou jednotlivé hrany, které je možné projít. Počáteční stav je seznam počátečních uzlů včetně hodnot proměnných po inicializaci a cílové stavy určuje konkrétní strategie procházení.

Algorithm 6 Strategie zohledňující předchozí hrany

```
minCount = MAX
for edge E1 in AvailableEdges do
  if E1.LastEdges.Count < minCount then
    minCount = E1.LastEdges.Count
    Edge = E1
  end if
  if E1.ContainsSync then
    for edge E2 in SyncableEdges do
      if E2.SyncWith(E1) and E2.Count < minCount then
        minCount = E2.Count
        Edge = E1
      end if
    end for
  end if
end for
if Edge.ContainsSync then
  minCount = MAX
  for edge E2 in SyncableEdges do
    if E2.SyncWith(Edge) and E2.Count < minCount then
      minCount = E2.Count
      Edge2 = E2
    end if
  end for
  return Edge, Edge2
else
  return Edge
end if
```

Všechny možné stavy systému, tedy možné kombinace aktivních uzlů a hodnot proměnných dávají dohromady stavový prostor X. Z aktivního stavu x se akcí u, kterou může být projití hrany nebo aktualizace proměnných, přejde do stavu x'. Vztah mezi stavy a akcemi udává přechodová funkce f. Přechodová rovnice mezi stavy lze napsat jako

$$x' = f(x, u), \tag{7.1}$$

kde proměnné x, x', u a funkce f byly popsány výše. Akční prostor všech akcí, které mohou proběhnout ve stavu x lze označit $U(x)$. [7]

Strategie prohledávacího stromu před výběrem následující hrany ze stavu x vyzkouší všechny akce z akčního prostoru čímž získá seznam stavů v první generaci

$$X_1 = \{f(x, u) | \forall u \ \in U(x)\}. \tag{7.2}$$

Všechny stavy z množiny X_1 poté rozšíří stejně jako původní stav x, čímž dostane množinu druhé generace X_2.

Všechny stavy v druhé generaci jsou ohodnoceny na základě počtu průchodů hran, které byly použity na dostání se do daného stavu, a hran, po kterých je možné stav opustit. Hodnotí se hrana s nejmenším počtem průchodů a dále ve které generaci se tato hrana nachází. V případě paralelních hran se prvně hodnotí paralelní hrany jako celek a až v případě, že tyto mají nejméně průchodů, tak se započítá hrana projitá nejméně. Tato strategie je také popsána algoritmem 7.

Algorithm 7 Strategie prohledávacího stromu

```
StatesToExpand.Enqueue(ActualState)
while not StatesToExpand.Empty() do
  State s = StatesToExpand.Pop()
  if s.Depth == MaxDepth  then
    FinalStates.Enqueue(s)
  else
    PrepareAvailableAndSyncEdges() // see alg. 1
    for edge E in (AvailableEdges or SyncableEdges) do
      DestinationCount[E.Destination] += E.Count
    end for
    for edge E1 in AvailableEdges do
      minCount = MAX
      if E1.ContainsSync then
        for edge E2 in SyncableEdges do
          if E2.SyncWith(E1) and E2.Count < minCount then
            minCount = E2.Count
            Edge = E1
          end if
          if E1.Count < minCount then
            minCount = E1.Count
            Edge = E1
          end if
          StatesToExpand.Enqueue(UpdateActualState())
        end for
      else
        StatesToExpand.Enqueue(UpdateActualState())
      end if
    end for
  end if
end while
minParalel
for State s in FinalStates do
  if s.MinCount < MinCount then
    MinCount = s.MinCount
    Edge = s.Edge
  else if s.MinCount == MinCount and s.step < MinStep then
    MinStep = s.step
    Edge = s.Edge
  end if
  if Edge.ContainsSync then
    Edge2 = s.SyncedEdge
  end if
end for
return  Edge, Edge2
```

Kapitola 8
Použité modely

V této kapitole jsou popsány dva systémy, jejichž modely byly v práci použity. První model byl vytvořen na FEL ČVUT a popisuje chování automatického ovládání pátých dveří automobilu. Druhý model vytvořil Michal Veselka. Jedná se o bezklíčové otevírání vozu KESSY[8]. Druhý model byl pro tuto práci upraven.

8.1 Ovládání pátých dveří automobilu

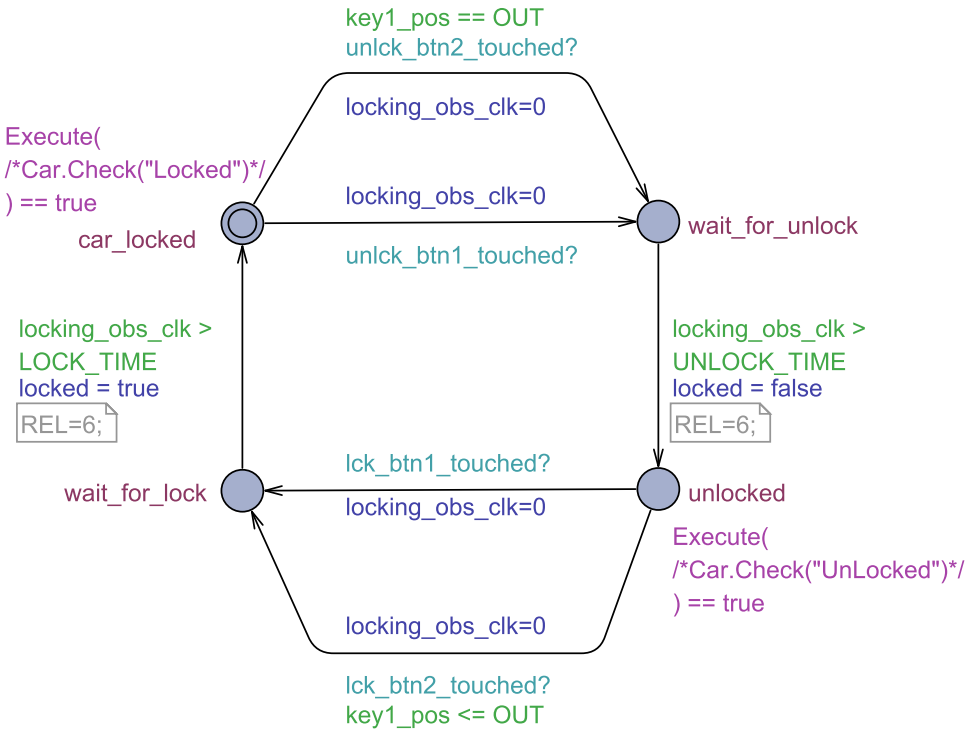
Tento model se skládá z šesti tlačítek, klíče, zámku a motoru, který ovládá páté dveře. Uživatel má k dispozici čtyři tlačítka ovládající páté dveře a dvě tlačítka, kterými může odemykat a zamykat celý automobil. Pokud jsou splněné podmínky, automobil je odemčený a je stisknuto tlačítko ovládající páté dveře, otevřou se. Obdobně funguje jejich zavírání. Pro uzamčení automobilu nesmí být klíč detekován uvnitř vozu.

Systém ovládání pátých dveří automobilu je složen ze tří časovaných automatů a obsahuje údaje o relevanci hran. První automat kontroluje jednotlivá tlačítka. Druhý ovládá odemykání a zamykání vozu včetně toho, že akce samotná trvá určitý čas. Poslední automat popisuje samotné chování pátých dveří. Ty mohou být otevřeny pouze pokud je vůz odemčen. Přechod mezi otevřeným a zavřeným stavem pátých dveří stejně jako u odemykání vozu trvá určitou dobu. Páté dveře mohou být uzavřeny i po uzamčení vozu.

V tomto modelu dochází k situacím, ve kterých není k dispozici žádná hrana, protože všechny dostupné hrany nesplňují časové podmínky pro projití. Druhý zmíněný automat je zobrazen na obrázku 8.1. Celý model je k dispozici na přiloženém CD.

8.2 Model KESSY

KESSY je zkratka z anglického označení Keyless-Entry-Start-and-Exit-System. Jedná o stále častější systém identifikace řidiče, kdy pro odemknutí vozu ani pro nastartování není potřebná interakce řidiče s klíčem. Kromě klasické funkce klíče je také možné vůz odemknout dotykem kliky dveří ve chvíli, kdy je klíč v blízkosti vozu.



Obrázek 8.1: Automat ovládající zamykání a odemykání vozu. Jedná se o jeden z automatů z modelu pátých dveří, který byl vytvořen na FEL ČVUT.

Systém popisuje automobil, který může být odemknut a nastartován dvěma různými klíči. Na každém jsou tři tlačítka, první odemyká automobil, druhé jej zamyká a třetí otevírá páté dveře. Nastartování motoru se provádí stiskem tlačítka ve voze. Klíče fungují bezdrátově a pro automatické otevírání i startování je potřeba, aby alespoň jeden z nich byl v dosahu antén vozu.

Tento systém je složen z jedenácti časovaných automatů a původně neobsahoval údaje o relevanci. Ty byly doplněny v průběhu práce. Tento systém je také k dispozici na přiloženém CD.

Kapitola 9

Porovnání strategií

Na systému KESSY je ukázán vývoj rozložení projitých hran do jednotlivých automatů v čase. Na systému automatického ovládání pátých dveří automobilu je zobrazeno pokrývání hran a uzlů v čase.

9.1 Volání automatů

Pokud analyzovaný systém sestává z několika automatů, lze znázornit vývoj rozdělení vybraných hran mezi automaty v čase. Pokud se průběh bude opakovat, tak je velmi pravděpodobné, že se opakují stejné kroky. Ze změn v rozložení lze také odhalit, že nějaký automat přestal být vybírán či případně začal být procházen až po jisté době.

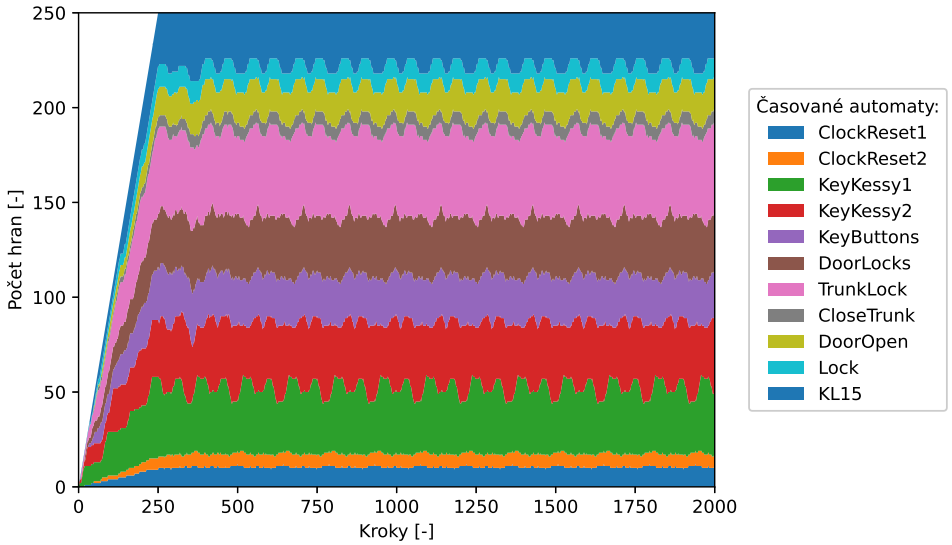
Takto budou strategie porovnány na KESSY modelu. V následujících grafech je pro každý krok zobrazeno rozdělení posledních 250 projitých hran mezi jednotlivé automaty.

Na grafech 9.1, 9.2 je vidět, že průchody heuristické, respektive systematické strategie po určitém počtu kroků mají opakující se rozložení automatů. Lze tedy předpokládat, že zmíněné průběhy se opakují. Opakování začíná přibližně na 700. hraně a je dlouhé 144 hran.

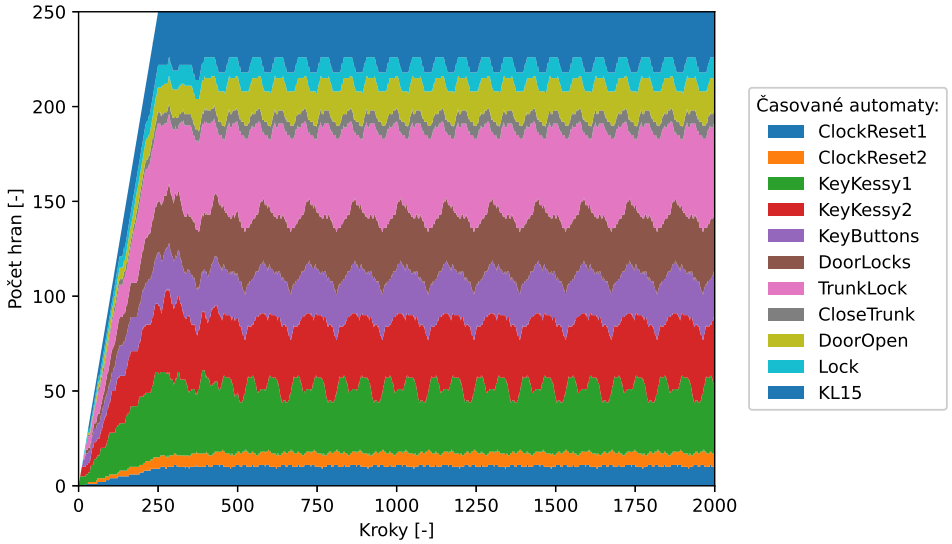
Zbylé dvě původní strategie nejsou deterministické a tedy by se neměl objevit opakující se vzor. Obě strategie vybírají hrany náhodně s rozdílem, že u vážené metody mají některé zvýšenou pravděpodobnost vybrání. Na grafu 9.3 je zobrazen průběh náhodné strategie a graf 9.4 zobrazuje průběh vážené strategie. Jejich průběhy se dle očekávání neopakují a jiné rozložení automatů ukazuje na vzájemně různé průchody.

Průběh nově vytvořené strategie prohledávacího stromu je na grafu 9.5. Strategie, která zohledňuje dvě předchozí hrany má průběh zobrazený na grafu 9.6.

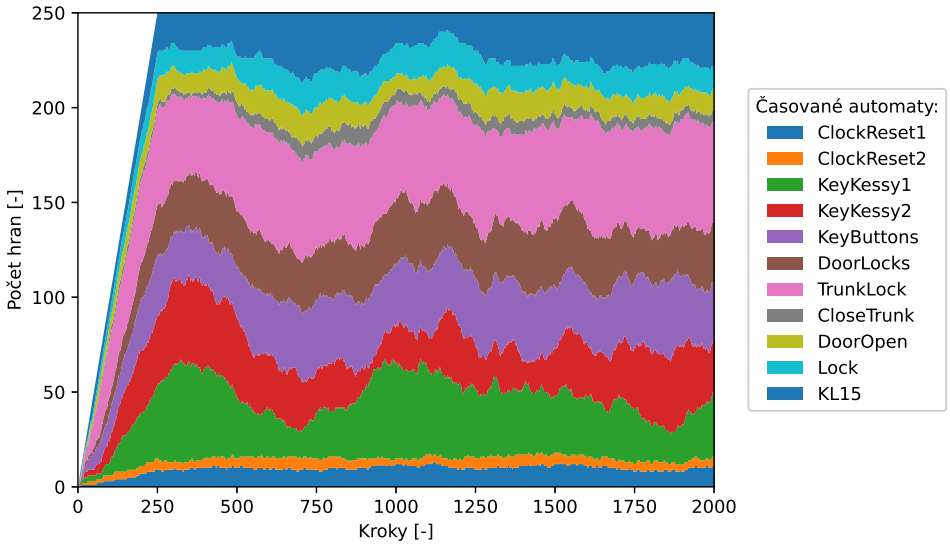
Dvě nové strategie také neobsahují opakující se vzor, i když strategie zohledňující předchozí hrany je deterministická. V porovnání s ostatními strategiemi má strategie prohledávacího stromu podobné rozložení jako systematická a heuristická metoda. Druhá nová strategie má ze všech průběhů nejvýraznější změny rozložení v čase.



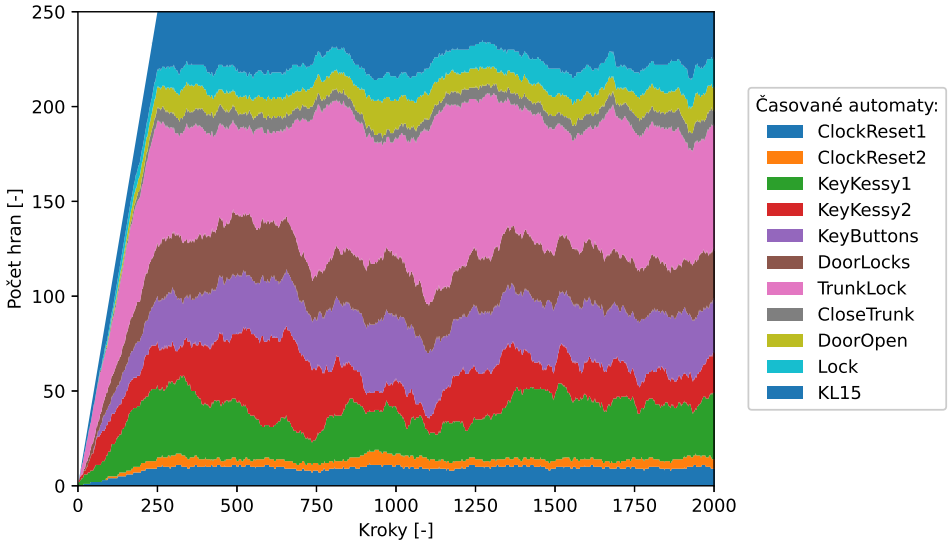
Obrázek 9.1: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu s heuristickou strategií.



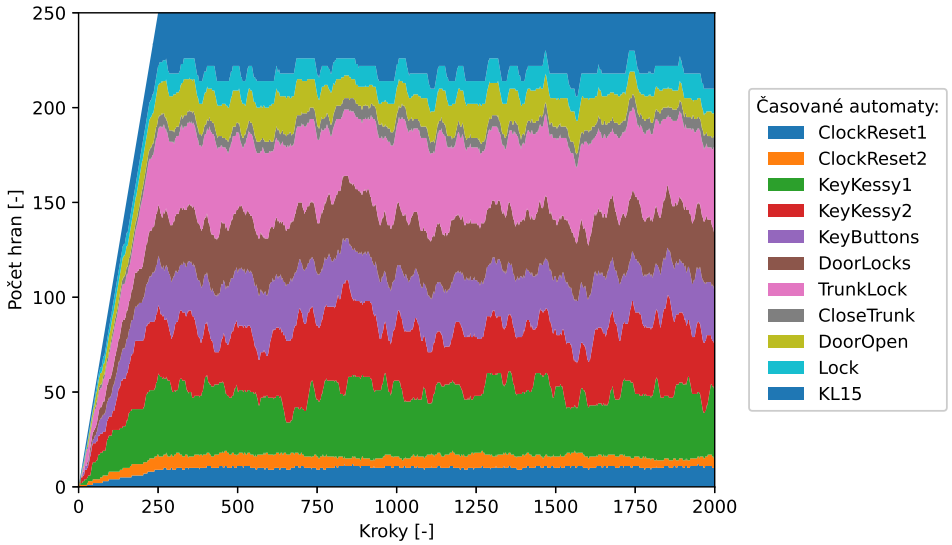
Obrázek 9.2: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu se systematickou strategií.



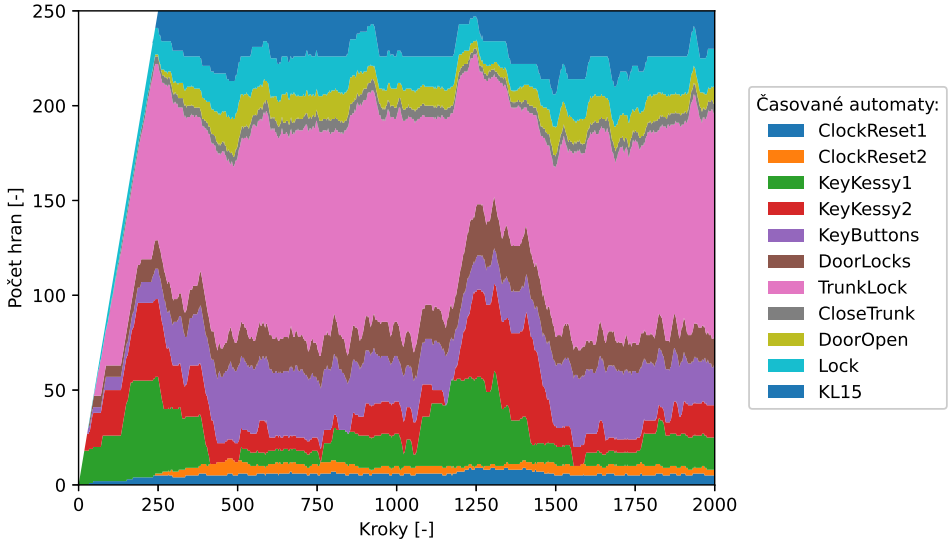
Obrázek 9.3: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu s náhodnou strategií.



Obrázek 9.4: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu s váženou strategií.



Obrázek 9.5: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu se strategií prohledávacího stromu.



Obrázek 9.6: Počet hran z jednotlivých automatů za posledních 250 kroků při průchodu se strategií zohledňující předchozí hrany.

9.2 Pokrytí uzlů a hran v čase

Pro prezentaci schopnosti pokrýt systém byl použit model automatického ovládání pátých dveří automobilu. Tento model byl vybrán, protože jej některé strategie nejsou schopné pokrýt celý.

Pro průchody byl použitý režim bez pevné délky kroku, ale časové podmínky zůstaly v platnosti. Bez podmínek by pokrytí strategiemi bylo vyšší, ale dané průchody by poté nedávaly smysl. Veškeré grafy zobrazují průběh pokrytí hran a uzlů během prvních 5000 kroků.

Na grafu 9.7 je pokrytí hran a uzlů celého systému heuristickou strategií v čase. Na grafu 9.8 je tento průběh se systematickou strategií. Obě strategie mají jiný průchod systémem, ale průběh pokrytí je u obou stejný. Obě se dokázaly pokrýt pouze 71 % hran a 92 % uzlů.

Pokrytí náhodnou strategií je na grafu 9.9. Dokázala pokrýt všechny uzly a 84 % hran. Vážená strategie je znázorněna na grafu 9.10. Ta také dokázala pokrýt všechny uzly a z hran jich využila 87 %. Vzhledem k náhodné podstatě obou strategií různé průběhy dosahují různých výsledků.

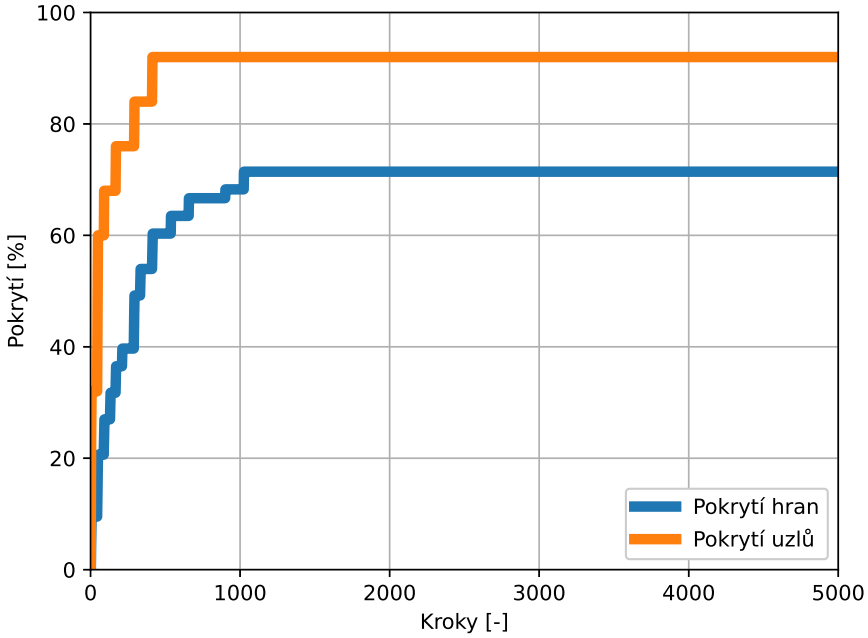
Prohledávací strom pokryl všechny uzly v systému a 84 % hran. Toto je zobrazené na grafu 9.11. Strategie zohledňující předchozí hrany pokryla také všechny uzly, ale jen 83 % hran.

9.3 Shrnutí porovnání

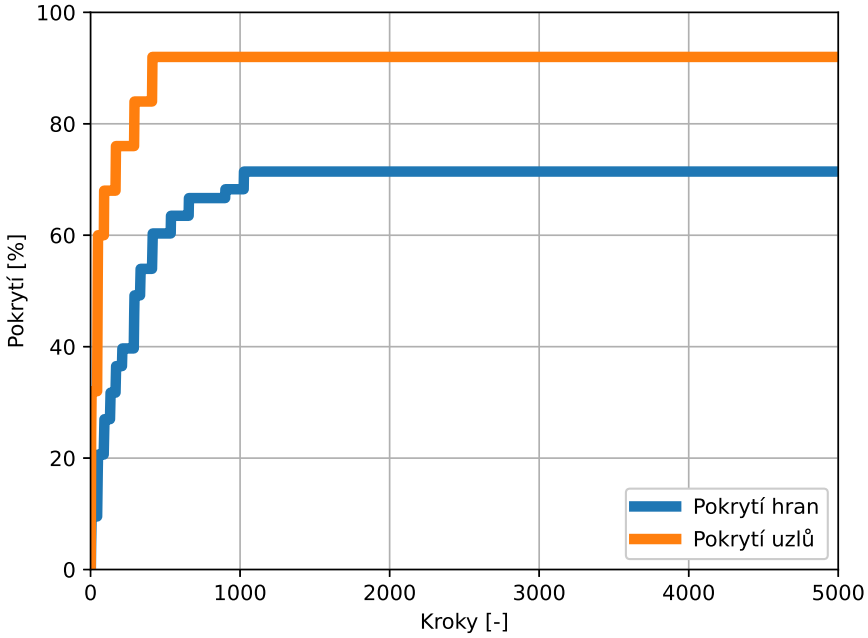
Další analýza zjistila, že se heuristická a systematická strategie u obou modelů obě zacyklily. Hrany v průchodu KESSY modelem se začaly opakovat přibližně po 250. hraně. Sekvence se poté opakovala vždy po 144 hranách. U modelu pátých dveří byla první hrana z cyklické sekvence kolem 50. kroku a celá sekvence obsahovala 2928 kroků. Vzhledem k tomu, že tento model vytváří prázdné kroky, tak hran v sekvenci bylo pouze 192.

V modelu pátých dveří tyto dvě strategie také nedokázaly pokrýt všechny uzly ani hrany. Oba nedostatky jsou způsobené problémem pořadí definic, který je popsaný v podkapitole 6.2. Nově vytvořená deterministická strategie zohledňující předchozí hrany těmito nedostatky netrpí.

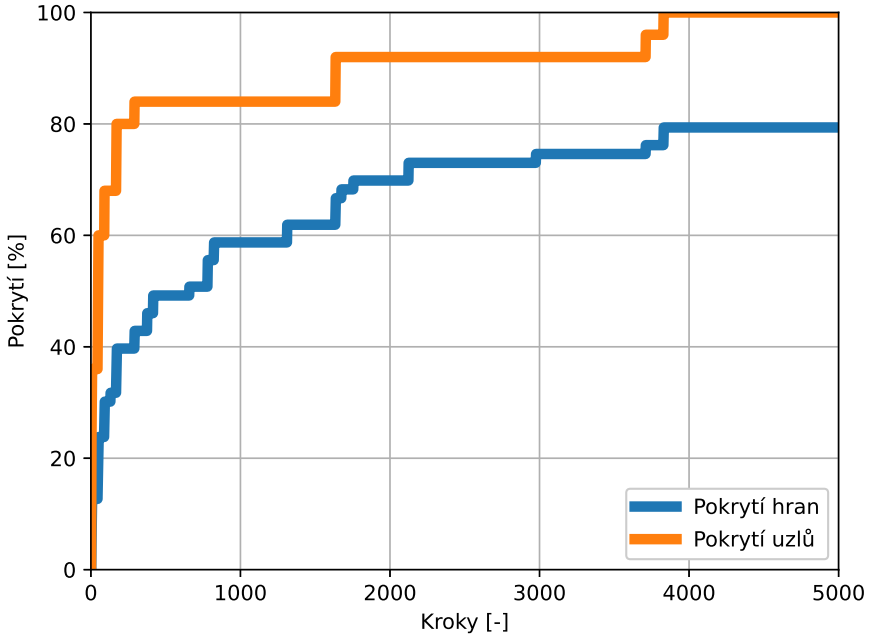
Podle rozložení automatů v čase lze usoudit, že průchod prohledávacího stromu bude mít podobný charakter jako systematické a heuristické průchody. Na rozdíl od nich ale dokáže pokrýt větší část modelu a nezacyklí se. Strategie zohledňující předchozí hrany prochází modely systematicky, ale rozložení hran mezi automaty v jejím průchodu je od ostatních odlišné.



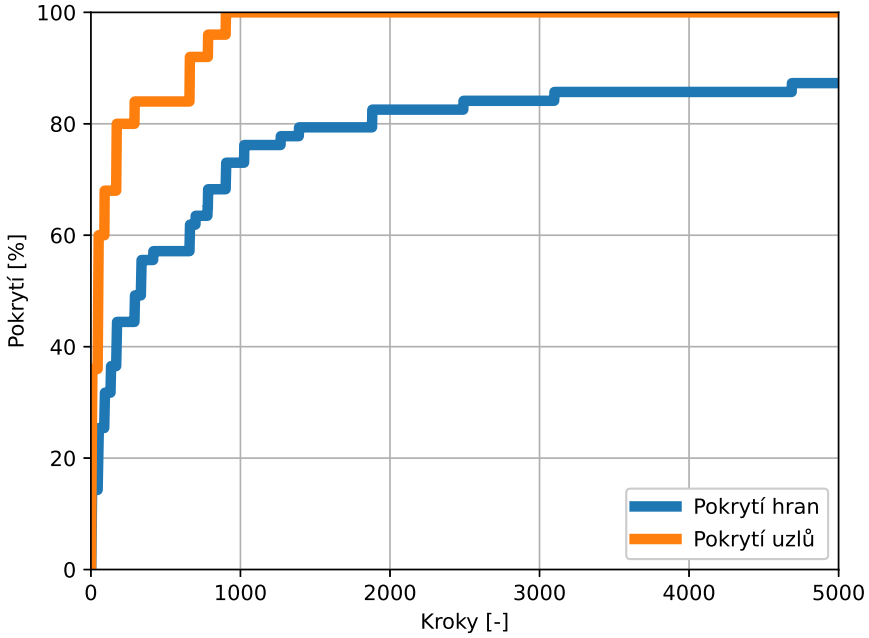
Obrázek 9.7: Pokrytí hran a uzlů v čase při použití heuristické strategie.



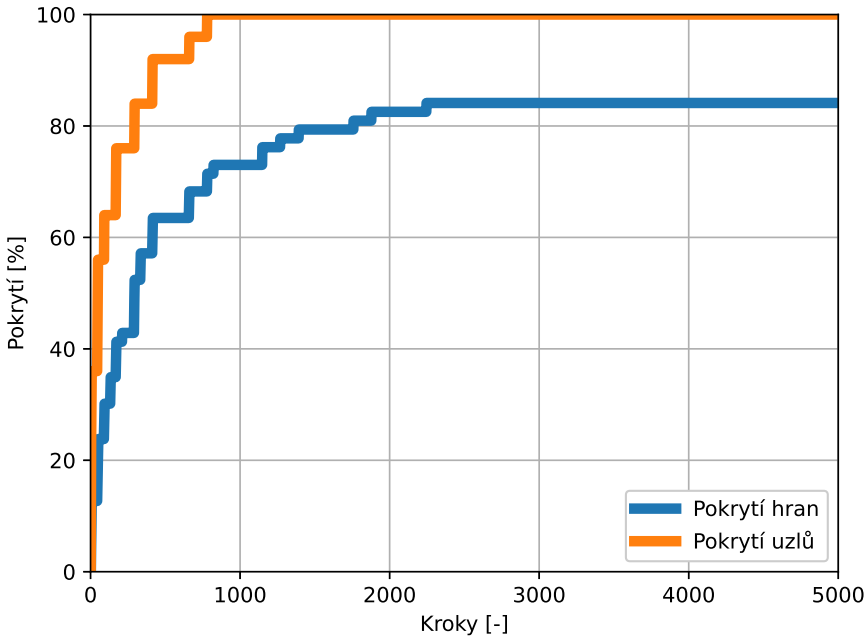
Obrázek 9.8: Pokrytí hran a uzlů v čase při použití systematické strategie.



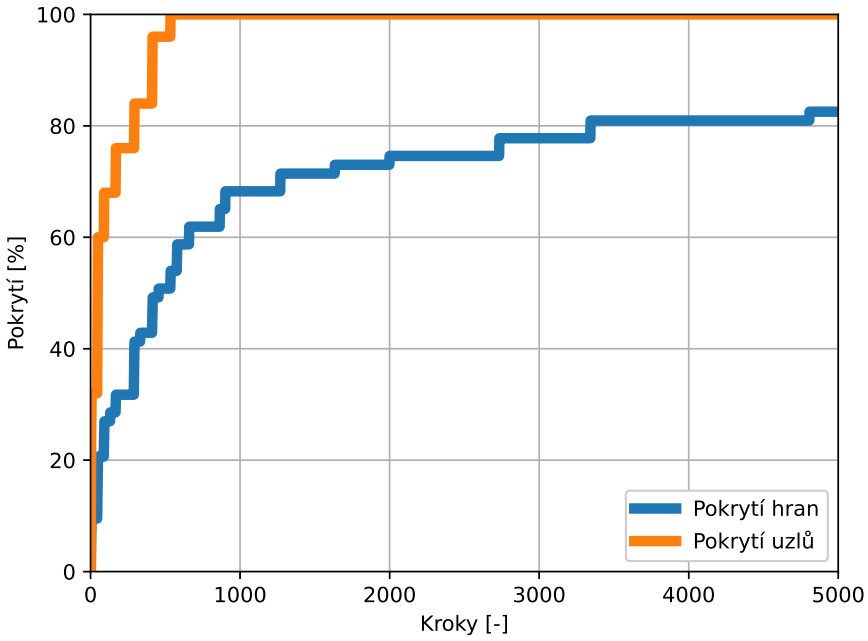
Obrázek 9.9: Pokrytí hran a uzlů v čase při použití náhodné strategie.



Obrázek 9.10: Pokrytí hran a uzlů v čase při použití vážené strategie.



Obrázek 9.11: Pokrytí hran a uzlů v čase při použití strategie prohledávacího stromu.



Obrázek 9.12: Pokrytí hran a uzlů v čase při použití strategie zohledňující předchozí hrany.

Kapitola 10

Diskuze k budoucí práci

Ve všech strategiích se zatím pro hodnocení hran používají různé počty průchodů, náhoda či váhy určené hodnotou relevance. Žádná strategie zatím nezkoumá, kdy naposledy byla konkrétní hrana projita. Tento údaj může být základem nové strategie nebo doplnit stávající. Obě nově vytvořené strategie je možné změnou nastavení přimět k jinému procházení systémem. Vliv těchto nastavení je možné dále vyšetřit.

U strategie zohledňující předchozí hrany se v současné podobě používají tři hrany v konfiguraci $2 + 1$. Tedy dvě hrany, po nichž se automat dostal do aktuálního stavu, a jedna, po které může odejít. Lze vyzkoušet i jiné kombinace, jako třeba $4 + 1$ či $1 + 1$. Varianta $0 + 1$ odpovídá původní systematické strategii.

Také lze prozkoumat vliv velikosti automatu a počtu hran ve strategii na průchody. Nebo uvažovat pro různě velké automaty v systému různé konfigurace této strategie. Další možností je použít jako základ místo systematické strategie strategii heuristickou.

U strategie prohledávacího stromu je také možné zjistit, zda má hloubka prohledávání vliv na rychlost simulace a výsledné průchody. Dále lze vyzkoušet s prohledávacím stromem jiná hodnotící kritéria. Opět připadá v úvahu obdoba heuristické strategie či použití údaje o době od posledního průchodu. S většími úpravami je možné hodnotit všechny hrany v nalezené cestě a nejen jednu nejlepší.

Kapitola 11

Závěr

V rámci této práce byl nástroj Taster rozšířen jak možnostmi nastavení, tak i novými strategiemi. Nové strategie je možné využít při generování testů pro integrační testování automobilů.

Pro snazší a rychlejší analýzu jednotlivých strategií byl vytvořen simulační mód, který umožňuje rychlejší procházení modelů. Zrychlený mód je nastavitelný pomocí tří položek v nastavení. První položka umožňuje potlačit pevný interval mezi jednotlivými kroky. Při navolení tedy kroky probíhají ihned po ukončení předchozího kroku. Druhá položka dovoluje nerespektovat časové podmínky na hranách. Toto je ale vhodné jen pro analýzu algoritmů, nikoli pro generování testů. Nerespektování časových podmínek má totiž vliv na samotné průchody. Poslední položka deaktivuje vykreslování aktuálního stavu systému.

S pomocí zmíněného rychlejšího procházení proběhla analýza původních strategií a byly odhaleny jejich nedostatky. Na popis strategií tedy navazuje popis jejich nedostatků. Se znalostí původních strategií včetně nedostatků byly navrženy dvě nové strategie.

Jedna strategie se stejně jako systematická strategie řídí podle nejmenší počtu průchodů. Liší se ale v tom, jak průchody počítá. Na rozdíl od systematické strategie nepočítá průchody pro každou hranu, ale pro trojice hran. Tato trojice sestává ze dvou hran, které předcházely příchodu do aktuálního uzlu, a jedné, po které může strategie pokračovat.

Druhá strategie využívá prohledávací strom. V tomto stromu projde všechny hrany do hloubky dva. Při průchodu hledá hrany s nejmenším počtem průchodů. Pokud by takových hran bylo více, tak je vybrána taková, která je blíže původnímu uzlu.

Poté, co jsou popsány i nové strategie, tak jsou všechny porovnány na dvou modelech. V diskuzi jsou navrženy další možné úpravy algoritmů a také je navrženo nové kritérium, podle něhož by bylo možné vybírat hrany k projití.

Literatura

[1] KRECL, Ing. Jaromír. *HIL simulace jako prostředek pro testování řídících jednotek v automobilu* [online]. In: . 2003 [cit. 2020-07-19]. Dostupné z: http://dsp.vscht.cz/konference_matlab/matlab03/krecl.pdf

[2] UTTING, Mark a Bruno LEGEARD. *Practical model-based testing: a tools approach*. San Francisco: Morgan Kaufmann Publishers, 2007. ISBN 9780123725011.

[3] BEHRMANN, Gerd, Alexandre DAVID a Kim G. LARSEN. A Tutorial on Uppaal. BERNARDO, Marco a Elavio CORRADINI, ed. *Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*. Springer-Verlag Berlin Heidelberg, 2004, s. 200-236. DOI: 10.1007/b110123. ISBN 978-3-540-30080-9. Dostupné také z: http://people.cs.aau.dk/~adavid/publications/21-tutorial.pdf

[4] BENGTSSON, Johan a Wang YI. Timed Automata: Semantics, Algorithms and Tools. In: *Lectures on Concurrency and Petri Nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 2004, s. 87-124. Lecture Notes in Computer Science, 3098. DOI: 10.1007/978-3-540-27755-2_3. ISBN 978-3-540-22261-3. Dostupné také z: http://link.springer.com/10.1007/978-3-540-27755-2_3

[5] ALUR, Rajeev a David L. DILL. A theory of timed automata. In: *Theoretical Computer Science*. 2. 1994, s. 183-235. DOI: 0304-3975(94)90010-8. ISSN 0304-3975. Dostupné také z: http://www.sciencedirect.com/science/article/pii/0304397594900108

[6] UPPAAL. *UPPAAL* [online]. Uppsala, Aalborg, 2019, 2019 [cit. 2020-07-01]. Dostupné z: http://www.uppaal.org/

[7] LAVALLE, Steven Michael. *Planning algorithms* [online]. Cambridge: Cambridge University Press, 2006, 20.4.2012, 842 s. [cit. 2020-07-30]. Dostupné z: http://lavalle.pl/planning/

[8] VESELKA, Michal. *Integrační testování metodou Model-Based Testing - případová studie*. 2019. Dostupné také z: `https://dspace.cvut.cz/handle/10467/83294`. Diplomová práce. Fel ČVUT. Vedoucí práce Ing. Jan Sobotka, Ph.D.

Příloha A

Obsah přiloženého CD

- Bakalářská práce ve formátu PDF
- Model otevírání pátých dveří automobilu
- Upravený model KESSY systému