České vysoké učení technické v Praze
Fakulta elektrotechnická

Diplomová práce

Simulátor transportních sítí

Vypracoval: Bc. Miloslav Stibor
Vedoucí práce: Ing. Michal Kutil

České vysoké učení technické v Praze − Fakulta elektrotechnická

Katedra řídicí techniky **Školní rok:** 2006/2007

Z A D Á N Í D I P L O M O V É P R Á C E

Student: Bc. Miloslav S t i b o r

Obor: Technická kybernetika

Název tématu: Simulátor transportních sítí

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se s formalismem Petriho sítí a jeho využitím pro řešení problémů z oblasti transportních sítí.
2. Implementujte knihovnu objektů a metod pro podporu formalismu Petriho sítí v objektově orientovaném jazyku C++.
3. Proveďte příkladové simulace z oblasti transportních sítí, využívající výše zmíněnou knihovnu.

Seznam odborné literatury: Dodá vedoucí práce

Vedoucí diplomové práce: Ing. Michal Kutil

Datum zadání diplomové práce: zimní semestr 2006/2007

Termín odevzdání diplomové práce: leden 2008

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze, dne 21.02.2007

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. …….
podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu mé práce Ing. Michalu Kutilovi, bez jehož pomoci, rad a připomínek by tato práce nemohla vzniknout. Můj vděk patří také všem, jenž mě v mé práci vytrvale podporovali.

Téma

Simulátor transportních sítí

Abstrakt

Tato práce popisuje návrh a implementaci obecného simulátoru transportních sítí, založeného na formalizmu diskrétních a spojitých Petriho sítí. Simulátor je realizován v podobě knihovny tříd a metod objektově orientovaného programovacího jazyku *C++*. Poskytuje tak uživateli oporu formalizmu Petriho sítí v tomto prostředí. Dále byla implementována jednoduchá koncová aplikace, využívající tuto knihovnu.

Práce rovněž popisuje sestavení modelů základních prvků transportní sítě městské dopravy pomocí spojitých Petriho sítí. Řadu konkrétních modelů pak uvádí v rozsáhlé knihovně. Za použití implementované koncové aplikace, je provedena příkladová simulace na reálných datech z oblasti městské dopravy. Simulovaná oblast je částí regionu Praha − Smíchov o deseti navazujících křižovatkách a použitá data odpovídají provozu za jeden pracovní den.

Theme

Transport nets simulator

Abstract

This work describes design and implementation of general transport nets simulator based on discrete and continuous Petri Net formalism. Simulator is realized in the form of standalone library for *C++*. It supports the Petri Net formalism in this object oriented programming language. Further a simple front-end application based on this library is implemented.

Work describes design of models of basic urban traffic elements also. Set of concrete models is presented in the form of pre-defined library. With implemented simulator the simulation of urban traffic is performed on real data of the part of Prague − Smichov region. The region consists of ten crossroads and used data are corresponding to the traffic during one day period.

Obsah

Seznam obrázků

Seznam textových souborů

1. Úvod

V dnešní uspěchané době tvoří rčení: „Dvakrát měř, jednou řež!" jedno ze základních pravidel návrhu a realizace projektů v mnoha oborech. S rostoucími rozměry takového projektu automaticky rostou i požadavky na ověření správnosti jeho návrhu. Důsledky plynoucí z nedostatečného pochopení jeho podstaty a všech eventualit, které mohou po jeho realizaci nastat, nezřídka dosahují nedozírných finančních částek a mnohdy představují i ohrožení bezpečnosti osob. Rozmach výpočetní techniky, jejíž výkon se neustále zvyšuje a pořizovací náklady klesají, dal mimo jiné za vznik aplikacím, zaměřeným na simulace takových projektů ještě před jejich vlastním vznikem.

Cílem mé práce bude návrh a implementace takového nástroje v podobě knihovny pro objektově orientovaný programovací jazyk *C++*. Tento nástroj bude určen pro simulace transportních sítí pomocí formalizmu diskrétních a spojitých Petriho sítí. Uživateli tedy musí poskytnout oporu alespoň části, tohoto velmi širokého formalizmu, zaměřené na výpočet značení Petriho sítí. Dalším požadavkem je možnost snadného spojování obecného počtu Petriho sítí do větších celků tak, aby bylo možné nejprve vypracovat modely základních prvků dané transportní sítě a tyto prvky následně spojit do požadovaného celkového modelu simulované oblasti.

Dalším bodem práce bude návrh a vypracování knihovny základních modelů pro problematiku městské dopravy. Kromě prvků, jakými jsou křižovatky či ulice definujeme i modely vstupu a výstupu vozidel ze simulované oblasti. Zaměříme se zejména na modely založené na spojitých Petriho sítích, které se z mnoha důvodů jeví jako zajímavý způsob aproximace této, jinak diskrétní, transportní sítě.

Posledním bodem této práce bude provedení příkladové simulace na základě navržených modelů a implementovaného simulátoru. Simulace bude založena na reálných datech z regionu Praha – Smíchov. Oblast zahrnuje deset různých, navazujících křižovatek a svým rozsahem tak poskytne dostatečný prostor pro posouzení správnosti návrhu modelů i implementace samotného simulátoru.

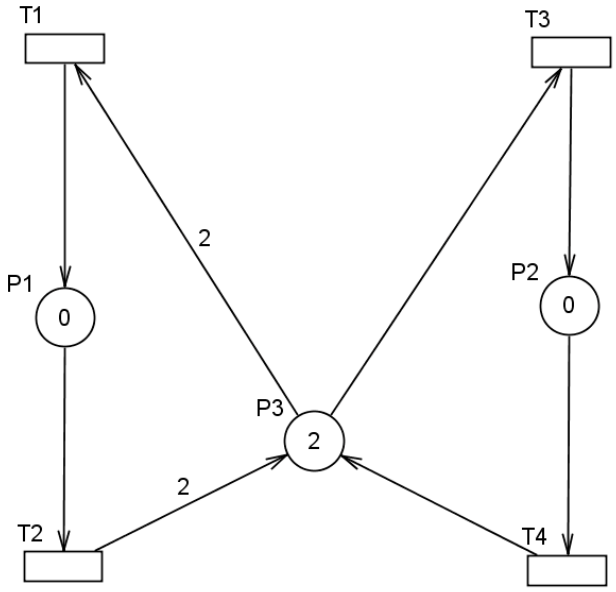
2. Petriho sítě

Formalizmus Petriho sítí je nástroj původně určený pro modelování a analýzu systémů diskrétních událostí. Základy tohoto oboru byly položeny v roce 1962 disertační prací [Petri] německého matematika *C. A. Petri*. Od té doby se v důsledku specifických požadavků různých oborů a snahy o další zjednodušování popisu systémů diskrétních událostí rozrostl o velké množství typů a speciálních vlastností Petriho sítí. Od obecných, časovaných (deterministické, stochastické), prioritních a spojitých Petriho sítí až po hybridní, zkrácené (barvené, kapacitní) a rozšířené (s inhibitovanou hranou) Petriho sítě.

Oproti stavovým automatům, které se v této oblasti typicky používají, umožňují Petriho sítě výrazně zjednodušit popis a analýzu zejména distribuovaných systémů. Pro popis distribuovaného systému pomocí stavového automatu je nutné, aby pro každou kombinaci stavů jeho podsystémů byl zaveden zvláštní stav. V Petriho sítích je však stav distribuovaného systému dán aktuálním značením sítě. V této práci se budeme zajímat především o časované Petriho sítě a spojité Petriho sítě s konstantní maximální rychlostí (*CCPN − Constant speed Continuous Petri Nets*).

2.1. Obecná Petiho síť

Graficky lze Petriho síť znázornit jako orientovaný ohodnocený bipartitní graf. Skládá se tedy ze dvou typů uzlů (míst a přechodů), navzájem propojených orientovanými hranami, které jsou ohodnoceny vahou. Hrana přitom spojuje vždy dva uzly odlišného typu. Grafický popis Petriho sítí je velmi silným nástrojem, umožňujícím jednoznačnou reprezentaci zkoumaného systému. S výhodou lze modelovat distribuované systémy po jednotlivých podsystémech a ty poté spojovat do větších celků. Na obrázku **obr. 2-1** je uveden příklad obecné Petriho sítě.



obr. 2-1: obecná Petriho síť

Petriho sítě lze dle vztahu s okolním prostředím dělit na autonomní a neautonomní. O autonomní Petriho síti hovoříme tehdy, pokud systém, jenž představuje, pracuje autonomně, tedy nezávisle na svém okolí a to včetně času. Naproti tomu neautonomní Petriho síť popisuje systém, jehož funkce je závislá na vnějších událostech, ať už se jedná o vnější vstupy do systému (synchronizovaná Petriho síť) nebo čas (časovaná / časová Petriho síť).

2.1.1. Neoznačená Petriho síť

Neoznačená Petriho síť je v [Hanzálek I] definována jako uspořádaná čtveřice (*P, T, Pre, Post*), kde *P* je konečná množina míst (*place*), neboli $P = \{P_1, ..., P_i, ..., P_m\}$, *T* je konečná množina přechodů (*transition*), neboli $T = \{T_1, ..., T_j, ..., T_n\}$, *Pre* je matice zobrazení $P \times T \rightarrow Z_0^+$ reprezentující spojení přechodu s předchozím místem (*precondition*) a *Post* je matice zobrazení $P \times T \rightarrow Z_0^+$ reprezentující spojení přechodu s následujícím místem (*postcondition*). Matice *Pre* a *Post*, reprezentující váhy hran, lze souhrnně zapsat pomocí incidenční matice $W = Post - \mathrm{Pr}\,e$, která však v některých případech není schopna jednoznačně popsat strukturu sítě. Pro Petriho síť uvedenou na **obr. 2-1** lze tedy sestrojit následující matice:

$$\mathrm{Pr}\,e = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix} \quad Post = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix} \quad W = Post - \mathrm{Pr}\,e = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -2 & 2 & -1 & 1 \end{bmatrix}. \qquad (2.1)$$

Pro zjednodušení zápisu dále uveďme následující značení. Množina $^\circ T_j$ obsahuje všechna místa vstupující do přechodu T_j a množina T°_j obsahuje všechna místa vystupující z tohoto přechodu. Dále pak množina $^\circ P_i$ obsahuje všechny přechody vstupující do místa P_i a množina P°_i obsahuje všechny přechody vystupující z místa P_i.

2.1.2. Označená Petriho síť

Označená Petriho síť je v [Hanzálek I] definována jako uspořádaná pětice (*P, T, Pre, Post, m_0*), kde m_0 je vektor zobrazení $P \rightarrow Z_0^+$ reprezentující počáteční značení (*initial marking*). Prvek $m_0(P_i)$ reprezentuje počet tokenů (značek, pešků) obsažených v místě P_i na počátku vývoje Petriho sítě. Vektory *m, m', m_1, m_2, ...* reprezentují možná značení Petriho sítě v průběhu vývoje systému. Pro Petriho síť uvedenou na **obr. 2-1** je vektor $m_0 = [0\ 0\ 2]'$.

2.1.3. Vývoj značení Petriho sítě

Je třeba připomenout, že obecné Petriho sítě, tak jak jsme je definovaly, jsou časově invariantní. Vývoj značení takové sítě je tedy zcela autonomní proces a k pohybu tokenů

(přeskoku) přes přechody v síti dochází pouze za splnění podmínky, že je přechod tzv. uvolněn.

Dle [Hanzálek I] je přechod T_j uvolněn (*enabled*) právě když pro všechna místa P_i vstupující do přechodu T_j platí, že počet tokenů je větší nebo roven váze hrany z P_i do T_j. Pokud je přechod uvolněn, potom může být přeskočen. Přeskok (*firing*, zapálení) přechodu T_j odejme tokeny z míst vstupujících do přechodu a vloží tokeny do míst vystupujících z přechodu. Dodejme jen, že přeskok je elementární, dále nedělitelná akce.

Matematicky lze vyjádřit přechod sítě do dalšího stavu, vlivem přeskoku některého z přechodů, jako $\forall P_i \in P$; $m'(P_i) = m(P_i) + Post(P_i, T_j) - \Pr e(P_i, T_j)$, kde $m(P_i)$ je počet tokenů v místě P_i před přeskokem a $m'(P_i)$ po přeskoku přechodu T_j. Pro přeskok přechodu T_1 Petriho sítě uvedené na **obr. 2-1** lze tedy určit nové značení sítě jako:

$$m' = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \tag{2.2}$$

2.1.4. Konflikt

Konflikt v Petriho síti vyjadřuje nedeterministické chování systému a většině aplikací se mu tedy snažíme předejít nebo nalézt cestu k jeho deterministickému řešení. Rozlišujeme následující dva typy konfliktu. Dle [Hanzálek I] má Petriho síť strukturální konflikt v místě P_i právě když existují alespoň dva přechody vystupující z tohoto místa. Efektivní konflikt v místě P_i má Petriho síť, pokud má v tomto místě strukturální konflikt a značení $m(P_i)$ nepostačuje pro přeskok všech uvolněných přechodů vystupujících z tohoto místa. Petriho síť na obrázku **obr. 2.1** má tedy efektivní konflikt v místě P_3.

Nejjednodušším opatřením je použití prioritních Petriho sítí a diferenciace priorit přechodů, jenž v konfliktu vystupují. Uvolněné přechody jsou pak přeskočeny v posloupnosti nerostoucích priorit a po přeskoku každého přechodu je znovu vyhodnocena uvolněnost zbývajících přechodů.

2.1.5. Vlastnosti Petriho sítí

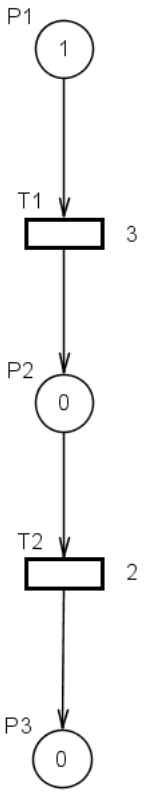
Velmi zajímavým odvětvím tohoto oboru je analýza vlastností Petriho sítí. Vlastnosti reálného systému, modelovaného pomocí Petriho sítě, tak lze velmi přesně zkoumat (verifikace systému) ještě před jeho vlastní realizací. Analýzu některých vlastností Petriho sítě lze, díky matematickému aparátu o který se opírá, provést bez nutnosti vyčíslení grafu všech dosažitelných značení. Z vlastností závislých na značení sítě jmenujme *ohraničenost*, *živost*, *reverzibilitu* a *deadlock*. Naproti tomu mezi vlastnosti nezávislé na značení sítě patří tzv. $P-$ *invarianty* a $T-$ *invarianty*.

V této práci se analýze vlastností Petriho sítí nebudeme podrobně věnovat, pro simulace transportních sítí nejsou podstatné. Definujme pouze *deadlock* (uváznutí), který je důležitou vlastností pro ukončení vývoje značení sítě. Petriho síť má *deadlock* pokud žádný z přechodů sítě nemůže být přeskočen. Síť uvedená na **obr. 2-1** nemá *deadlock*, jelikož neexistuje žádná posloupnost přeskoků přechodů, jenž by vedla k tomu, že žádný z přechodů nebude uvolněn. Detailní rozbor ostatních vlastností Petriho sítí lze nalézt v [Hanzálek I].

2.2. Časovaná Petriho síť

V Petriho sítích je možné definovat čas více způsoby. Každému přechodu v síti je možné například přiřadit časový interval a k přeskoku přechodu po jeho uvolnění může dojít kdykoliv v tomto časovém intervalu. Taková síť se nazývá časová Petriho síť. Druhou možností je přiřazení konkrétní hodnoty času, po který musí být daný přechod uvolněn před vlastním přeskokem. Pak se jedná o časovanou Petriho síť.

Časované Petriho sítě je možné definovat jako *T − časované* (čas přiřazen přechodům) nebo *P − časované* (čas přiřazen místům). V této práci se budeme věnovat pouze *T − časovaným* Petriho sítím, které mohou být jednoduchým způsobem převedeny na *P − časované*, viz. [Hanzálek II].



obr. 2-2: časovaná Petriho síť

Časovaná Petriho síť je tedy uspořádaná šestice (P, T, Pre, $Post$, m_0, t), kde t je vektor zobrazení $T \rightarrow R_0^+$ reprezentující čas po kterém může být uvolněný přechod přeskočen. Grafické znázornění časované Petriho sítě je obdobné jako u obecné Petriho sítě. U každého přechodu je navíc číselně uveden čas $t(T_j)$.

2.2.1. Vývoj značení časované Petriho sítě

Vývoj značení časované Petriho sítě vychází z obecné Petriho sítě s tím rozdílem, že k přeskoku uvolněného přechodu T_j může dojít nejdříve po uplynutí definovaného času $t(T_j)$ od jeho uvolnění. Uvažujme příklad Petriho sítě z **obr. 2-2**. Vývoj značení této sítě je uveden v tabulce **2-1**. Vzhledem k faktu, že v čase $t = 5$ již není uvolněn žádný z přechodů, končí vývoj této sítě stavem *deadlock*.

iterace	čas	*m(P1)*	*m(P2)*	*m(P3)*
0	0	1	0	0
1	3	0	1	0
2	5	0	0	1

tabulka 2-1: vývoj značení časované Petriho sítě

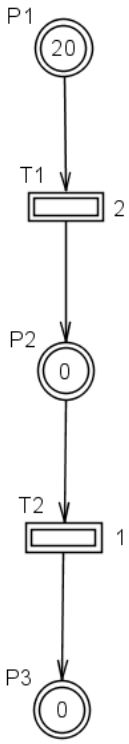
2.2.2. Konflikt v časované Petriho síti

Časovaná Petriho síť má efektivní konflikt v místě P_i, pokud má v tomto místě strukturální konflikt a značení $m(P_i)$ nepostačuje pro přeskok všech uvolněných přechodů z množiny $P°_i$, které jsou připraveny k přeskoku. Přechod T_j je připraven k přeskoku, pokud byl uvolněn minimálně po dobu $t(T_j)$. K deterministickému řešení konfliktu v časované Petriho síti lze, stejně jako v obecné Petriho síti, použít diferenciaci priorit přechodů, vystupujících v konfliktu.

2.3. Spojitá Petriho síť

Spojité Petriho sítě představují poněkud odlišný přístup k modelování systémů. Značení Petriho sítě je zde uvažováno jako v čase proměnná spojitá veličina a každému přechodu je přiřazena maximální rychlost pohybu značení přes tento přechod z oboru nezáporných reálných čísel. Pomocí spojitých Petriho sítí lze modelovat spojité systémy a aproximovat systémy diskrétních událostí. Vývoj diskrétních Petriho sítí je tak možné realizovat ve velmi zhuštěné podobě. Grafická reprezentace spojité Petriho sítě, kterou budeme v této práci používat, je patrná z obrázku **obr. 2-3**.

Spojitá Petriho síť je uspořádaná šestice (P, T, Pre, $Post$, m_0, V), kde V je vektor zobrazení $T \rightarrow R_0^+$ reprezentující maximální rychlosti přechodů. K označení skutečné rychlosti přechodu T_j v daném čase budeme používat označení $v_j(t)$. K pochopení vývoje značení spojité Petriho sítě je nejprve třeba definovat některé základní termíny dle [Hanzálek III].



obr. 2-3: spojitá Petriho síť

Značené místo (*marked place*)

Místo P_i je v čase t značené, pokud $m(P_i, t) > 0$.

Silně uvolněný přechod (*strongly enabled transition*)

Přechod T_j je silně uvolněn v čase t, pokud všechna místa P_i z $°T_j$ jsou značená.

Zásobené místo (*supplied place*)

Místo P_i je v čase t zásobené, pokud existuje alespoň jeden přechod T_j v množině $°P_i$, který je uvolněný (silně nebo slabě).

Zásobovací místo (*supplying place*)

Místo P_i je zásobovací, pokud existuje přechod T_j v množině $P°_i$, který je uvolněn (silně nebo slabě).

Slabě uvolněný přechod (*weakly enabled transition*)

Přechod T_j je v čase t slabě uvolněn, pokud existuje P_i z množiny $°T_j$, které není značené, je zásobené a zbývající místa z množiny $°T_j$ jsou značená nebo zásobená.

Uvolněný přechod (*enabled transition*)

Přechod T_j je uvolněn pokud je slabě nebo silně uvolněn.

Rovnováha místa (*balance*)

Dle [Hanzálek III] odpovídá rovnováha místa P_i v čase t derivaci značení tohoto místa. Udává tedy přírůstek (kladná hodnota) nebo úbytek (záporná hodnota) počtu tokenů v daném místě za jednotku času. Matematicky lze rovnováhu místa P_i v čase t vyjádřit rovnicí:

$$B_i(t) = \sum_{T_j \in {}^\circ P_i} Post(P_i, T_j) \cdot v_j(t) - \sum_{T_k \in P^\circ{}_i} \Pr e(P_i, T_k) \cdot v_k(t) . \qquad (2.3)$$

Je patrné, že definice slabě uvolněného přechodu a zásobeného místa jsou navzájem rekurzivní. Z tohoto důvodu není možné tyto stavy určit přímým postupem. Iterativní algoritmus s rychlou konvergencí k určení množin silně a slabě uvolněných přechodů pro dané značení sítě je uveden v [Alla].

Dále budeme uvažovat spojité sítě s maximální možnou rychlostí přechodů (*maximum speed CCPN*). To znamená, že silně uvolněný přechod má vždy skutečnou rychlost rovnu své maximální rychlosti a slabě uvolněný přechod má nejvyšší možnou rychlost, menší než maximální rychlost V_j a větší než nula.

2.3.1. Vývoj značení spojité Petriho sítě

Výpočet vývoje značení spojité Petriho sítě probíhá, i přes svůj spojitý charakter, v diskrétních iteracích rozdělených na tzv. IB-stavy[1]. Spojitá Petriho síť přechází do dalšího IB-stavu, pokud dojde ke změně skutečné rychlosti libovolného přechodu nebo více přechodů v síti. Mezi jednotlivými IB-stavy jsou skutečné rychlosti všech přechodů konstantní a lze tedy vypočítat čas, ve kterém vlivem změny značení sítě a tedy i rychlostí přechodů dojde k přechodu do dalšího IB-stavu. Právě tato skutečnost vede ke zhuštění vývoje značení aproximované diskrétní sítě.

V každé iteraci je nejprve třeba určit, pomocí výše uvedeného iterativního algoritmu, množiny značených míst, zásobených míst, silně a slabě uvolněných přechodů. Vycházíme tedy z aktuálního značení sítě v čase t. Na základě stavů jednotlivých uzlů sítě můžeme dále určit skutečné rychlosti přechodů. Skutečná rychlost každého uvolněného přechodu musí být menší než jeho maximální rychlost a větší než nula. U spojitých Petriho sítí s maximální rychlostí navíc dle [Alla] platí, že každý silně uvolněný přechod má skutečnou rychlost rovnu jeho maximální rychlosti a skutečná rychlost každého slabě uvolněného přechodu musí nabývat maximální možné rychlosti dle vztahu:

$$v_j(t) = \min[V_j, \min_i(B_i(t) + v_j(t))] \qquad (2.4)$$

pro taková i, že $P_i \in {}^\circ T_j$ a $m_i(t) = 0$.

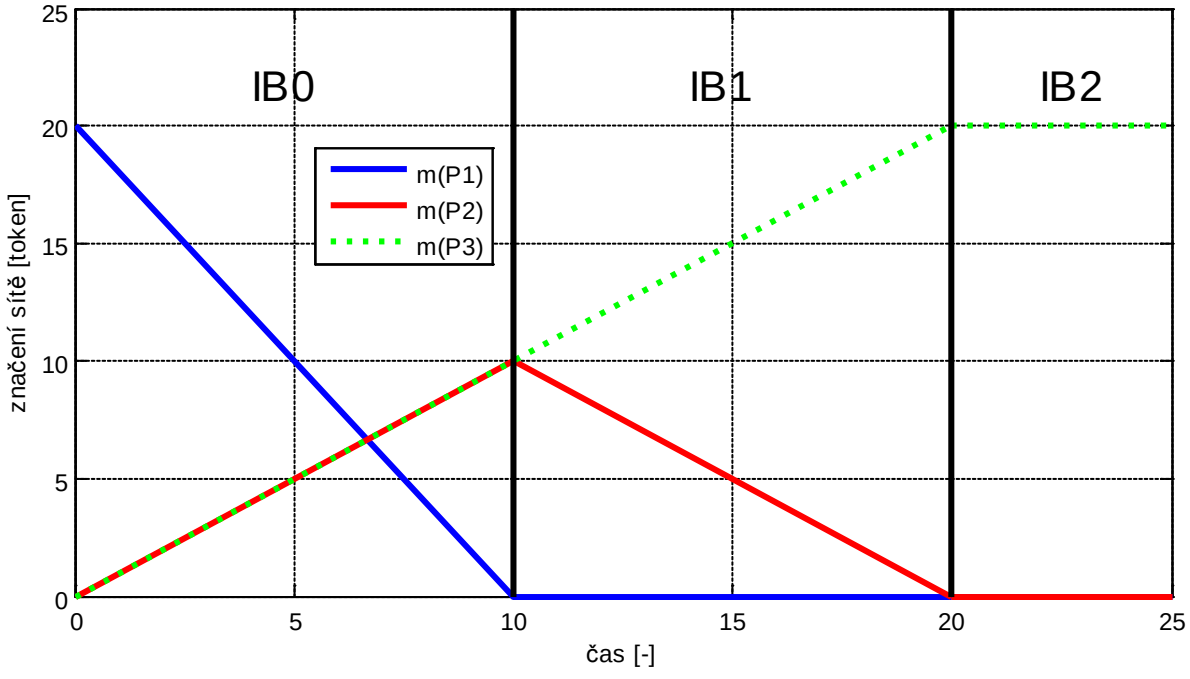
[1] z anglického *Invariant Behavior state (IB-state)*

Z výše uvedeného vyplývá, že pro výpočet skutečných rychlostí přechodů je možné sestavit soustavu lineárních nerovnic, které omezují prostor možných řešení. Dále lze tedy tuto úlohu řešit metodou lineárního programování s kritériem maximalizace sumy skutečných rychlostí přechodů v dané Petriho síti.

Petriho síť na obrázku **obr. 2-3** má v čase $t = 0$ jedno značené místo P_1. Přechod T_1 je tedy silně uvolněn, místo P_2 je zásobené. Přechod T_2 je vzhledem k nulovému značení tohoto místa slabě uvolněn a místo P_3 je rovněž zásobené. Skutečná rychlost přechodu T_1 je rovna jeho maximální rychlosti, jelikož je silně uvolněn. Tedy $v_1 = V_1 = 2$. Skutečná rychlost přechodu T_2 musí být větší než nula a menší než jeho maximální rychlost. Vzhledem k faktu, že hovoříme o spojité Petriho sítí s maximální rychlostí, musí navíc nabývat maximální možné hodnoty. Maximální možná rychlost tohoto přechodu je dle (2.4) omezena $v_2(0) = \min[1,2]$. Skutečná rychlost přechodu T_2 je tedy $v_2 = 1$. Skutečné rychlosti přechodů lze určit metodou lineárního programování. Problém lineárního programování pro Petriho síť na obrázku **obr. 2-3** v čase $t = 0$ lze definovat jako:

$$\begin{aligned} v_1(t) &= 2 \\ v_2(t) &\geq 0 \\ v_2(t) &\leq 1 \\ v_2(t) &\leq v_1(t) \end{aligned} \quad , (2.5) \qquad \max(v_1(t) + v_2(t)). \quad (2.6)$$

Grafické znázornění vývoje značení této sítě je uvedeno na **obr. 2-4**. Je patrné, že celý výpočet byl zpracován pouze ve dvou iteracích a třech IB-stavech i přes poměrně vysoké počáteční značení místa P_1. Vývoj sítě je zakončen v čase $t = 20$ stavem *deadlock*.



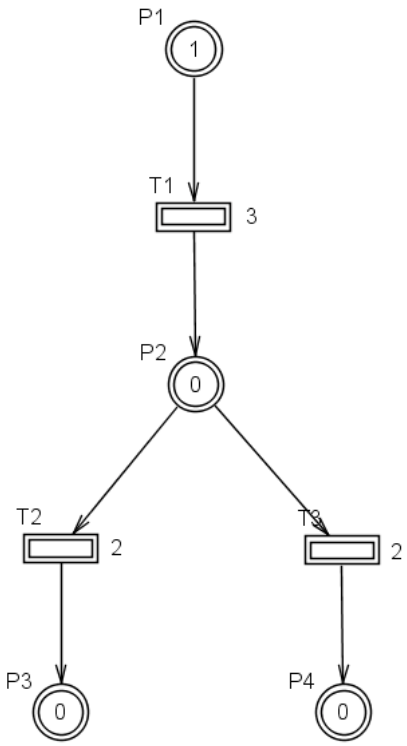
obr. 2-4: vývoj značení spojité Petriho sítě

2.3.2. Konflikt ve spojité Petriho síti

Strukturální konflikt je ve spojité Petriho síti definován stejně jako v obecné Petriho síti. Spojitá Petriho síť má v místě P_i efektivní konflikt, pokud má v tomto místě strukturální konflikt, místo je neoznačené, zásobené a platí, že:

$$\sum_i v_i(t) < \sum_j V_j \,, \tag{2.7}$$

pro všechna *i* a *j* taková, že $T_i \in {}^\circ P_i$ a $T_j \in P^\circ{}_j$. V takovém případě je tedy místo P_i zásobené rychlostí, jenž není schopna zcela pokrýt maximální rychlosti všech přechodů z množiny $P^\circ{}_i$ a nelze rozhodnout, jakým způsobem se tato celková maximální rychlost rozdělí mezi přechody v množině $P^\circ{}_i$.



obr. 2-5: spojitá Petriho síť s konfliktem

Jedním ze způsobů jak tuto situaci řešit deterministicky, stejně jako v předchozích případech, je použití prioritní spojité Petriho sítě a diferenciace priorit přechodů v každé množině přechodů s konfliktem. Výpočet skutečných rychlostí pak probíhá postupně, pro každou úroveň priorit a výsledné rychlosti přechodů vyšší priority se přidají do podmínek problému lineárního programování jako konstanty. Takový přístup však není vždy vhodný, jelikož striktně upřednostňuje přechody s vyšší prioritou.

Jiným způsobem je proporcionální řešení efektivního konfliktu, který je blíže popsán v [Kutil]. Rychlost, jíž je konfliktní místo P_i zásobeno, je rozdělena mezi přechody z množiny $P°_i$ v poměru, který je nejblíže poměru jejich maximálních rychlostí. Tento postup spočívá v přidání několika podmínek do výše uvedené soustavy lineárních nerovnic a úpravou kritéria lineárního programování. Pro Petriho síť uvedenou na **obr. 2-5** lze stanovit v čase $t = 0$ omezující podmínky a kritérium optimalizace:

$$\begin{aligned} &v_1(t) = 3 \\ &v_2(t) \geq 0 \\ &v_2(t) \leq 2 \\ &v_3(t) \geq 0 \\ &v_3(t) \leq 2 \\ &v_2(t) + v_3(t) \leq v_1(t) \end{aligned} \quad , (2.8) \qquad \max\left(v_2(t) + v_3(t) - \varepsilon \cdot \left| v_3(t) - v_2(t) \cdot \frac{V_3}{V_2} \right| \right), \quad (2.9)$$

$$\text{kde } \varepsilon \in \left(0, \min\left(1, \frac{V_2}{V_3} \right) \right) \qquad (2.10)$$

je dostatečně malá konstanta, která zaručuje, že maximalizace sumy rychlostí v kritériu má přednost před minimalizací vzdálenosti řešení od poměru maximálních rychlostí přechodů. Kritérium optimalizace (2.9) lze s použitím přidané proměnné z_2 přepsat do tvaru:

$$\max\left(v_2(t) + v_3(t) - \varepsilon \cdot z_2 \right), \qquad (2.11)$$

za předpokladu, že do omezení problému lineárního programování přidáme dvě další podmínky:

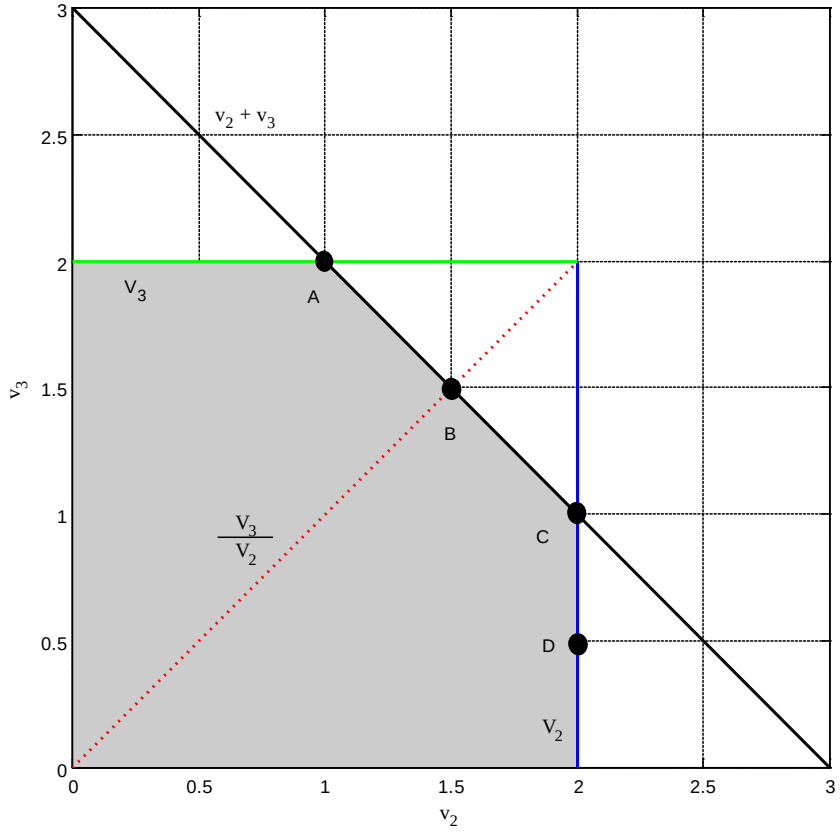
$$\begin{aligned} v_3(t) - v_2(t) \cdot \frac{V_3}{V_2} &\leq z_2 \\ v_3(t) - v_2(t) \cdot \frac{V_3}{V_2} &\geq -z_2 . \end{aligned} \qquad (2.12)$$

Tento postup lze zobecnit pro případ, že konflikt obsahuje obecný počet přechodů. Dostáváme tak následující obecné omezující podmínky, které budou přidány ke stávajícím a celkové kritérium optimalizace problému lineárního programování:

$$\begin{aligned} v_j(t) - v_i(t) \cdot \frac{V_j}{V_i} &\leq z_j \\ v_j(t) - v_i(t) \cdot \frac{V_j}{V_i} &\geq -z_j \end{aligned} \quad , (2.13) \qquad \max\left(\sum_{j \in P°_i} v_j(t) - \varepsilon \sum_{j \in P°_i \setminus i} z_j \right), \qquad (2.14)$$

kde v_i je aktuální rychlost libovolného přechodu $T_i \in P°_i$.

Řešení efektivního konfliktu ve spojité Petriho síti z obrázku **obr. 2-5** je graficky znázorněno na **obr. 2-6**. Skutečné rychlosti přehodů T_2 a T_3 jsou zdola omezeny osami grafu (nulovou hodnotou), zhora pak svými maximálními rychlostmi (zelená a modrá přímka). Součet skutečných rychlostí přechodů T_2 a T_3 musí být rovněž menší, než skutečná rychlost přechodu T_1 (černá úsečka). Pro obecné spojité Petriho sítě tak šedivě zabarvená oblast představuje prostor možných řešení při hledání skutečných rychlostí v_2 a v_3. Protože však uvažujeme spojité Petriho sítě s maximální rychlostí, je prostor všech možných řešení reprezentován pouze úsečkou *AC*. Červená přímka reprezentuje poměr mezi maximálními rychlostmi přechodů T_2 a T_3.

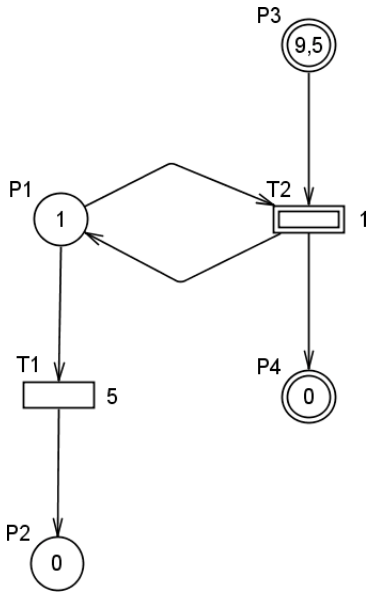


obr. 2-6: grafická reprezentace konfliktu a jeho řešení

Všechny body náležící úsečce *AC* splňují požadavky na nedeterministické řešení konfliktu. Pokud bychom použili metodu diferenciace priorit přechodů a např. přechodu T_2 přiřadili vyšší prioritu než přechodu T_3, řešením by byl bod *C*, v opačném případě pak bod *A*. Proporcionální metoda však hledá takové řešení na úsečce *AC*, které má nejmenší vzdálenost od poměru maximálních rychlostí přechodů T_2 a T_3 (červená přímka). V tomto konkrétním příkladu je tedy řešením bod *B*. Pokud bychom uvažovali maximální rychlost $V_3 = 0.5$, Petriho síť by neměla efektivní konflikt a řešením by byl bod *D*. Další metody řešení efektivního konfliktu ve spojitých Petriho sítích jsou popsány v [David I].

2.4. Hybridní Petriho síť

Hybridní Petriho síť je spojením diskrétní a spojité Petriho sítě. Dle [David II] lze hybridní Petriho síť definovat jako uspořádanou šestici (P, T, Pre, $Post$, m_0, h), kde h je tzv. hybridní funkce $P \cap T \rightarrow \{D, C\}$. Ta každému uzlu sítě přiřazuje hodnotu z množiny $\{D, C\}$ dle toho, zda se jedná o diskrétní uzel (množinu diskrétních míst značíme P^D, množinu diskrétních přechodů pak T^D) nebo spojitý uzel (množinu spojitých míst značíme P^C, množinu spojitých přechodů pak T^C). Příklad hybridní Petriho sítě je uveden na **obr. 2-7**.



obr. 2-7: hybridní Petriho síť

Prvky matic Pre, $Post$ a vektoru m_0 zde mohou nabývat jak hodnot z oboru přirozených čísel N (pro takové místo P_i, že $P_i \in P^D$) tak z oboru nezáporných reálných čísel R_0^+ (pro takové místo P_i, že $P_i \in P^C$). Pokud pro místo P_i a přechod T_j platí, že $P_i \in P^D$ a $T_i \in T^C$, pak musí být navíc splněna podmínka:

$$\Pr e(P_i, T_j) = Post(P_i, T_j). \tag{2.15}$$

Tato podmínka tedy říká, že pokud existuje hrana mezi diskrétním místem P_i a spojitým přechodem T_j, musí mezi těmito uzly existovat rovněž hrana opačného směru a stejné váhy. Tak je zaručeno, že značení diskrétního místa vždy nabývá hodnot z oboru přirozených čísel. Tato situace je na **obr. 2-7** reprezentována spojením diskrétního místa P_1 a spojitého přechodu T_2. Poznamenejme, že značení místa P_1 se podílí na uvolnění spojitého přechodu T_2 a diskrétního přechodu T_1. V uvedeném případě je diskrétní přechod T_1 uvolněn od času $t = 0$ do $t = 5$ bez přerušení. Bližší informace o hybridních Petriho sítích lze nalézt v [David II].

3. Jazyk *PNML*

PNML (*Petri Net Markup Language*) je návrh jazyka založeného na formátu *XML* (*eXtensible Markup Language*) pro ukládání, zpětné načítání a přenos dat v podobě souborů popisujících Petriho sítě. Snahu o popis podobného formátu v minulosti projevilo více skupin zabývajících se problematikou Petriho sítí. Jazyk *PNML* se však stal jedním z mála, který v tomto směru pokračuje ve snaze o vytvoření jistého standardu.

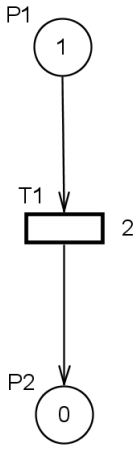
Je však nutné poznamenat, že i vývoj tohoto standardu se v poslední době zpomalil a definuje tak především základní prvky Petriho sítí. Další specifické vlastnosti je nutné definovat dodatečně, podle potřeb konkrétní aplikace. Pro potřeby této práce byl jazyk *PNML* doplněn o definice priorit přechodů a rozšířen o spojité Petriho sítě. Takto upravený *PNML* formát byl zpracován v podobě podpůrných definic pro obecný editor grafů *VSPNE*, který byl v rámci této práce použit jako primární nástroj pro návrh Petriho sítí. Bližší informace o tomto editoru lze nalézt v [VSPNE]. Podpůrné definice jsou pak k dispozici v příloze ***A***.

V této kapitole popíšeme jazyk *PNML* jednoduchou a přehlednou cestou názorných příkladů. Vyhneme se tak složitějšímu přístupu, vycházejícímu z přesných definic *XML* dokumentu pomocí jazyka *RELAX NG* [RELAX NG]. Podrobné informace o jazyku *PNML* jsou uvedeny v [PNML].

Každý *PNML* soubor začíná hlavičkou standardního *XML* souboru s uvedenou verzí a použitým kódováním. V kterémkoliv místě souboru je rovněž možné uvádět standardní *XML* komentáře uvozené speciálními posloupnostmi znaků <*!*-- a -->. Definice samotné Petriho sítě je pak zapouzdřena do dvou elementů. Prvním je element <*pnml*> s nepovinným atributem *xmlns*, odkazujícím na definice použitého *PNML* formátu. Druhým pak element <*net*> s povinným atributem *id*, tedy jedinečným identifikátorem dané sítě a nepovinným atributem *type*, odkazujícím na definice daného typu Petriho sítě.

3.1. Časovaná Petriho síť v jazyku *PNML*

Místo časované Petriho sítě je uvozeno elementem <*place*> a počáteční značení tohoto místa pak elementy <*initialMarking*> a <*text*>. Hodnota počátečního značení místa může nabývat pouze nezáporných celočíselných hodnot. Souřadnice *X* a *Y*, na kterých se místo nachází např. v grafickém editoru, jsou uvedeny jako atributy *x* a *y* elementu <*position*>. Ten je navíc zapouzdřen do elementu <*graphics*>. Přechod časované Petriho sítě je definován obdobně jako místo. Je uvozen elementem <*transition*> a grafické vlastnosti jsou totožné s grafickými vlastnostmi místa.



obr. 3-1: časovaná Petriho síť

V nepovinných elementech <*priority*><*text*> a <*time*><*text*> je pak uvedena priorita a čas přiřazený přechodu. Priorita nabývá rovněž nezáporných celočíselných hodnot a jako implicitní hodnotu budeme dále uvažovat nulovou prioritu. Čas nabývá nezáporných reálných hodnot s implicitní hodnotou rovněž rovnou nule.

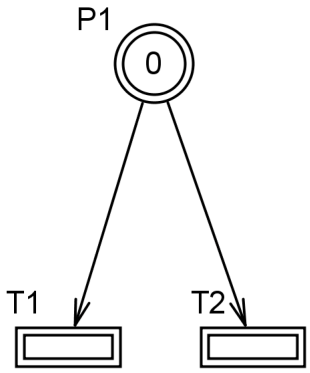
```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Document created by GPNS.
-->

<pnml xmlns="http://www2.stpnplay.com/vspne/schema/StochPNTD">
	<net id="SPN1" type="http://www2.stpnplay.com/vspne/schema/StochPNTD">

		<place id="P2">
			<graphics>
				<position x="653" y="317"/>
			</graphics>
			<initialMarking>
				<text>0</text>
			</initialMarking>
		</place>

		<place id="P1">
			<graphics>
				<position x="652" y="124"/>
			</graphics>
			<initialMarking>
				<text>1</text>
			</initialMarking>
		</place>

		<transition id="T1">
			<graphics>
				<position x="653" y="219"/>
			</graphics>
			<priority>
				<text>0</text>
			</priority>
			<time>
				<text>2</text>
			</time>
		</transition>

		<arc id="A1" source="P1" target="T1">
			<inscription>
				<text>1</text>
			</inscription>
		</arc>

		<arc id="A2" source="T1" target="P2">
			<inscription>
				<text>1</text>
			</inscription>
		</arc>

	</net>
</pnml>
```

PNML 3-1: časovaná Petriho síť

Hrana časované Petriho sítě je uvozena elementem <*arc*> se třemi povinnými atributy: *id* jako jedinečný identifikátor hrany, *source* jako jedinečný identifikátor uzlu sítě, ze kterého hrana vychází a *target* jako identifikátor uzlu do něhož vchází. Váha hrany je uvedena v elementech <*inscription*><*text*> a může nabývat hodnot z oboru nezáporných celých čísel. Pořadí definic míst a přechodů v souboru může být libovolné. Příklad struktury *PNML* souboru pro Petriho síť z **obr. 3-1** je uveden v **PNML 3-1**.

3.2. Spojitá Petriho síť v jazyku *PNML*

Spojitá Petriho síť není dosud ve standardu *PNML* definována. Proto bylo nutné některé vlastnosti míst a přechodů definovat dodatečně tak, aby svými vlastnostmi odpovídali místům a přechodům spojité Petriho sítě. Jako výchozí formát byl uvažován popis časované Petriho sítě, ve kterém došlo jen k několika nutným změnám.



obr. 3-2: spojitá Petriho síť

Místo a hrana spojité Petriho sítě jsou definovány shodně jako v místo a hrana časované Petriho sítě. Pouze počáteční značení místa a váha hrany může nabývat hodnot z oboru nezáporných reálných čísel.

Rychlost přechodu z oboru nezáporných reálných čísel je definována v elementech <*speed*><*text*>. Element <*share*> s atributem *id* je určen pro označení příslušnosti přechodu do skupiny přechodů, mezi kterými má být řešen efektivní konflikt proporcionální metodou. Atribut *id* je libovolný řetězec znaků označující tuto skupinu. Hodnota uzavřená v elementu <*text*> nabývá hodnot z oboru nezáporných reálných čísel a vyjadřuje podíl přechodu na řešení efektivního konfliktu. Pokud tato hodnota není uvedena, předpokládá se, že je tento podíl roven maximální rychlosti přechodu v souladu s kapitolou **2.3.2**.

Definice všech uzlů spojité Petriho sítě je navíc doplněna o atribut *type* nabývající hodnoty *continuous*. Tento nepovinný atribut slouží k rychlejšímu rozpoznání typu daného uzlu sítě při načítání dat ze souboru a zpřehledňuje do jisté míry i samotný zápis. Příklad *PNML* souboru pro spojitou Petriho síť z **obr. 3-2** je uveden v **PNML 3.2**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Document created by GPNS.
-->

<pnml xmlns="http://www2.stpnplay.com/vspne/schema/StochPNTD">
      <net id="CCPN1" type="http://www2.stpnplay.com/vspne/schema/StochPNTD">
            <place id="P1" type="continuous">
                  <graphics>
                        <position x="616" y="141"/>
                  </graphics>
                  <initialMarking>
                        <text>0</text>
                  </initialMarking>
            </place>

            <transition id="T2" type="continuous">
                  <graphics>
                        <position x="654" y="250"/>
                  </graphics>
                  <priority>
                        <text>0</text>
                  </priority>
                  <speed>
                        <text>0</text>
                  </speed>
                  <share id="">
                        <text>0</text>
                  </share>
            </transition>

            <transition id="T1" type="continuous">
                  <graphics>
                        <position x="583" y="250"/>
                  </graphics>
                  <priority>
                        <text>0</text>
                  </priority>
                  <speed>
                        <text>0</text>
                  </speed>
                  <share id="">
                        <text>0</text>
                  </share>
            </transition>

            <arc id="A2" source="P1" target="T2" type="continuous">
                  <inscription>
                        <text>1</text>
                  </inscription>
            </arc>

            <arc id="A1" source="P1" target="T1" type="continuous">
                  <inscription>
                        <text>1</text>
                  </inscription>
            </arc>
     </net>
</pnml>
```

PNML 3-2: spojitá Petriho síť

4. Implementace knihovny *GPNS*

Jedním z bodů této práce je implementace knihovny základních tříd a metod pro programovací jazyk *C++*, která by byla schopna zacházet s Petriho sítěmi, spojovat menší sítě do větších celků a provádět výpočty vývoje jejich značení v čase. Pro tyto účely byla implementována knihovna *GPNS* (*General Petri Nets Simulator*). Jako programovací jazyk byl zvolen jazyk *C++*, jelikož poskytuje propracovaný objektově orientovaný přístup, nízkou časovou náročnost základních operací a širokou základnu dodatečných knihoven, nástrojů pro vedení technické dokumentace atp.

4.1. Použité prostředky

Knihovna byla implementována ve vývojovém prostředí *Microsoft Visual Studio 2005*, které je pro studenty ČVUT Fakulty elektrotechnické poskytováno zdarma, pod záštitou projektu *MSDN AA* (*Microsoft Development Network Academic Aliance*). K realizaci projektu byly rovněž použity některé další nástroje a knihovny jazyka *C++*, které jsou stručně popsány v následujících podkapitolách.

4.1.1. *EXPAT*

Knihovna *EXPAT* je soubor zdrojových kódů v jazyku *C*, určený pro syntaktickou analýzu *XML* souborů (*XML parser*). Jedná se o jednoduchý jednoprůchodový analyzátor. Umožňuje uživateli definovat vlastní obslužné metody, které jsou volány v průběhu čtení *XML* souboru. V této práci je knihovna použita k načítání Petriho sítí z *PNML* souborů do paměti programu. Více informací o knihovně *EXPAT* je možné získat v [EXPAT].

Knihovna je velmi dobře použitelná i v jazyku *C++* a její vývoj stále pokračuje. Projekt je veden pod *MIT* licencí, která je velmi benevolentní z hlediska možných zásahů do zdrojových kódů a jejich další distribuce. Originální znění *MIT* licence je uvedeno v [MIT].

4.1.2. *uBLAS*

Knihovna *uBLAS* je soubor zdrojových kódů v jazyku *C++*, určený pro efektivní práci s hustými i řídkými vektory a maticemi. Vzhledem k použití přetížených operátorů a tříd definovaných v podobě šablon je její použití velice intuitivní. Projekt se do jisté míry snaží přiblížit knihovně *BLAS*[2] a poskytnout alespoň základní metody lineární algebry se srovnatelnou efektivitou, avšak v objektově orientované podobě. V této práci je knihovna použita zejména k reprezentaci a výpočtům s maticemi *Pre*, *Post* a vektory počátečního značení sítě m_0 pomocí řídkých matic a vektorů. Tím je dosaženo nízké paměťové náročnosti implementovaných algoritmů při zachování akceptovatelné časové náročnosti.

[2] *Basic Linear Algebra Subprograms* – Komplexní a velmi efektivní knihovna určená pro řešení problémů lineární algebry, implementována v jazyku *Fortran*, úspěšně použita např. pro některé výpočty v programu *MATLAB*. Srovnání efektivity knihoven *uBLAS* a *BLAS* je uvedeno v [Halmo].

V současnosti je *uBLAS* součástí komplexního souboru knihoven pro jazyk *C++* s názvem *Boost* se svou vlastní licencí. Více informací o souboru knihoven *Boost* nebo o samotné knihovně *uBLAS* je možno nalézt v [Boost].

4.1.3. *GLPK*

Knihovna *GLPK* (*GNU Linear Programming Kit*) je soubor zdrojových kódů implementovaných v jazyku *C*, určený pro řešení problémů lineárního programování. V této práci je použit pro výpočet skutečných rychlostí přechodů ve spojité Petriho síti dle kapitoly **2.3.1**. Knihovna je chráněna licencí *GPL* (*General Public License*). Více informací o nástroji *GLPK* je možné nalézt v [GLPK]. Přesné znění licence *GPL* je uvedeno v [GPL].

4.1.4. *Doxygen*

Technická dokumentace projektu je vedena pomocí nástroje *Doxygen*, doplněným o grafický balíček *Graphviz*, který slouží pro generování grafického znázornění struktury knihovny. Tento systém je založen na dokumentování kódu přímo ve zdrojových souborech pomocí speciálních značek. Následně je schopen vytvořit z těchto informací technickou dokumentaci ve formátu *HTML* nebo *LaTeX* a to včetně grafů, obrázků atp.

Tento přístup má mnoho výhod a výrazně šetří práci s dokumentací projektu. Neoddělitelnost dokumentace od vlastního projektu je navíc dobrým způsobem, jak ji udržovat stále aktuální. Systém spolupracuje s mnoha programovacími jazyky včetně *Java*, *C#*, *C++*, *Python* atd. Podrobný popis funkcí a možností systému *Doxygen* resp. *Graphviz* lze nalézt v [Doxygen] resp. [Graphviz]. Technická dokumentace knihovny *GPNS* je k dispozici v příloze ***A***.

4.2. Struktura knihovny *GPNS*

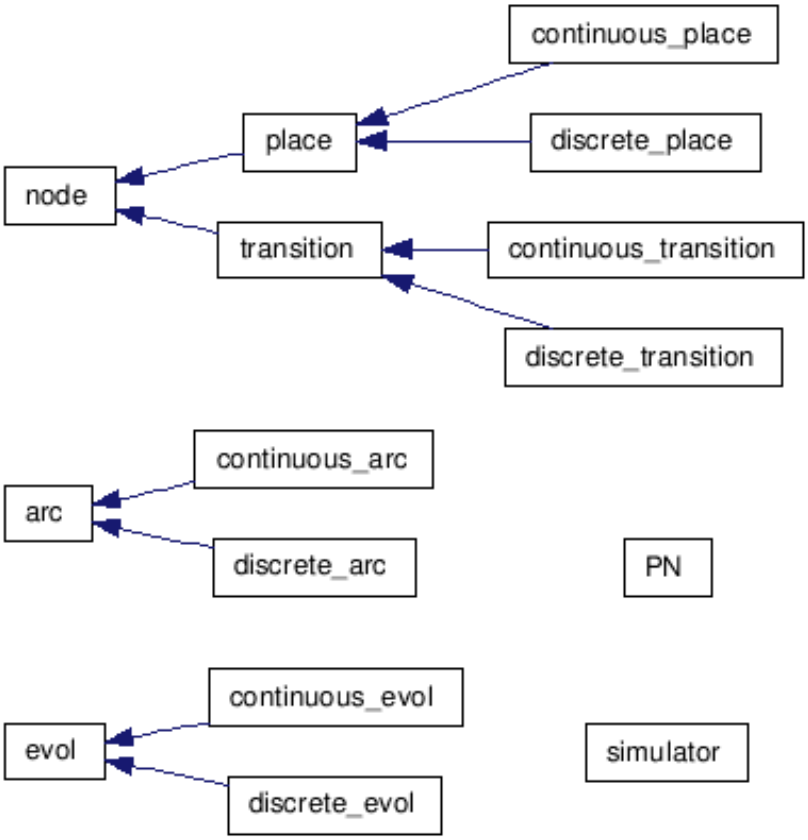
V průběhu řešení projektu byl kladen důraz na vysokou modularitu knihovny a možnost jejího budoucího rozšíření, například o podporu hybridních Petriho sítí nebo metod pro analýzu vlastností Petriho sítí. Výhodou konceptu samostatné knihovny je rovněž snadná dostupnost tohoto matematického modelu pro další projekty. Celkový náhled na hierarchii tříd knihovny *GPNS* je uveden na **obr. 4-1**. Zdrojové soubory knihovny *GPNS* jsou k dispozici v příloze ***A***.

4.2.1. Třída *node*

Základní třídou knihovny *GPNS* je třída *node*, která zapouzdřuje vlastnosti a metody společné pro všechny typy míst a přechodů jako jedinečný identifikátor *ID*, souřadnice objektu v grafickém editoru Petriho sítí nebo virtuální metody pro export objektu do souboru v *PNML* formátu atd. Třída *node* a její potomci jsou graficky znázorněny na **obr. 4-2**.

4.2.2. Třídy *place* a *transition*

Z rodičovské třídy *node* jsou děděním odvozeny třídy *place* a *transition*. Ty reprezentují obecná místa a přechody Petriho sítě. Třída *transiton* navíc umožňuje definovat prioritu daného přechodu pro deterministické řešení efektivního konfliktu v Petriho síti.



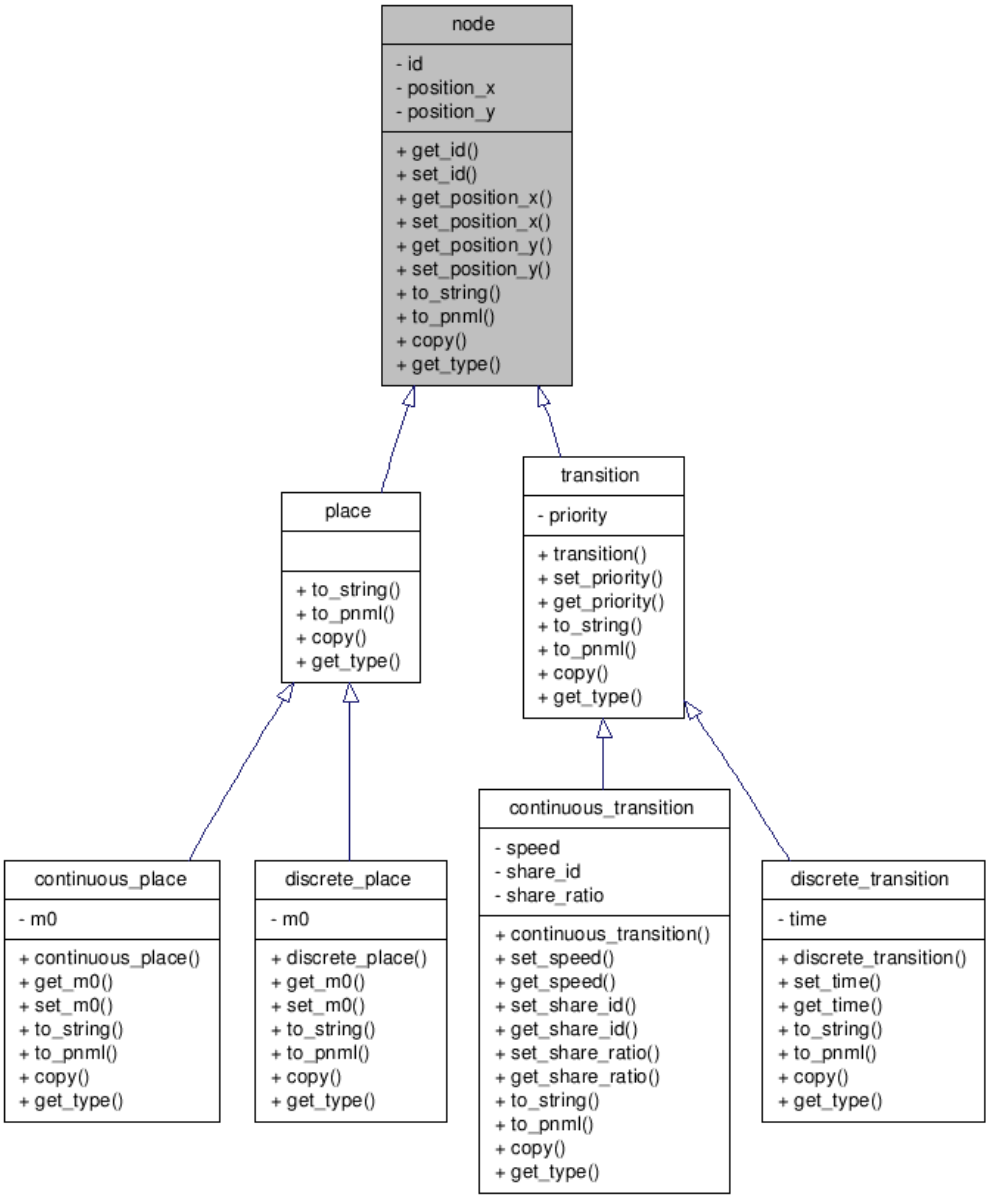
obr. 4-1: struktura knihovny *GPNS*

4.2.3. Třídy *discrete_place* a *discrete_transition*

Třídy *place* a *transition* mají jako potomky své diskrétní protějšky, popisující místa a přechody časovaných Petriho sítí. Třída *discrete_place* s počátečním značením místa m_0 a *discrete_transition* s časem přeskoku přechodu po jeho uvolnění.

4.2.4. Třídy *continuous_place* a *continuous_transtion*

Třídy *continuous_place* a *continuous_transition* jsou rovněž odvozeny od rodičovských tříd *place* a *transition*. Popisují místa a přechody spojité Petriho sítě. Spojité místo se od diskrétního liší pouze tím, že jeho počáteční značení může nabývat hodnot z oboru nezáporných reálných čísel. Spojitý přechod se pak vyznačuje maximální rychlostí přechodu, označením skupiny přechodů pro proporcionální řešení konfliktu a také poměrem, kterým se na tomto řešení podílí.

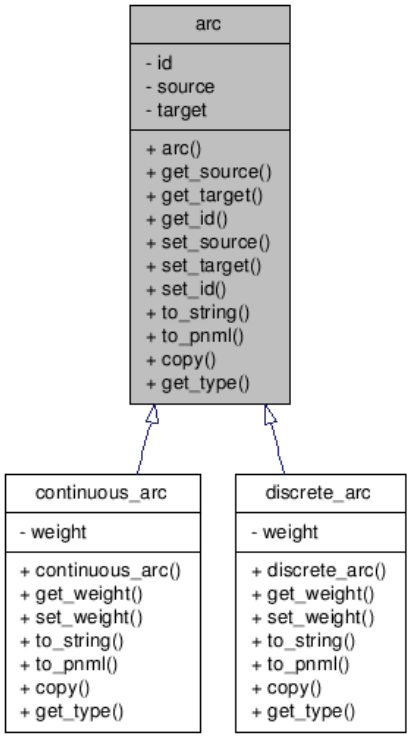


obr. 4-2: třídy uzlů Petriho sítě[3]

4.2.5. Třídy *arc*, *discrete_arc* a *continuous_arc*

Implementace tříd *arc*, *discrete_arc* a *continuous_arc* je pro uživatele knihovny skryta a je použita pouze pro interní potřeby knihovny *GPNS*. Jelikož zmíněné třídy obsahují proměnné typu ukazatel na třídu *node* (zdroj a cíl hrany), měly by být vybaveny neimplicitním kopírovacím konstruktorem, který by zajistil i kopírování objektů, na které tyto třídy odkazují. V našem případě je však toto, jinak objektově správné chování, nevhodné (kopíruji hranu, nikoliv trojici místo-hrana-přechod). Ukrytí implementace před uživatelem tedy znemožní samostatné použití těchto tříd a vznik programátorské chyby. Jejich grafické znázornění je uvedeno na **obr. 4-3**.

[3] Obrázky **obr. 4-2**, **obr. 4-3** a **obr. 4-5** jsou generovány systémem *Doxygen*. U metod tříd není uveden datový typ vstupních a výstupních parametrů. Některé metody tak mohou být v rámci jedné třídy uvedeny vícekrát, neboť se liší právě svými parametry. Detailní informace o všech třídách lze nalézt v příloze ***A***.



obr. 4-3: třídy hran Petriho sítí

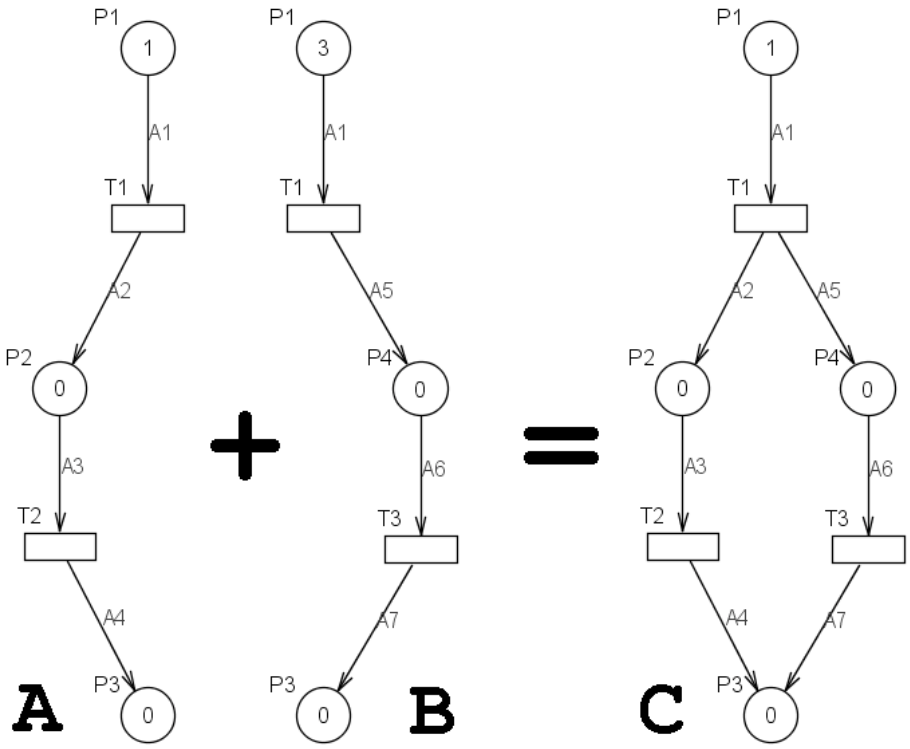
4.2.6. Třída PN

Třída *PN* zapouzdřuje všechny výše zmíněné třídy a poskytuje uživateli metody pro přidávání nových míst a přechodů do sítě, vytváření hran mezi nimi, spojování dvou objektů třídy *PN* na základě shodných identifikátorů míst a přechodů, sestavení matic *Pre*, *Post* a přístup k vlastnostem míst a přechodů v podobě řídkých vektorů.

Vedle bezparametrického konstruktoru třídy *PN* je implementován také konstruktor *PN(std::string)*, umožňující import Petriho sítě ze souboru ve formátu *PNML*. Tato metoda využívá pro zpracování *PNML* dokumentů knihovnu *EXPAT*. Vstupním parametrem je řetězec znaků, udávající absolutní cestu a jméno souboru, ze kterého má být Petiho síť načtena.

Implementována je rovněž metoda *to_pnml(std::string)*, umožňující uložit objekt *PN* zpět do souboru v *PNML* formátu. Vstupním parametrem je řetězec znaků, udávající absolutní cestu a jméno souboru, do kterého má být Petriho síť uložena. Spolu s metodou pro import Petriho sítě z *PNML* souboru tak výrazně přispívá k použitelnosti knihovny s ostatními nástroji, jako grafickými editory Petriho sítí atd.

Další důležitou metodou je metoda pro připojení jiné Petriho sítě ke stávající síti *PN* join(PN*)*. Spojování Petriho sítí je velmi důležitou metodou právě pro úlohy, jakými jsou simulace transportních sítí. Vstupním parametrem této metody je ukazatel na objekt třídy *PN*, který má být připojen ke stávajícímu objektu, vlastnícímu metodu. Výstupem je pak ukazatel na zcela nový objekt třídy *PN*, vzniklý jejich spojením. Původní objekty zůstávají zachovány. Samotné spojování probíhá na základě shodných identifikátorů míst a přechodů v obou sítích. Graficky je třída znázorněna na **obr. 4-5**.



obr. 4-4: spojování Petriho sítí

Na **obr. 4-4** je uveden příklad jednoduchého spojení dvou Petriho sítí. Předpokládejme, že Petriho sítě *A* a *B* jsou již reprezentovány ve formě objektu *PN* v paměti programu a síť *C* vznikne spojením těchto dvou sítí. Obě sítě mají shodné identifikátory míst *P1*, *P3*, přechodu *T1* a hrany *A1*. Volána je metoda sítě *A*. Algoritmus vychází ze struktury sítě *A*, do které přidává místa a přechody ze sítě *B*, jejichž identifikátor není v síti *A* uveden. Přitom zachovává spojení míst a přechodů hranami. Pokud mají dva prvky sítí stejné identifikátory, ale rozdílné vlastnosti (např. počáteční značení místa, čas přechodu), jsou ve výsledné síti uvažovány vlastnosti prvku, jenž patří do sítě, které náleží metoda pro spojování sítí. Místo *P1* v síti *C* má tedy počáteční značení $m_0 = 1$, shodně s místem *P1* v síti *A*. Vlastnosti místa *P1* v síti *B* jsou tak ignorovány.

V **C++ 4-1.** je uveden příklad kódu v jazyku *C++*, který řeší spojování Petriho sítí z **obr. 4-4**. Petriho sítě *A* a *B* jsou nejprve pomocí parametrického konstruktoru načteny do paměti programu z *PNML* souborů *join_01.xml* a *join_02.xml*. Voláním metody *PN* join(PN*)* instance *A* s ukazatelem na instanci *B* jako parametr, vznikne síť *C* jejich spojením.

```
#include "GPNS.h"
using namespace std;

int main(int argc, char* argv[])
{
	PN* A = new PN("C:/join_01.xml");
	PN* B = new PN("C:/join_02.xml");

	PN* C = A->join(B);

	cout << A << endl;
	delete A;

	cout << B << endl;
	delete B;

	cout << C << endl;
	delete C;

	system("PAUSE");
}
```

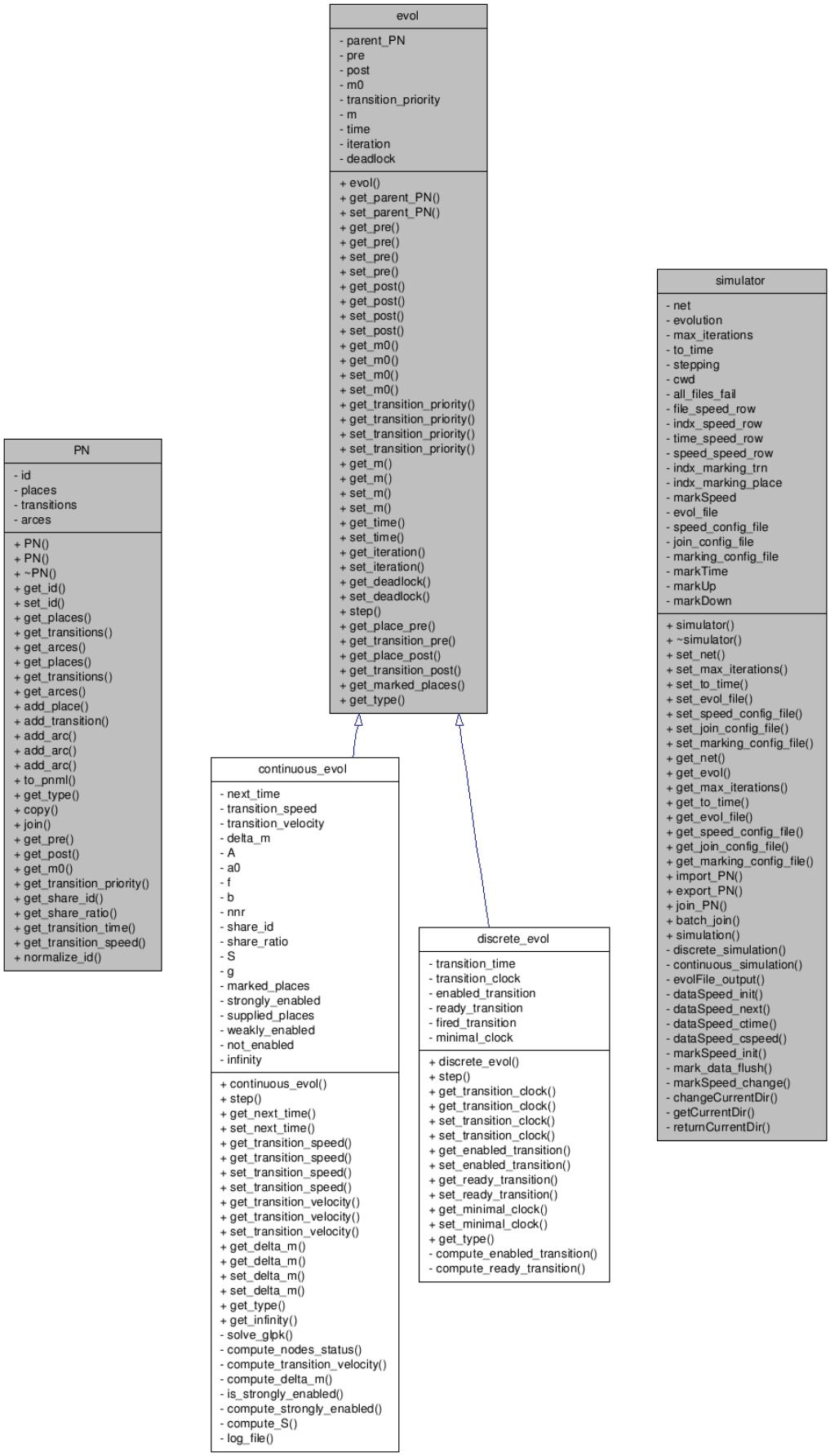
C++ 4-1: spojování Petriho sítí

Následně je možné se všemi třemi sítěmi dále pracovat. V našem případě jsme použili přetížený operátor << třídy *PN* pro postupný výpis struktury všech sítí na standardní výstup. Po výpisu detailních informací o každé síti je síť vymazána z paměti programu.

4.2.7. Třída *evol*

Třída *evol* je rodičovskou třídou pro třídy *discrete_evol* a *continuous_evol*. Pomocí svého konstruktoru odvozena ze třídy *PN*. Jedná se o matematickou reprezentaci Petriho sítě, vhodnou pro výpočet vývoje značení. Zapouzdřuje proměnné a metody společné pro výpočet vývoje jak časované, tak spojité Petriho sítě. Uchovává strukturu sítě vyjádřenou maticemi *Pre* a *Post*, počáteční a aktuální značení sítě, priority přechodů a aktuální čas a iteraci vývoje.

Mezi jednotlivými iteracemi vývoje značení sítě se přechází voláním virtuální metody *evol::step()*. Po ukončení každé iterace je uživateli umožněno zasahovat do proměnných objektu *evol* i jeho potomků *discrete_evol* a *continuous_evol*. Lze tak realizovat nelineární chování Petriho sítě, změnu značení sítě v závislosti na libovolné události a dokonce i změnu struktury samotné sítě. Této skutečnosti bylo využito např. při implementaci třídy *simulator*, kde byla použita pro řízení maximálních rychlostí přechodů v závislosti na čase simulace (viz. kapitola **4.2.10**). Aktuální čas a iterace vývoje značení jsou přes metody dostupné v proměnných *iteration* a *time*. Pokud vývoj značení sítě dospěje do stavu *deadlock*, je příznak *deadlock* nastaven na hodnotu *true*. Grafické znázornění třídy je uvedeno na **obr. 4-5**.

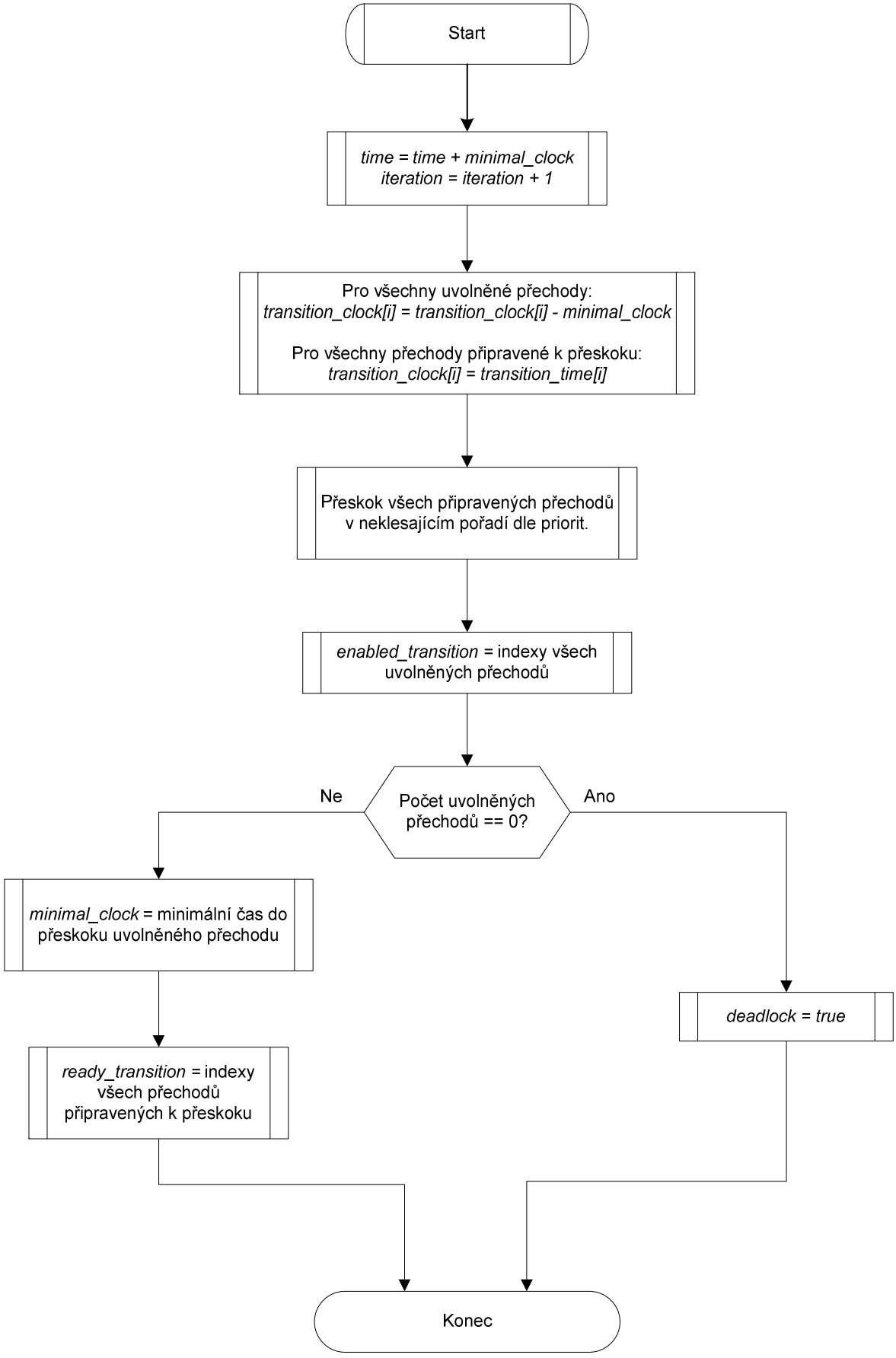


obr. 4-5: třída *PN*, *simulator*, *evol* a její potomci

4.2.8. Třída *discrete_evol*

Tato třída je potomkem třídy *evol* a je určena pro výpočet vývoje značení časované Petriho sítě. K tomu využívá knihovnu *uBLAS* pro maticový počet. Efektivní konflikt v síti je řešen řazením přechodů připravených k přeskoku do nerostoucí posloupnosti dle priorit, jejich postupným přeskokem a kontrolou, zda další přechod v této posloupnosti je stále uvolněn. Grafické znázornění třídy je uvedeno na **obr. 4-5**.

Na **obr. 4-6** je uveden vývojový diagram metody *discrete_evol::step()*. Je třeba poznamenat, že v samotném konstruktoru třídy *discrete_evol* jsou již určeny výchozí hodnoty všech proměnných, vyjadřující stav vývoje značení sítě, neboť se jedná o nultou iteraci tohoto vývoje v čase $t = 0$. Určena je tedy množina uvolněných přechodů *enabled_transition*, minimální čas zbývající do přeskoku uvolněných přechodů *minimal_clock* a množina přechodů připravených k přeskoku *ready_transition*. V rámci volání metody *discrete_evol::step()* dojde k přechodu do další iterace a čas vývoje značení je navýšen o minimální čas, zbývající do přeskoku některého z přechodů. Následně je snížen čas všech uvolněných přechodů o tento minimální čas a pokud byl přechod připraven na přeskok, je nastaven jeho čas na původní hodnotu. Dále jsou přeskočeny všechny připravené přechody v neklesající posloupnosti dle jejich priorit. Pro další iteraci je určena množina všech uvolněných přechodů. Pokud je tato množina prázdná, je indikován stav *deadlock*. V opačném případě je určen minimální čas do přeskoku dalších přechodů a množina přechodů připravených na přeskok.



obr. 4-6: vývojový diagram metody *discrete_evol::step()*

4.2.9. Třída *continuous_evol*

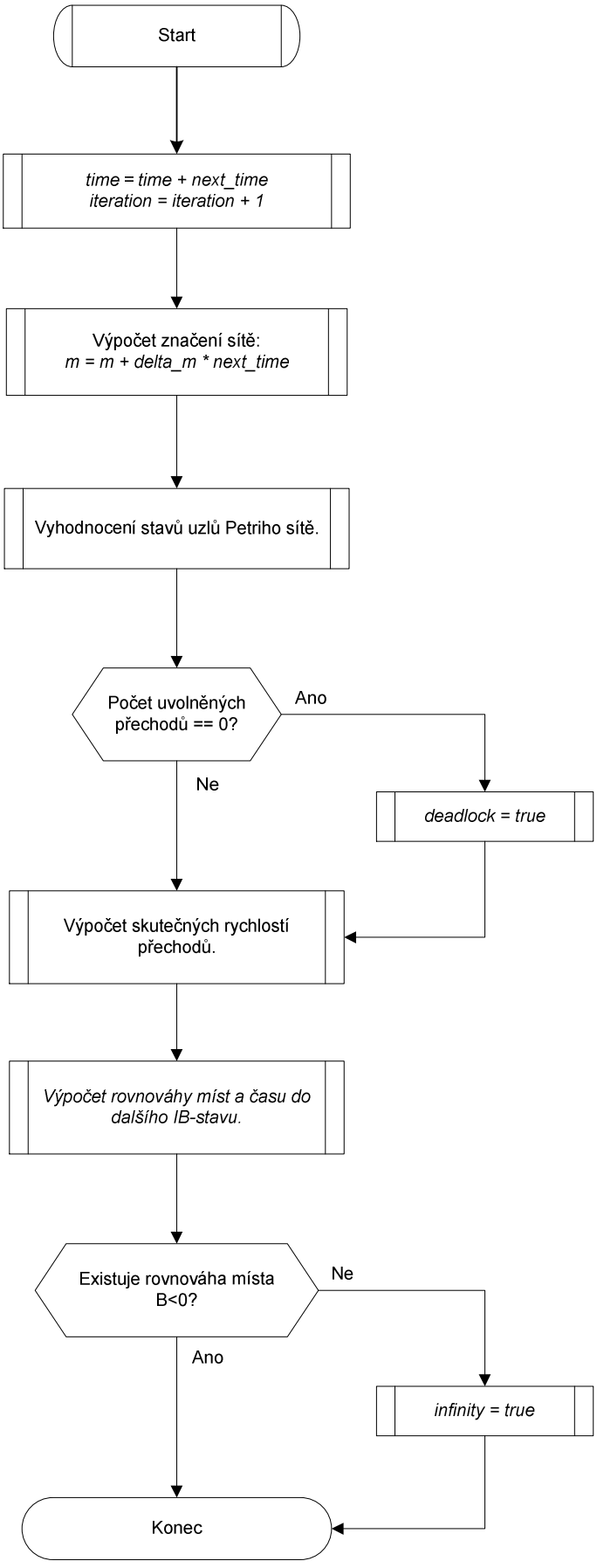
Tato třída je rovněž potomkem třídy *evol* a je určena pro výpočet vývoje značení spojité Petriho sítě. Základní koncept je převzat ze třídy *discrete_evol* a mezi jednotlivými IB-stavy (viz. kapitola **2.3.1**) vývoje je přecházeno voláním metody *continuous_evol::step()*. Tato metoda je schopna řešit efektivní konflikt síti jak prioritní tak proporcionální metodou. Proporcionální metoda navíc akceptuje libovolný podíl přechodu na řešení konfliktu, který může být rozdílný od maximální rychlosti přechodu. Grafické znázornění třídy je uvedeno na **obr. 4-5**.

Oproti třídě *evol* zapouzdřuje navíc vektor maximálních rychlostí přechodů *transition_speed*, vektor skutečných rychlostí přechodů *transition_velocity*, čas zbývající do dalšího IB-stavu *next_time* vektor derivací značení míst v čase (*balance*) *delta_m* a příznak stavu *infinity*. Ten nabývá hodnoty *true*, pokud se další IB-stav nachází v čase $t = \infty$. Ostatní proměnné uvedené na **obr. 4-5** jsou uživateli skryté a slouží pro interní potřeby třídy *continuous_evol*.

Stejně jako v předchozím případě jsou hodnoty všech proměnných, vyjadřujících aktuální stav vývoje Petriho sítě, určeny již v konstruktoru této třídy v rámci nulté iterace. Na **obr. 4-7** je znázorněn vývojový diagram metody *continuous_evol::step()*. Po volání této metody dojde k přechodu do další iterace a výpočtu značení sítě. Dále jsou pak určeny nové hodnoty vnitřních proměnných třídy *continuous_evol* tak, aby byla připravena na další spuštění a mohla poskytnout uživateli aktuální informace o stavu sítě.

Vyhodnocení stavů uzlů sítě probíhá dle algoritmu zmíněném v kapitole **2.3.1**. Jsou tak určeny množiny značených a zásobených míst a dále množiny silně uvolněných, slabě uvolněných a neuvolněných přechodů. Pokud je množina uvolněných přechodů prázdná, je uživateli indikován stav *deadlock* nastavením hodnoty proměnné *deadlock* na *true*.

Dále probíhá výpočet skutečných rychlostí přechodů, založený na řešení problému lineárního programování. K jeho řešení využívá metoda knihovnu *GLPK*, kterou lze případně nahradit jiným nástrojem pro řešení problému lineárního programování. Na základě skutečných rychlostí přechodů je určena derivace značení míst v čase a čas zbývající do dalšího IB-stavu. Pokud neexistuje místo, jenž by mělo zápornou derivaci značení, je nastavením proměnné *infinity* na logickou hodnotu *true* signalizována nemožnost určení času dalšího IB-stavu.



obr. 4-7: vývojový diagram metody *continuous_evol::step()*

4.2.10. Třída *simulator*

Třída *simulator* je pokročilou třídou pro simulace Petriho sítí. Zapouzdřuje jak třídu *PN*, tak třídu *evol* na jejímž místě může být definována diskrétní i spojitá verze. Je koncipována jako rozhraní knihovny s koncovou aplikací. Umožňuje uživateli zapisovat průběh výpočtu vývoje Petriho sítě do textového souboru, který slouží pro pozdější analýzu simulací. Na základě dávkového konfiguračního souboru lze rovněž spojovat více Petriho sítí do jedné sítě. Pokročilými metodami je pak možné řídit maximální rychlosti přechodů spojité Petriho sítě v závislosti na čase simulace. Grafické znázornění třídy je uvedeno na **obr. 4-5**.

Soubor, do kterého má být průběh simulace zaznamenáván, lze definovat pomocí metody *simulator::set_evol_file(std::string)* s parametrem, udávajícím absolutní cestu a jméno souboru. Při každém volání metody *simulator::simulation()* jsou do tohoto souboru zapsány informace o aktuálním značení sítě v daném čase. Pokud se jedná o vývoj spojité Petriho sítě, jsou navíc vypisovány i skutečné rychlosti přechodů.

```
<
This is evolution history of PN generated by GPNS_console demo application.
NetID: CCPN1
NetM0: [3](0,0,20)
ToTime: 10000
MaxIterations: 100
Order of nodes:
TIME, P3, P2, P1, T2, T1
>
0,0,0,20,1,2
10,10,10,0,1,0
20,20,0,0,0,0
// deadlock
```

TXT 4-1: výstupní soubor simulace

V **TXT 4-1** je uveden záznam vývoje spojité Petriho sítě z **obr. 2-3**. V hlavičce souboru jsou uvedeny základní informace o simulaci jako je identifikátor Petriho sítě, její počáteční značení, maximální čas a iterace simulace a význam sloupců v datové části souboru. Řádky představují jednotlivé iterace vývoje sítě. První sloupec reprezentuje čas simulace, sloupce P3, P2 a P1 značení těchto míst a sloupce T2 a T1 pak skutečné rychlosti těchto přechodů.

Pro hromadné spojování více Petriho sítí je určena metoda *simulator::batch_join()*. Před voláním této metody, je třeba nastavit úplnou cestu a jméno konfiguračního souboru pomocí metody *set_join_config_file(std::string)*. Konfigurační soubor je textový soubor s jednoduchou syntaxí. Na jeho základě je možné do operační paměti načíst více Petriho sítí z *PNML* souborů pod libovolnými identifikátory a následně tyto sítě spojit. Tento proces má

tři fáze a konfigurační soubor je čten dvěma průchody. Prázdné řádky souboru a mezery jsou ignorovány. V první fázi algoritmus prochází konfigurační soubor a hledá příkazy se syntaxí:

id < *path*,

kde *id* je identifikátor výsledné instance třídy *PN* a *path* je relativní nebo absolutní cesta k *PNML* souboru, ze kterého má být tato instance importována. Jako výchozí adresář pro relativní cestu je uvažován adresář, ve kterém se nachází konfigurační soubor. Takto importovaná Petriho síť má tedy jedinečný identifikátor *id* a identifikátory všech prvků sítě jsou upraveny do formátu *id.node_id*, kde *node_id* je původní identifikátor daného prvku.
Ve druhé fázi dochází k přejmenování uzlů načtených Petriho sítí tak, aby mohly být později spojeny. Algoritmus hledá v konfiguračním souboru příkazy se syntaxí:

netA_ID.nodeA_ID = netB_ID.nodeB_ID,

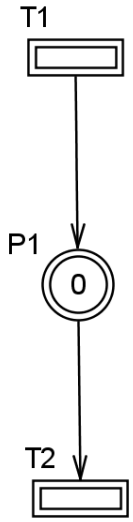
kde *netA_ID* je identifikátor sítě a *nodeA_ID* je identifikátor uzlu této sítě, který má být přejmenován na stejný identifikátor, jako *nodeB_ID* v síti *netB_ID*. Ve třetí fázi algoritmus postupně spojuje Petriho sítě pomocí výše popsané metody *PN::join(PN*)* v pořadí, v jakém byly načteny v konfiguračním souboru.

```
netA < C:\models\model_01.xml
netB < C:\models\model_01.xml
netC < .\models\specific\model_02.xml

netA.T1 = netB.T2
netB.T1 = netC.T2
```

TXT 4-2: konfigurační soubor dávkového spojování sítí

V **TXT 4-2** je uveden příklad konfiguračního souboru metody *simulator::batch_join()*. Sítě *netA* a *netB* jsou načteny ze shodného *PNML* souboru *C:\models\model_01.xml*, síť *netC* pak pomocí relativní adresy ze souboru *.\models\specific\model_02.xml*. Následně je přechod *netA.T1* přejmenován na *netB.T2* a přechod *netB.T1* na *netC.T2*. Dále jsou Petriho sítě postupně spojovány na základě shodných identifikátorů uzlů v pořadí, v jakém byly uvedeny v konfiguračním souboru, tedy *netA*, *netB* a *netC*.

Další pokročilou vlastností třídy *simulator* je možnost řídit maximální rychlost libovolných přechodů v závislosti na čase simulace. Každému přechodu může být přiřazen datový soubor, jenž specifikuje průběh maximální rychlosti přechodu v čase.



obr. 4-8: spojitá Petriho síť

Toto přiřazení je definováno v konfiguračním souboru nastavovaném metodou *set_speed_config_file(std::string)*. Vstupním parametrem této metody je opět úplná cesta ke konfiguračnímu souboru. Ten používá obdobnou syntaxi jako soubor pro dávkové spojování sítí:

trn_id < *path*,

kde *trn_id* je identifikátor přechodu a *path* je absolutní nebo relativní cesta k datovému souboru. Příklad takového konfiguračního souboru je uveden v **TXT 4-3**.

```
T1 < dp_sotd_T1.txt
T2 < dp_sotd_T2.txt
```

TXT 4-3: datový soubor závislosti *rychlost - čas*

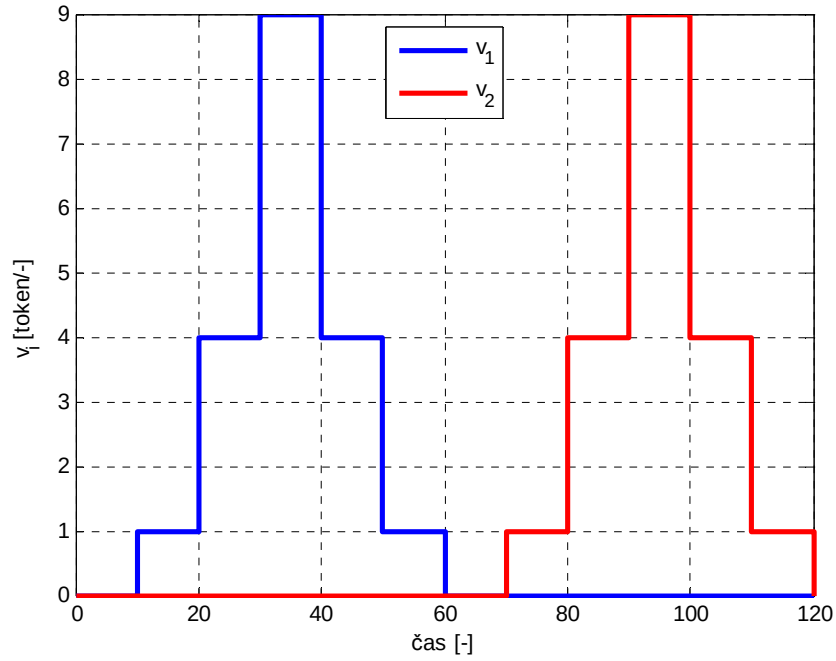
```
0,0
10,1
20,4
30,9
40,4
50,1
60,0
```

TXT 4-4: dp_sotd_T1.txt

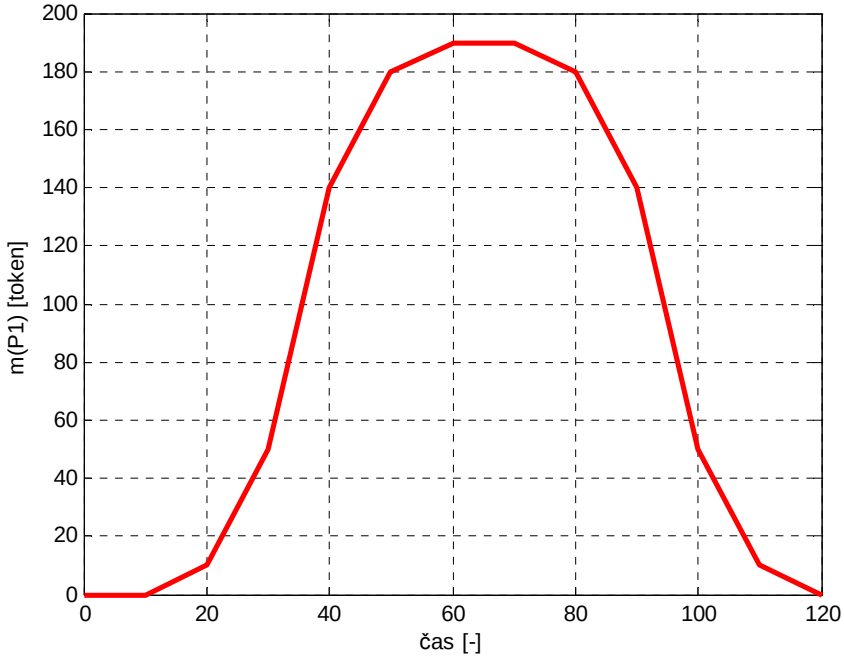
```
0,0
70,1
80,4
90,9
100,4
110,1
120,0
```

TXT 4-5: dp_sotd_T2.txt

Datový soubor je prostý textový soubor, obsahující dva sloupce oddělené čárkou. První sloupec reprezentuje čas simulace a druhý sloupec maximální rychlost daného přechodu. Čas musí být uveden v rostoucí posloupnosti. Uvažujme nyní Petriho síť z **obr. 4-8**, konfigurační soubor **TXT 4-3** a odpovídající datové soubory **TXT 4-4** a **TXT 4-5**. Vývoj skutečných rychlostí přechodů této sítě je znázorněn na **obr. 4-9** a vývoj značení pak na **obr. 4-10**.

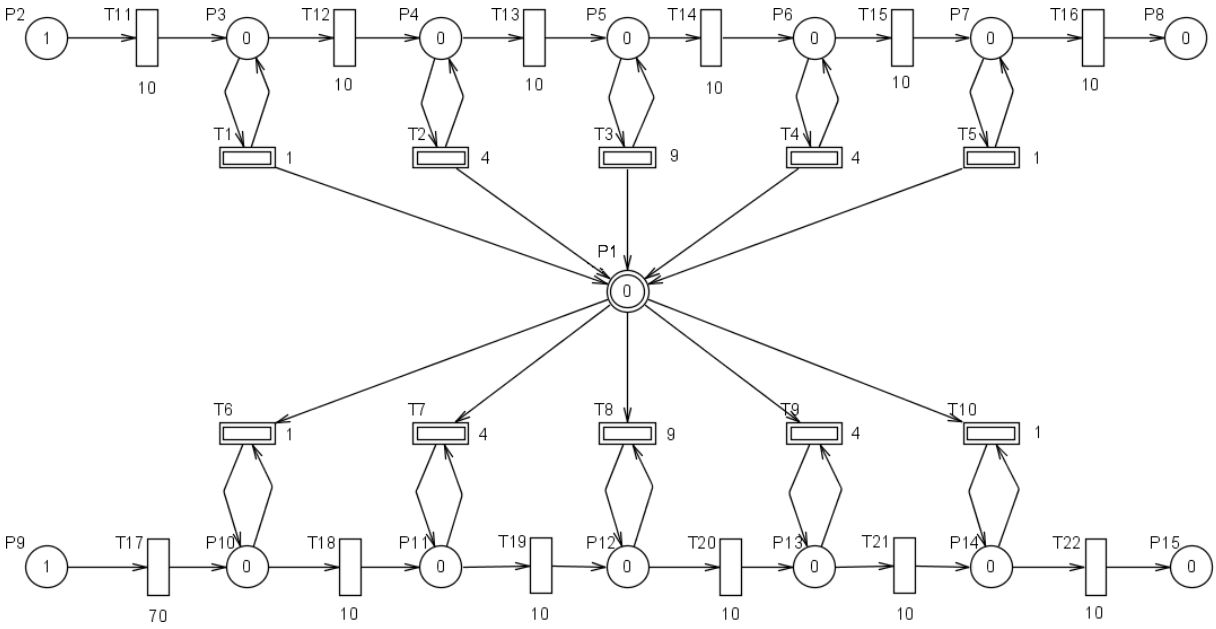


obr. 4-9: vývoj skutečných rychlostí přechodů



obr. 4-10: vývoj značení Petriho sítě

Pokud bychom chtěli realizovat takový vývoj značení v místě P_1 bez uvedeného řízení maximálních rychlostí přechodů, museli bychom použít hybridní Petriho síť, kde by diskrétní část pracovala obdobně jako stavový automat a ovlivňovala tak spojitou část. Příklad takové sítě je uveden na **obr. 4-11**.



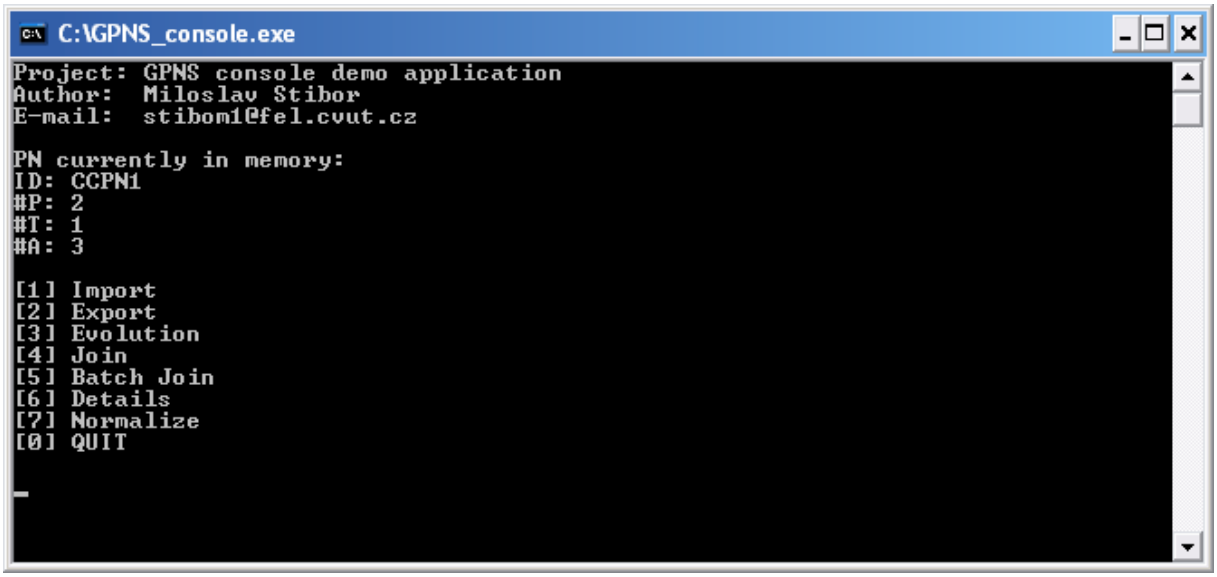
obr. 4-11: hybridní Petriho síť

Spojité místo P_1 na **obr. 4-11** odpovídá místu P_1 na **obr. 4-8**, spojité přechody T_1-T_5 reprezentují přechod T_1, přechody T_6-T_{10} pak reprezentují přechod T_2. Maximální rychlosti těchto spojitých přechodů odpovídají požadovaným rychlostem dle konfiguračního souboru **TXT 4-4**. resp. **TXT 4-5**. Diskrétní místa P_2-P_8 a přechody T_{11}-T_{16} reprezentují první stavový automat, jenž postupně uvolňuje spojité přechody T_1-T_5 a realizuje tak vývoj značení místa P_1 v čase t = 0 až t = 60. Druhý stavový automat je pak reprezentován diskrétními místy P_9-P_{15} a přechody T_{17}-T_{22}. Na jeho základě jsou postupně uvolňovány spojité přechody T_6-T_{10} a realizuje vývoj značení místa P_1 v čase t = 70 až t = 120.

Prostým porovnáním spojité sítě z **obr. 4-8** a hybridní sítě z **obr. 4-11** je patrný dramatický rozdíl v požadavcích na množství použitých uzlů sítě, jenž ve výsledku realizují totožnou funkci. Zatímco spojitou Petriho sítí s řízením maximálních rychlostí přechodů lze uvedenou funkci realizovat pomocí jednoho místa a dvou přechodů u hybridní sítě je zapotřebí patnácti míst a dvaceti dvou přechodů.

5. Koncová aplikace knihovny

Pro předvedení hlavních funkcionalit knihovny *GPNS* byla v jazyku *C++* rovněž implementována konzolová aplikace *GPNS_console*. Ta umožňuje uživateli načítání Petriho sítí z *PNML* souborů, zpětné ukládání do *PNML* souborů, spouštění výpočtu vývoje značení Petriho sítí, jednoduché i dávkové spojování sítí, výpis detailních informací o struktuře sítě a normalizaci identifikátorů sítě. V průběhu simulace je uživatel informován o aktuální iteraci, času simulace a reálném času. Tato aplikace byla použita pro všechny výpočty značení a spojování Petriho sítí, uvedé v této práci. Úvodní obrazovka aplikace po načtení Petriho sítě je zobrazena na **obr. 5-1**.



obr. 5-1: konzolová aplikace

Tato aplikace samozřejmě nepokrývá všechny možnosti knihovny *GPNS*. Jako navazující projekt by bylo možné implementovat aplikaci s grafickým uživatelským rozhraním, které by umožnilo přímo modelovat Petriho sítě a interaktivně sledovat vývoj jejího značení. Vzhledem k širokým možnostem knihovny by však i tato aplikace byla značně rozsáhlým projektem. Dále byl implementován jednoduchý skript v jazyku *MATLAB*, který umožňuje snadnou grafickou interpretaci vývoje značení Petriho sítí na základě výstupního souboru výše popsané třídy *simulator*. Zdrojové soubory koncové aplikace, její spustitelná verze a uvedený *MATLAB* skript jsou k dispozici v příloze ***A***.

6. Simulace transportních sítí

Dynamický rozvoj průmyslu, telekomunikací, dopravy a dalších odvětví s sebou nevyhnutelně přináší zvyšující se požadavky na kvalitu návrhu, realizace, údržby a případného vylepšování již existujících transportních sítí, se kterými tyto obory pracují. Zejména u projektů většího rozsahu jsou simulace téměř jedinou možnou cestou, jak ověřit správnost návrhu a odstranit případné nedostatky ještě před samotnou realizací projektu. S rozvojem výpočetní techniky se simulace staly velmi přesnou a rychlou metodou prvotní analýzy projektu. V porovnání s celkovými náklady na realizaci projektu navíc mnohdy velmi levnou metodou.

Ve většině zmíněných oborů proto vzniká celá řada simulačních nástrojů, které se liší jak přístupem k problematice, tak specifickými vlastnostmi takovým způsobem, aby co nejlépe odpovídaly požadavkům, kladeným na různorodé aplikace. Například pro simulace městské dopravy se stal uznávaným standardem systém *AIMSUN NG* [Aimsun], který je dostupný již ve své šesté verzi. Jedná se o profesionální, komplexní systém nástrojů pro simulace a analýzu dopravních sítí. Skrze přehledné grafické rozhraní umožňuje uživateli interaktivně procházet simulovanou oblastí. Nabízí rovněž možnost realizace adaptivního řízení dopravy atp. Dalším nástrojem, který již není tak úzce zaměřen je *OMNeT++* [OMNeT++]. Přes své původní určení pro modelování a simulace komunikačních sítí je díky své flexibilitě použitelný i pro analýzu hardwarových řešení, komplexních sítí informačních technologií, teorii front atp. Jedná se o nekomerční a volně dostupný nástroj, pracující na platformách Unix i Windows.

Simulace transportních sítí, konkrétně i městské dopravy, na základě Petriho sítí není zcela novou myšlenkou. Formalizmus Petriho sítí se opírá o silný matematický aparát a vzhledem ke své otevřenosti a variabilitě je do značné míry vhodný např. pro návrh a testování nových algoritmů řízení dopravy. Řada vědeckých skupin se tedy touto problematikou zabývá.

Jmenujme například studii, zabývající se návrhem řídicí strategie pro světly řízené křižovatky pomocí diskrétních Petriho sítí a dynamického přepínání signálních plánů [Gallego]. Obdobným tématem se zabývá i [List]. Modelování základních dopravních prvků pomocí hybridních Petriho sítí popisuje například [Di Febbraro I]. Stejný autor pak v [Di Febbraro II] též popisuje modelování transportních systémů pomocí speciálního typu Petriho sítí, tzv. *HSPN* (*Hybrid Stochoastic Petri Nets*) a jako příklad uvádí model dálniční sítě. Modelováním dopravních sítí pomocí spojitých Petriho sítí se pak zabývá např. [Júlvez].

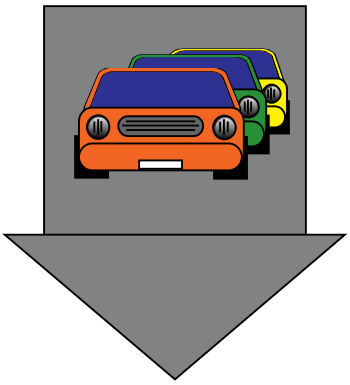
7. Knihovna modelů základních dopravních prvků

V této práci se budeme věnovat simulacím městské dopravy. Za tímto účelem byla navržena knihovna modelů základních dopravních prvků ve formalizmu spojitých Petriho sítí s proměnnou rychlostí přechodů a proporcionálním řešením konfliktu. Tento přístup je vhodný zejména pro simulace oblastí s vysokým počtem vozidel. Diskrétní veličina, jakou je zde počet vozidel, je reprezentována značením Petriho sítě z oboru nezáporných reálných čísel. Jak plyne z kapitoly **2.3.1**, je možné touto aproximací diskrétní veličiny spojitou veličinou dosáhnout značného snížení počtu iterací, potřebných pro výpočet simulace.

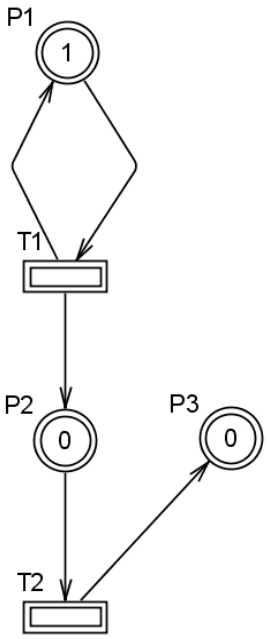
Knihovna obsahuje celkem devatenáct modelů rozličných typů křižovatek, ulic a speciálních prvků, jakými jsou vstupy a výstupy vozidel. Tyto modely, uložené ve formátu *PNML*, pak mohou být spojovány do větších celků například pomocí výše zmíněné koncové aplikace *GPNS_console*, která umožňuje i simulaci výsledného modelu. V této kapitole popíšeme některé z těchto základních prvků, na jejichž základě lze odvodit ostatní modely obsažené v knihovně. Zmíněná knihovna je k dispozici v příloze ***A***. V příloze ***B*** je pak uveden celkový náhled na všechny prvky knihovny.

7.1. Vstup vozidel

Model vstupu vozidel do simulované oblasti, se schématickou značkou na **obr. 7-1**, je realizován Petriho sítí z **obr. 7-2**. Spojení místa P_1 a přechodu T_1 zaručuje, že tento přechod bude vždy silně uvolněn. Toto spojení lze nahradit samostatným zdrojovým přechodem, ale k dosažení co největší kompatibility s ostatními nástroji použijeme tento kompletní model. Maximální a tím i skutečná rychlost tohoto přechodu je řízena daty s požadovaným průběhem rychlosti vstupního proudu vozidel.



obr. 7-1: schématická značka vstupu vozidel

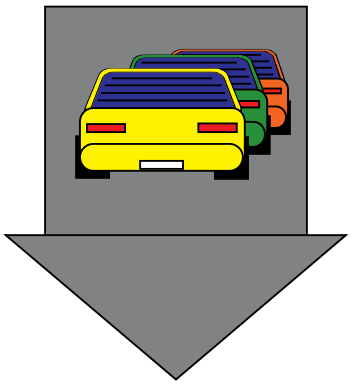


obr. 7-2: model vstupu vozidel

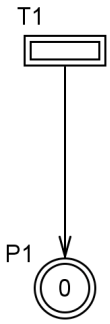
Přechod T_2 slouží k napojení vstupu vozidel do modelu simulované oblasti a jeho rychlost není omezena. Místo P_2 odděluje přechody T_1, T_2. Místo P_3 pak slouží k analýze množství vystupujících vozidel, případně i jejich rychlosti pomocí derivace značení tohoto místa v čase.

7.2. Výstup vozidel

Model výstupu vozidel ze simulované oblasti se schématickou značkou na **obr. 7-3**, je reprezentován Petriho sítí z **obr. 7-4**. Přechod T_1 slouží k napojení modelu na simulovanou oblast a jeho maximální rychlost není omezena. V místě P_1 pak dochází k akumulaci vozidel vystupujících z oblasti. Toto místo lze rovněž použít pro analýzu množství vystupujících vozidel, případně i jejich rychlosti pomocí derivace značení.



obr. 7-3: schématická značka výstupu vozidel



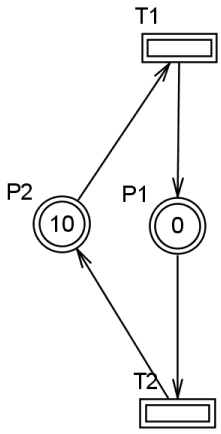
obr. 7-4: model výstupu vozidel

7.3. Ulice

Model ulice o jednom dopravním pruhu je uveden na **obr. 7-6**. Na **obr. 7-5**. je pak uvedeno schéma takové ulice. Přechod T_1 je vstupním a T_2 výstupním přechodem. Tyto přechody slouží k napojení modelu na simulovanou oblast a jejich rychlost není omezena. Značení místa P_1 reprezentuje aktuální počet vozidel v ulici. Místo P_2 je komplementárním místem k P_1. To znamená, že suma značení míst *P1* a *P2* je konstantní a udává celkovou kapacitu ulice v počtu vozidel.

Tímto způsobem lze modelovat ulici o obecném počtu dopravních pruhů libovolného směru. Model jednoduše použijeme tolikrát, kolik dopravních pruhů daná ulice má. Směr konkrétního dopravního pruhu je dán zapojením modelu do simulované oblasti.

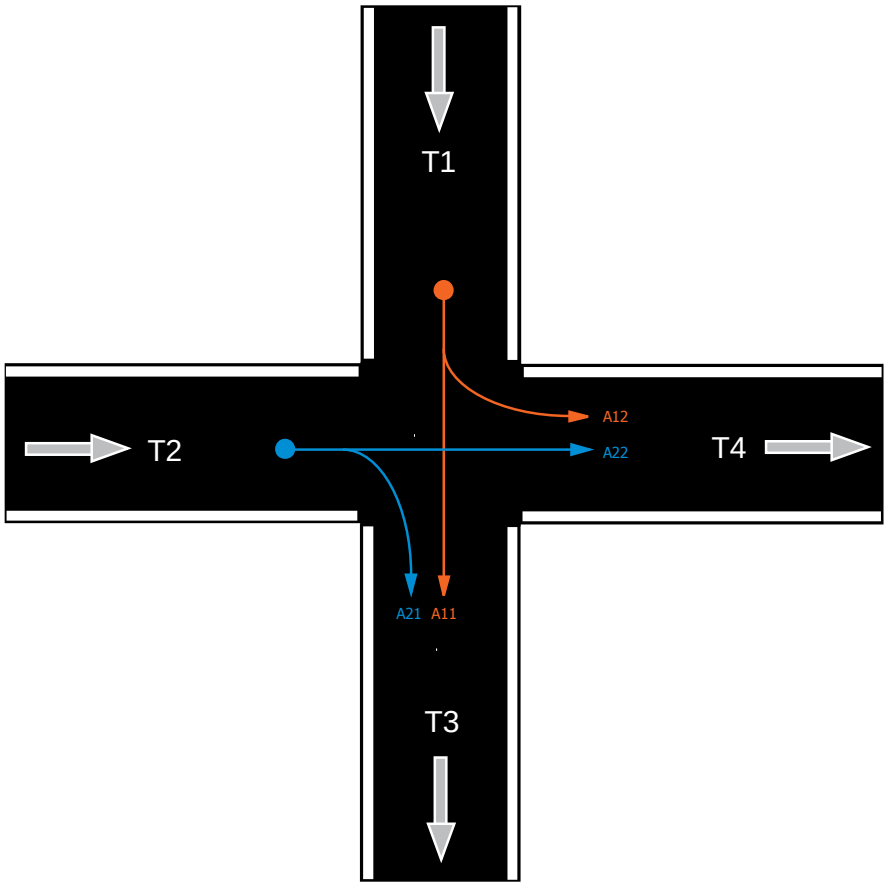
obr. 7-5: schéma ulice



obr. 7-6: model ulice

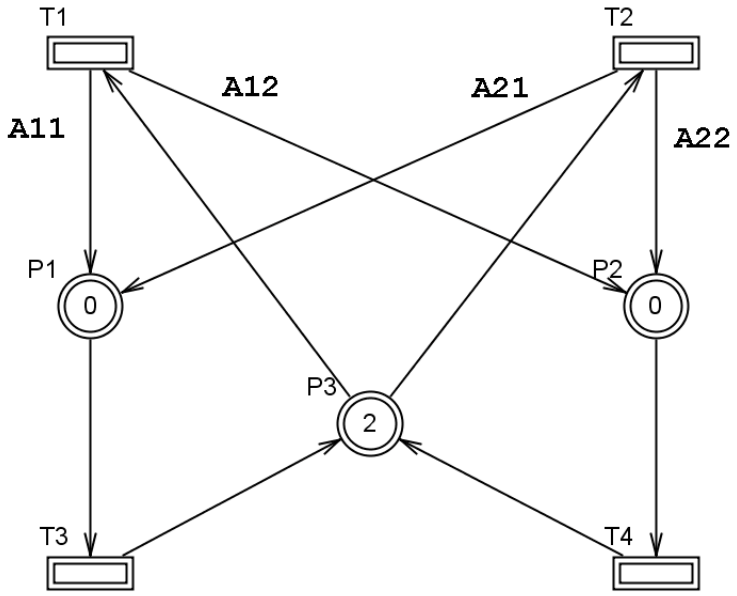
7.4. Křižovatka

V rámci této práce byl navržen model obecné křižovatky jako spojení dvou ulic o libovolném počtu dopravních pruhů. Sestavení tohoto modelu bude prezentováno na jednoduchém příkladu křižovatky dvou ulic o jednom dopravním pruhu z **obr. 7-7**.



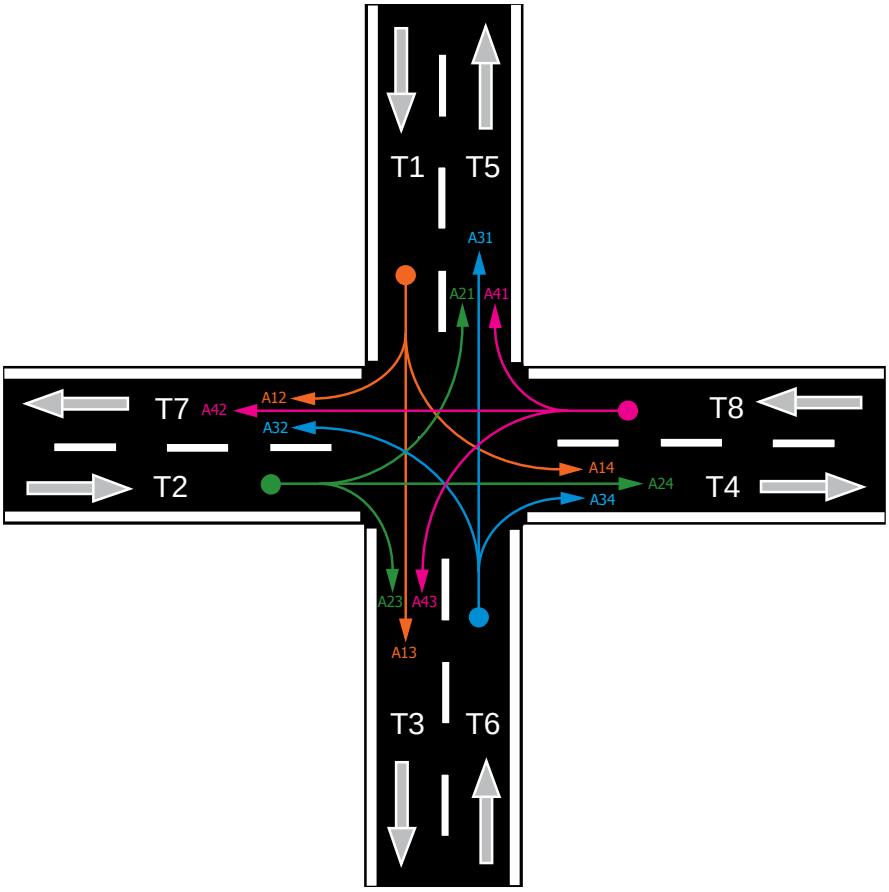
obr. 7-7: schéma křižovatky dvou jednosměrných ulic

Model této křižovatky je uveden na **obr. 7-8**. Přechody T_1 a T_2 jsou vstupními přechody modelu. Jejich maximální rychlost není omezena a slouží k napojení modelu do simulované oblasti. Spojení místa P_1 s přechodem T_3 a místa P_2 s přechodem T_4 reprezentují výstupní část modelu. Přechody T_3 a T_4 jsou tedy výstupními přechody modelu pro spojení se simulovanou oblastí. Hrany vedoucí ze vstupních přechodů T_1 a T_2 do míst P_1 a P_2 rozdělují vstupní proud vozidel do těchto výstupních částí dle schématu na **obr. 7-7**. Váha hran je určena na základě odbočovacích poměrů křižovatky. Místo P_3 je komplementárním místem pro P_1 a P_2. Suma značení míst *P1*, *P2* a *P3* tedy udává celkovou kapacitu prostoru křižovatky v počtu vozidel. Efektivní konflikt v místě P_3 je řešen proporcionální metodou.

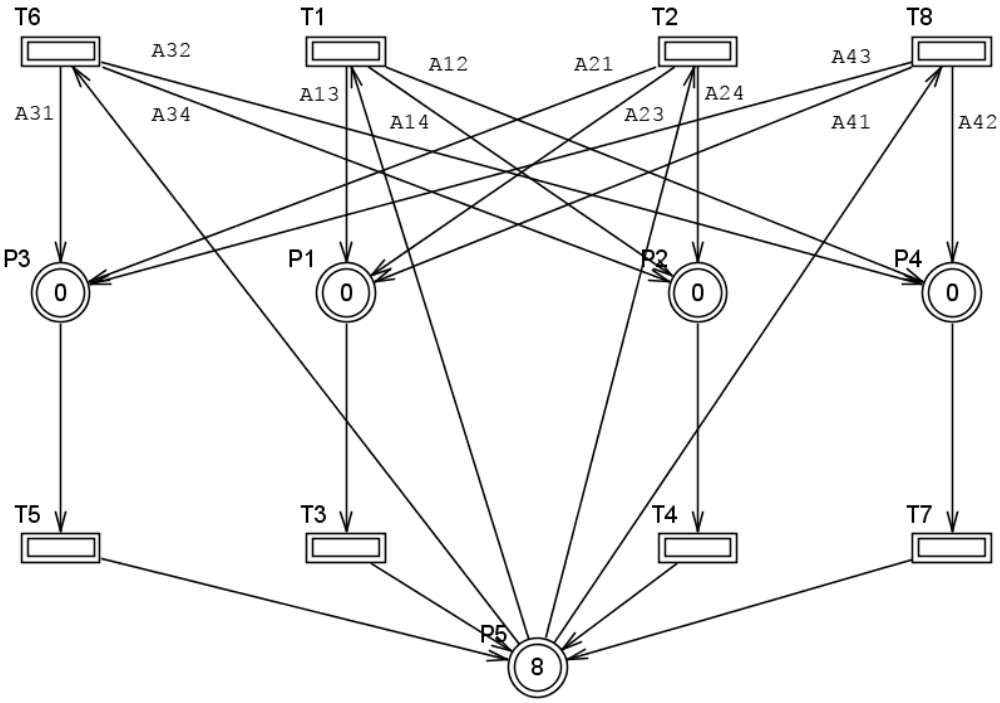


obr. 7-8: model křižovatky dvou jednosměrných ulic

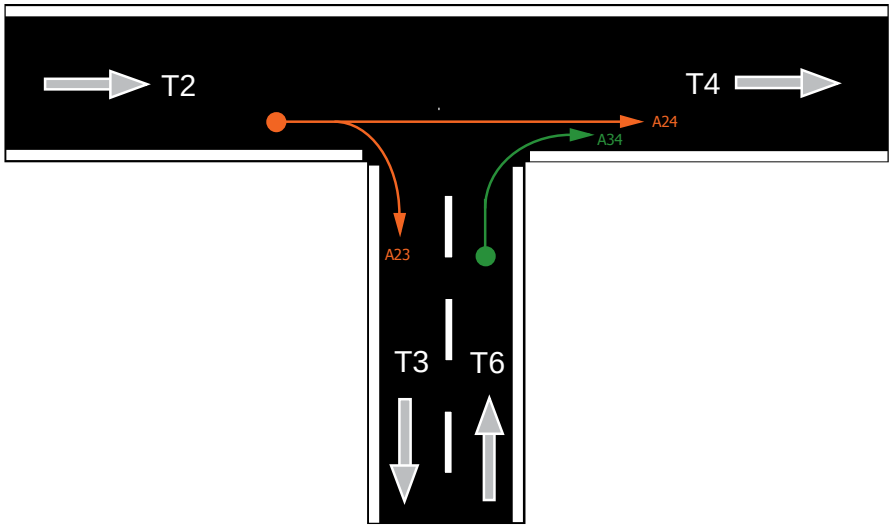
Je zřejmé, že obdobný model lze sestavit prakticky pro libovolnou křižovatku. Pro každý vstup do křižovatky definujeme jeden přechod (T_1, T_2). Pro každý výstup z křižovatky přidáme dvojici místo – přechod (P_1-T_3, P_2-T_4). Vstupní přechody pak spojíme hranami (A_{11}, A_{12}, A_{22}, A_{21}) s těmito výstupními místy na základě odbočovacích poměrů, případně spojení vynecháme nebo položíme váhu hrany rovnu nule. Posledním krokem je pak přidání společného komplementárního místa (P_3), do něhož vstupují hrany ze všech výstupních přechodů a vystupují hrany do všech vstupních přechodů. Značení tohoto místa volíme dle kapacity prostoru křižovatky. Model křižovatky dvou ulic o dvou dopravních pruzích, sestavený dle tohoto postupu, je uveden na **obr. 7-10**. Schéma této křižovatky je uvedeno na **obr. 7-9**. Jedná se o výchozí model knihovny, ze kterého jsou zjednodušením odvozeny všechny ostatní modely křižovatek. Křižovatka tvaru písmene *T*, odvozená z tohoto modelu, je uvedena na **obr. 7-11**. Její model je pak na **obr. 7-12**.



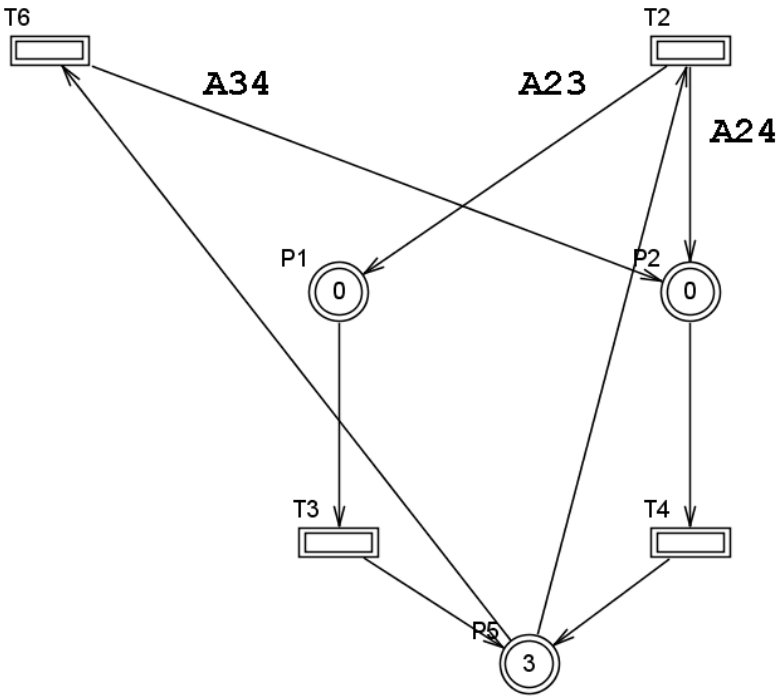
obr. 7-9: schéma křižovatky dvou dvousměrných ulic



obr. 7-10: model křižovatky dvou dvousměrných ulic



obr. 7-11: schéma křižovatky tvaru písmene *T*



obr. 7-12: model křižovatky tvaru písmene *T*

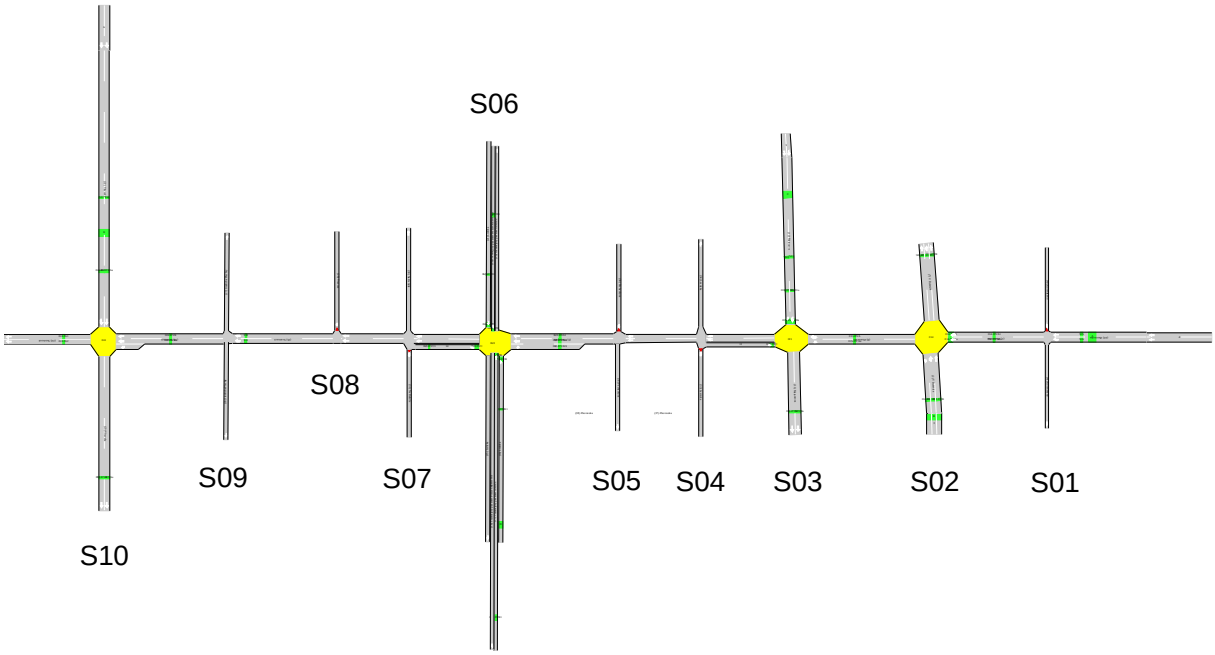
Prezentovaný model obecné křižovatky nerozlišuje mezi řízenou a neřízenou křižovatkou. Neuvažuje tedy signální plán řízené křižovatky, na jehož základě by bylo možné například omezit rychlosti vstupních přechodů modelu. Rovněž není uvažováno dopravní zpoždění v modelu ulice. Z tohoto důvodu nejsou uvedené modely vhodné pro simulace utváření kolon vozidel v simulované oblasti a jsou určeny především pro statistickou analýzu z hlediska rychlosti průjezdu vozidel. Praktické použití těchto modelů je uvedeno v kapitole **8**.

8. Simulace dopravy na reálných datech

V této kapitole je uvedena rozsáhlá příkladová simulace na reálných datech z oblasti městské dopravy. Použity jsou modely dopravních prvků, uvedené v předchozí kapitole. Jako simulační nástroj je použita implementovaná aplikace *GPNS_console*, využívající knihovnu *GPNS* (viz. kapitola **4**).

8.1. Popis simulované oblasti

Simulovanou oblast tvoří celkem deset křižovatek (S01 – S10) z lokality Praha – Smíchov. Jedná se o čtyři řízené (označené žlutou barvou) a šest neřízených křižovatek. V dané oblasti je rozmístěna řada senzorů (označené zelenou barvou), snímajících průměrnou rychlost průjezdu vozidel. Simulace je provedena na datech získaných z těchto senzorů v průběhu jednoho pracovního dne, tedy od půlnoci do půlnoci (0 – 86400s). Na **obr. 8-1** je uveden celkový náhled uvažované oblasti.



obr. 8-1: simulovaná oblast

Všechny křižovatky byly zjednodušeny do podoby s jedním dopravním pruhem. Bylo tak docíleno zmenšení rozsahu celkového modelu. Navíc tak bylo možné z uvažované ulice o více dopravních pruzích vždy vybrat senzor, jehož data jsou úplná a nepoškozená. Vzhledem k rozsáhlosti simulované oblasti popíšeme jednotlivé křižovatky, včetně výsledků simulace, postupně v následujících kapitolách. Náhled na zjednodušené schéma celé uvažované oblasti je k dispozici v příloze ***C***. Celkový model oblasti vzniknul spojením dílčích modelů z výše uvedené knihovny a pro jeho rozsah ho nebudeme uvádět. Vstupními daty jsou aproximovaná data ze senzorů ulic, jenž vstupují do uvažované oblasti (na grafech znázorněny modrým průběhem). Výstupními daty jsou pak výsledky simulace (na grafech znázorněny červeným

průběhem). Příklad dat získaných ze vstupního senzoru je uveden na **obr. 8-5**. Je patrné, že data jsou zatížena vysokou úrovní šumu a nejsou tak vhodná pro přímé řízení maximálních rychlostí přechodů spojité Petriho sítě. Přímím řízením na základě těchto dat by došlo k enormnímu nárůstu počtu IB-stavů simulace. Z tohoto důvodu byla data vždy aproximována schodovitým průběhem o deseti krocích a tento průběh byl následně použit pro řízení maximálních rychlostí přechodů spojité Petriho sítě. Konfigurační a datové soubory pro spojení dílčích modelů do celkového a řízení maximálních rychlostí přechodů jsou uvedeny v příloze ***A***.

Pro účely simulace je nutné rovněž přepočítat rychlost pohybu vozidel na rychlost pohybu tokenů ve spojité Petriho síti dle:

$$w = \frac{v}{(3.6 \cdot d)}\ , \tag{8.1}$$

kde w je rychlost pohybu tokenů ve spojité Petriho síti [*token/s*], v je rychlost pohybu vozidel v simulované oblasti [*km/h*] a d je průměrná předpokládaná délka vozidla včetně rozestupu vozidel [*m*]. Hodnota d je konstantní a pro účely simulace byla uvažována $d = 5m$. Pro grafickou reprezentaci výsledků simulace a porovnání se skutečnými daty ze senzorů je pak dle (8.1) zpětně určena rychlost průjezdu vozidel v jednotách *km/h*.

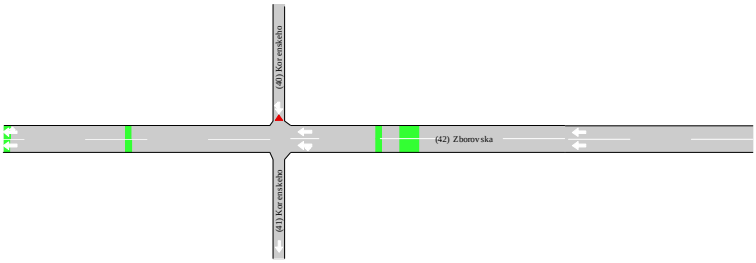
Kapacita modelů ulic, spojujících jednotlivé křižovatky, je určena na základě délky všech mezilehlých ulic dle:

$$c = \frac{l}{d}, \tag{8.2}$$

kde c je celkové počáteční značení míst modelu ulice [*token*] a l je součet délek všech mezilehlých ulic mezi uvažovanými křižovatkami [*m*].

8.2. Křižovatka S01 (*Zborovská – Kořenského*)

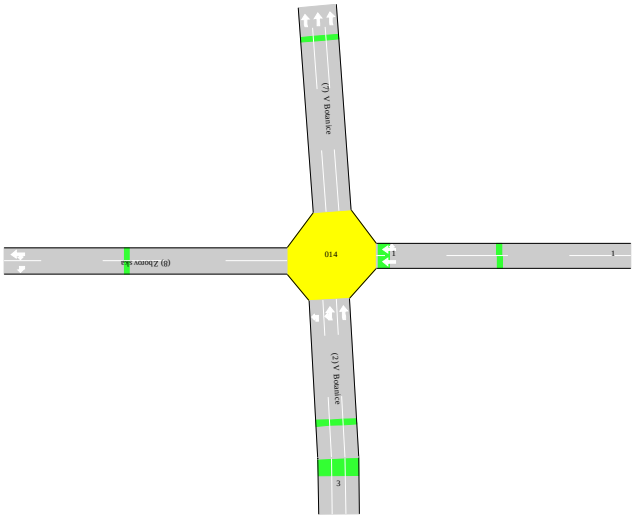
Nákres této křižovatky je uveden na **obr. 8-2**. Je patrné, že v ulici *Kořenského* není zastoupen žádný senzor. V tomto místě tedy není možné napojit na ulici *Kořenského* vstup vozidel, jenž by byl založen na reálných datech a ani porovnat skutečný výstup vozidel z této ulice s výsledky simulace. Proto byla křižovatka ze simulace zcela vyřazena.



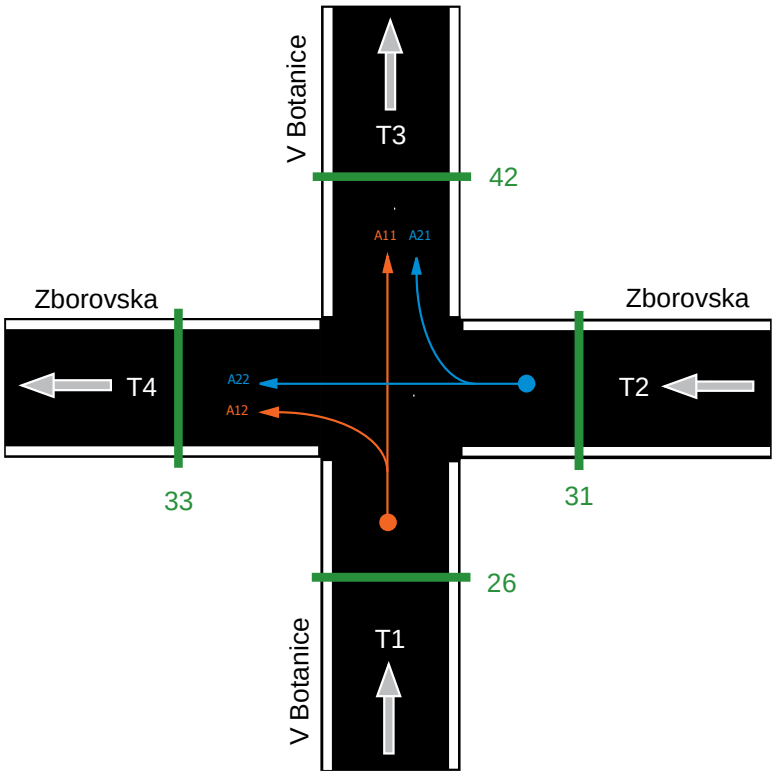
obr. 8-2: křižovatka S01

8.3. Křižovatka S02 (*Zborovská – V Botanice*)

Nákres křižovatky ulic *Zborovská* a *V Botanice* je uveden na **obr. 8-3**. Zjednodušené schéma, uvažující vždy pouze jeden dopravní pruh je pak uvedeno na **obr. 8-4**.



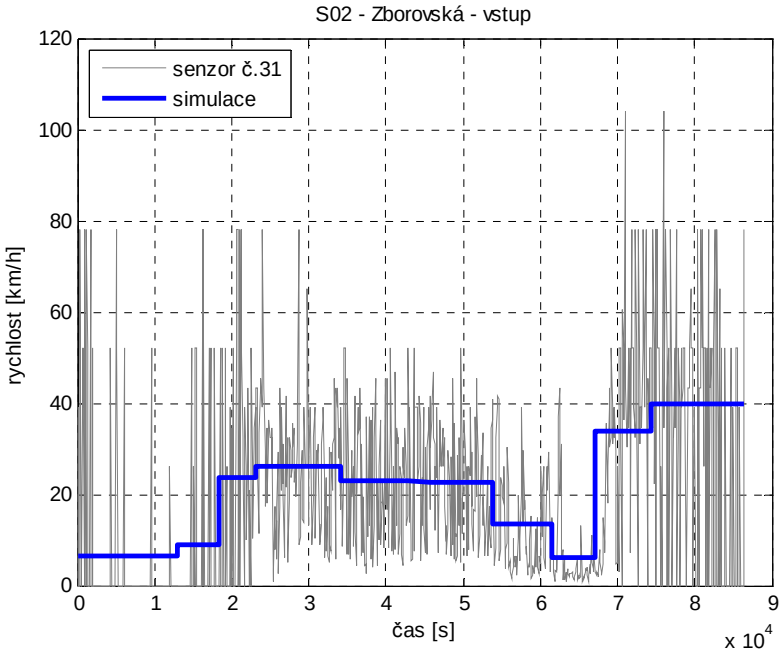
obr. 8-3: křižovatka S02



obr. 8-4: schéma křižovatky S02

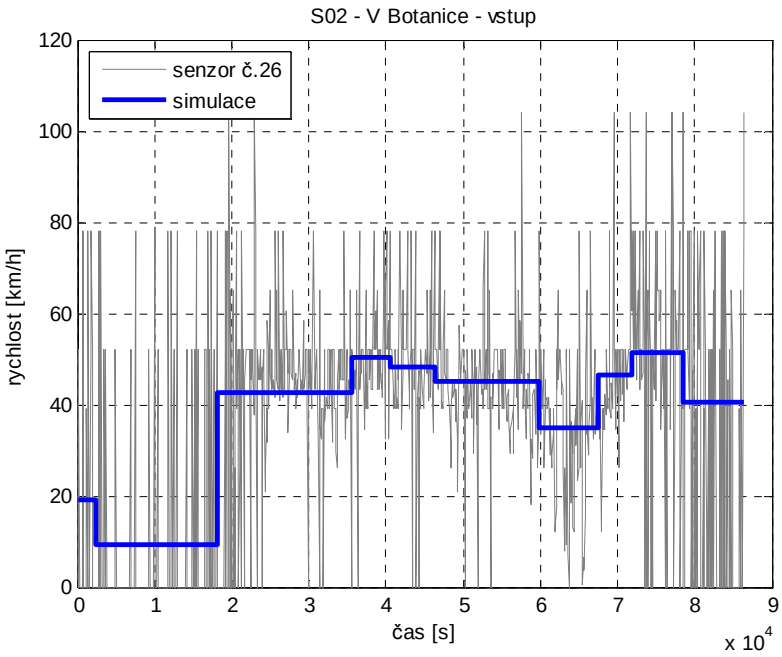
Zelenými pruhy je vždy označen senzor, použitý pro simulaci. Šipky značí směr pohybu vozidel a identifikátor přechodu pak reprezentuje daný vstupní nebo výstupní přechod modelu křižovatky. Odbočovací poměry z ulice *Zborovská* jsou 0.79 (A22) do ulice *Zborovská* a 0.21 (A21) do ulice *V Botanice*.

Z ulice *V Botanice* pak 0.23 (A12) do ulice *Zborovská* a 0.77 (A11) do ulice *V Botanice*. Na základě těchto odbočovacích poměrů byl tedy parametrizován model křižovatky dle kapitoly **7.4**.



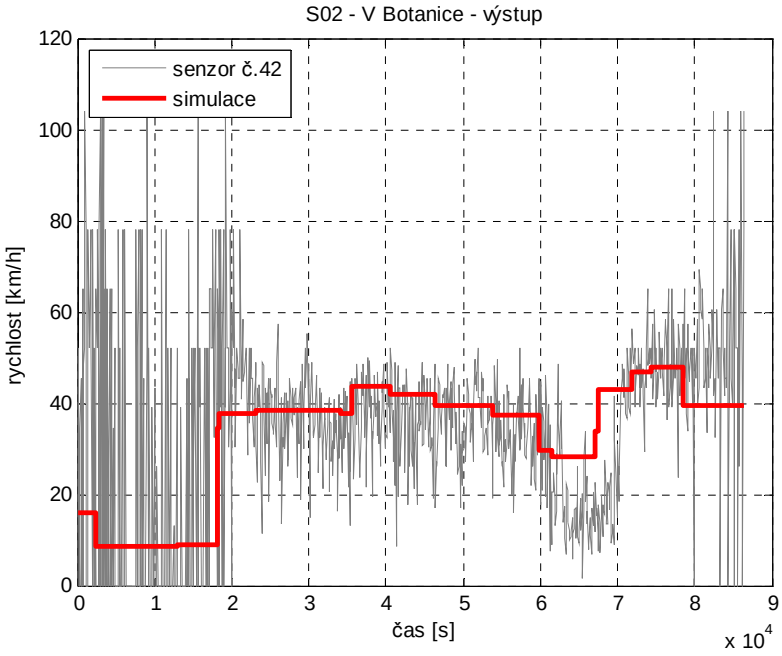
obr. 8-5: vstup vozidel z ulice *Zborovská*

Na **obr. 8-5** je uveden skutečný průběh signálu ze senzoru č.31 a jeho aproximace, použitá pro řízení maximální rychlosti přechodu v modelu vstupu vozidel z ulice *Zborovská*. Vstup vozidel z ulice *V Botanice* je spolu s jeho aproximací použitou pro řízení uveden na **obr. 8-6**.

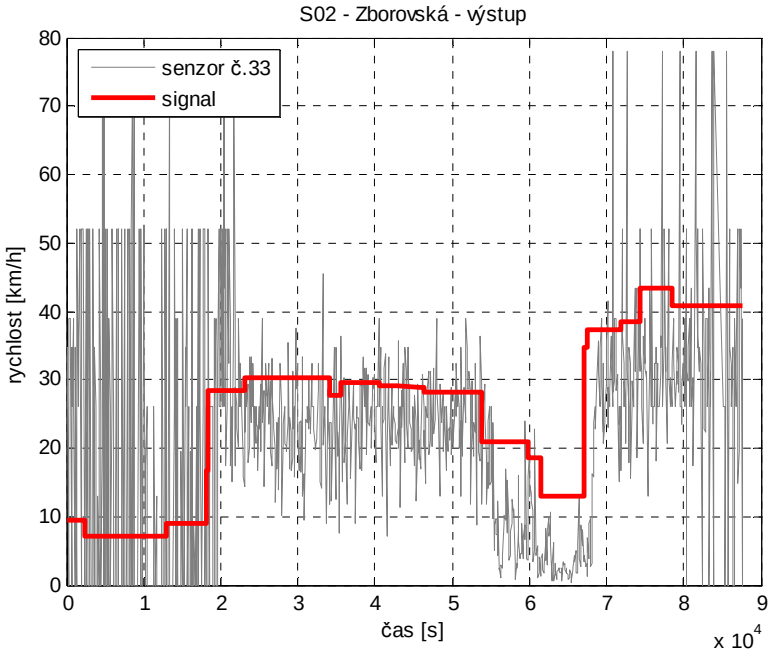


obr. 8-6: vstup vozidel z ulice *V Botanice*

Na **obr. 8-7** je pak zobrazeno srovnání skutečného výstupu vozidel do ulice *V Botanice* a výsledku simulace. Stejné porovnání je pak pro výstup do ulice *Zborovská* uvedeno na **obr. 8-8**.



obr. 8-7: výstup do ulice *V Botanice*

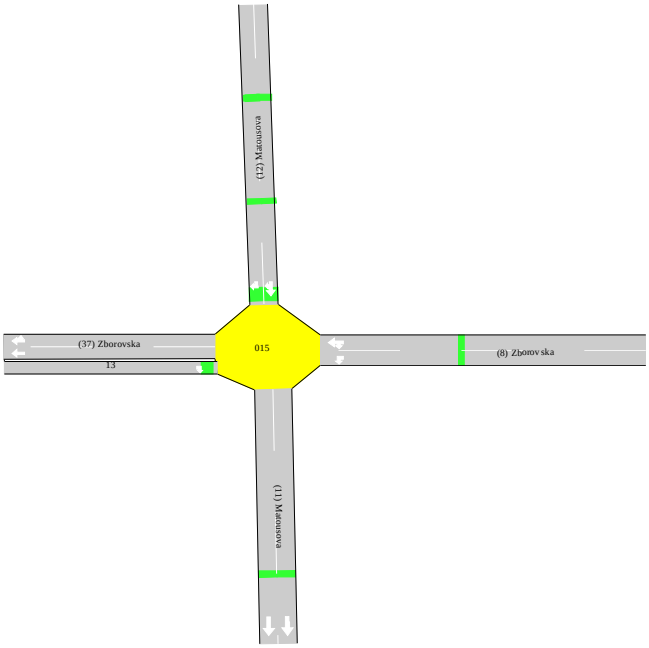


obr. 8-8: výstup do ulice *Zborovská*

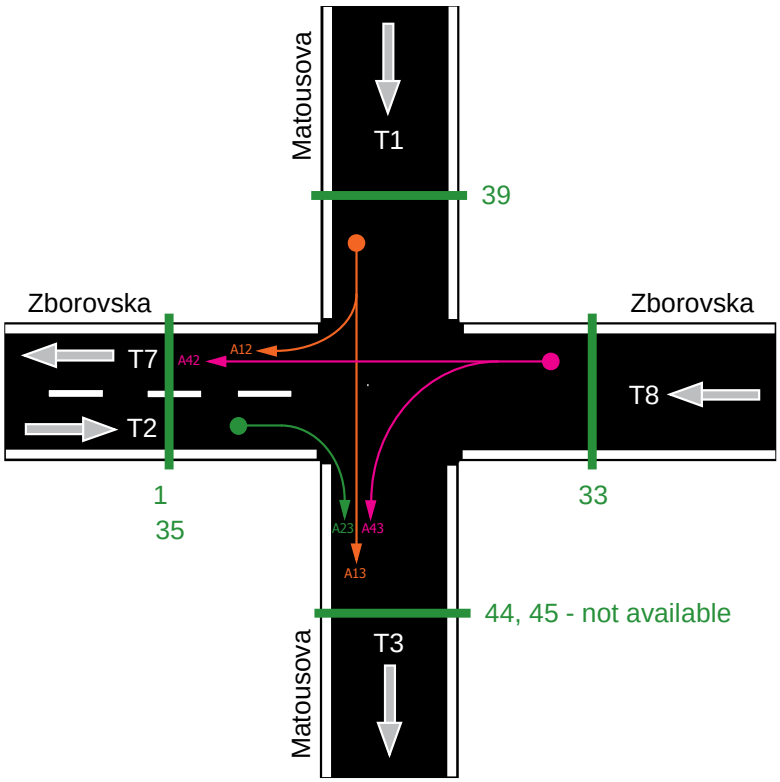
Je patrné že oba výstupní průběhy, vypočítané simulací, věrně aproximují skutečné průběhy ze senzorů v této křižovatce. Výstupní průběh na **obr. 8-8** zároveň slouží jako vstupní průběh pro křižovatku S03. Úsek ulice *Zborovská*, mezi křižovatkou S02 a S03, měří *80m*. Model této ulice má tedy dle (8.2) kapacitu 16 tokenů.

8.4. Křižovatka S03 (*Zborovská – Matoušova*)

Nákres křižovatky ulic *Zborovská* a *V Botanice* je uveden na **obr. 8-9**. Zjednodušené schéma, uvažující vždy pouze jeden dopravní pruh, je pak uvedeno na **obr. 8-10**.

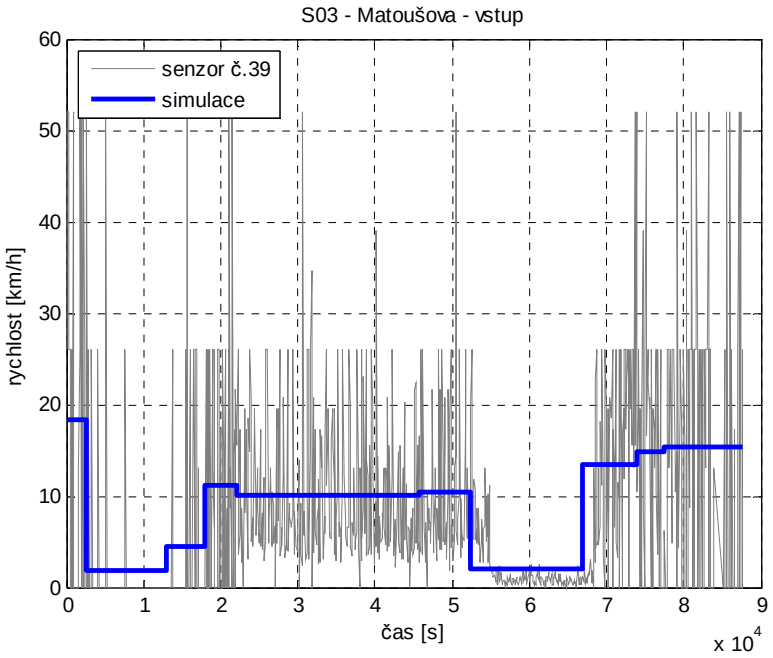


obr. 8-9: křižovatka S03



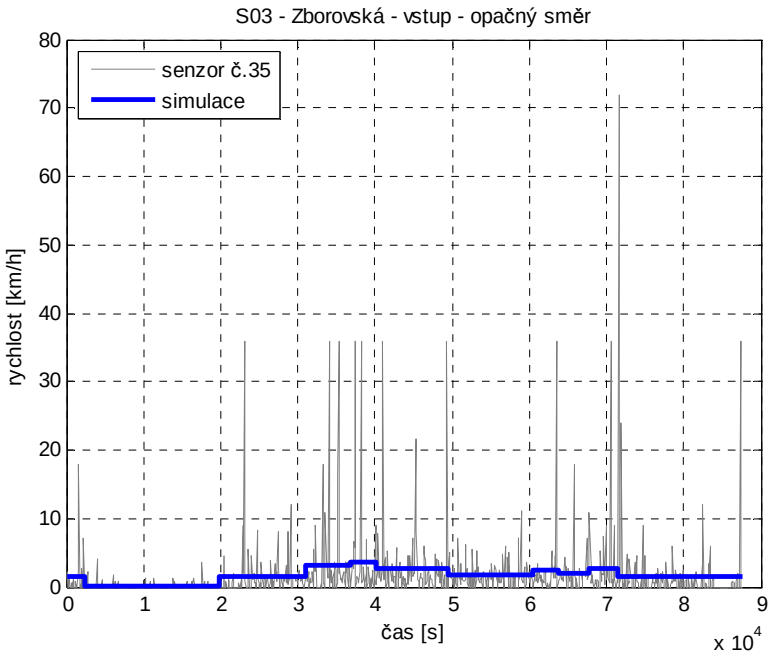
obr. 8-10: schéma křižovatky S03

Vstupní signál z ulice *Zborovská* tvoří výstupní signál z křižovatky S02 a je tedy zobrazen na **obr. 8-8**. Vstupní signál z ulice *Matoušova* je pak uveden na **obr. 8-11**. Odbočovací poměry z ulice *Zborovská* jsou 0.76 (A42) do ulice *Zborovská* a 0.24 (A43) do ulice *V Botanice*.



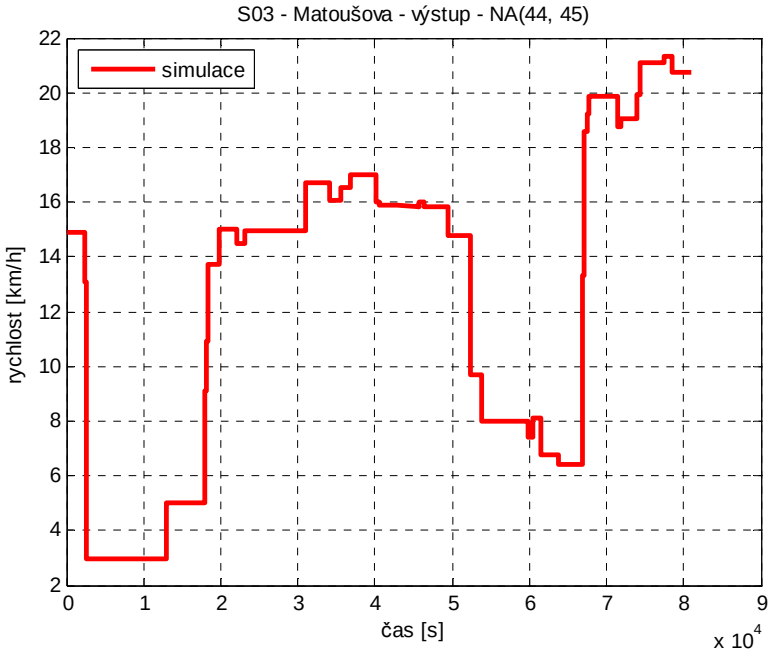
obr. 8-11: vstup z ulice *Matoušouva*

Z ulice *Matoušova* pak 0.39 (A12) do ulice *Zborovská* a 0.69 (A11) do ulice *Matoušova*. Vstupní signál z ulice *Zborovská* v protisměru je uveden na **obr. 8-12**.



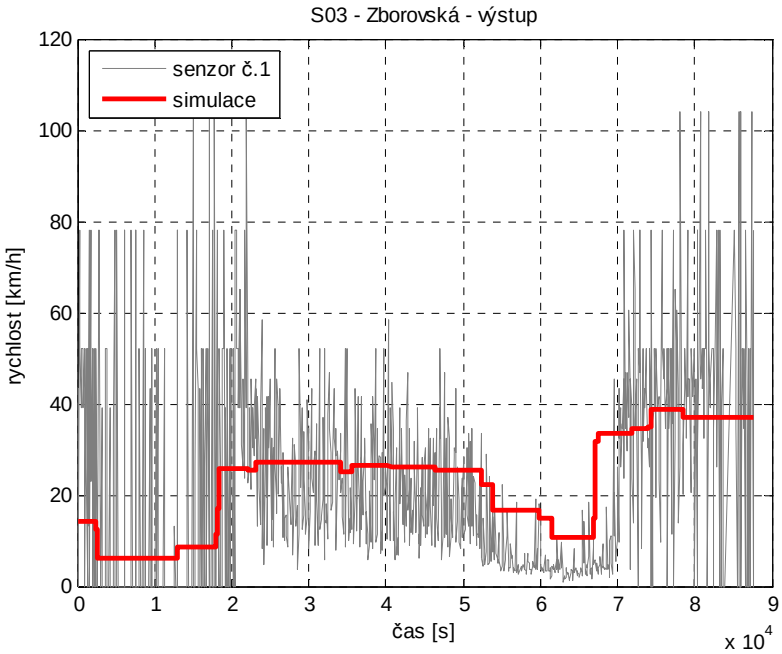
obr. 8-12: vstup z ulice *Zborovská* v protisměru

Výstup do ulice *Matoušova* je uveden na **obr. 8-13**. Reálná data senzorů z obou dopravních pruhů této ulice však nejsou k dispozici a tento graf má tedy pouze informativní charakter.



obr. 8-13: výstup do ulice *Matoušova*

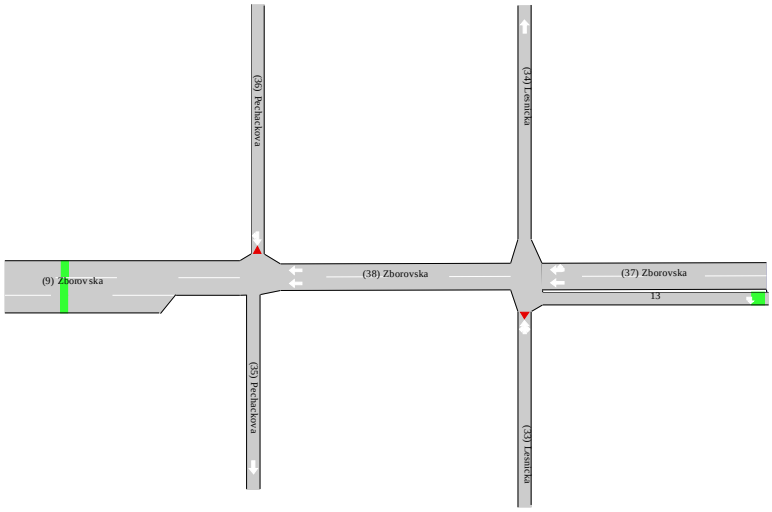
Výstup do ulice *Zborovská*, který zároveň slouží i jako vstupní signál pro křižovatku S06 je uveden na **obr. 8-14**. Je patrné, že výsledky simulace opět korelují s reálným signálem ze senzoru č.1.



obr. 8-14: výstup z ulice *Zborovská*

8.5. Křižovatky S04 (*Zborovská – Lesnická*) a S05 (*Zborovská – Plecháčkova*)

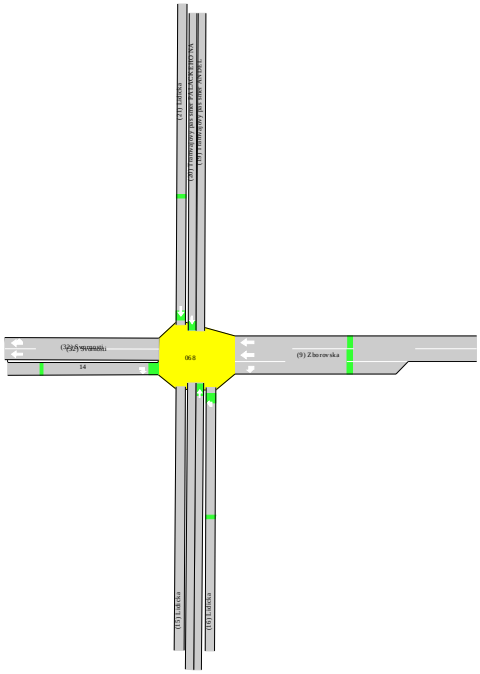
Křižovatky ulic *Zborovská – Lesnická* a *Zborovská – Plecháčkova*, jak je patrné z **obr. 8-15**, nedisponují žádnými senzory. V tomto místě tedy není tedy možné do simulace přidat relevantní vstupní data, ani provést srovnání výstupů se simulací. Vzhledem k faktu, že obě křižovatky by poskytly vždy jeden vstup a jeden výstup v simulované oblasti, nezbývá než předpokládat, že množství vozidel vstupující ulicemi *Lesnická* a *Plecháčkova* do simulované oblasti by zhruba odpovídalo množství vozidel, těmito ulicemi vystupujícími. Ze simulace byly tedy tyto křižovatky vyřazeny a kapacita modelu ulice, spojující křižovatky S03 a S06 byla uvažována jako součet kapacit všech mezilehlých ulic, tedy 36.2 tokenů (*181m*). Lze však předpokládat, že toto opatření vnese do dalších výsledků simulace chybu.



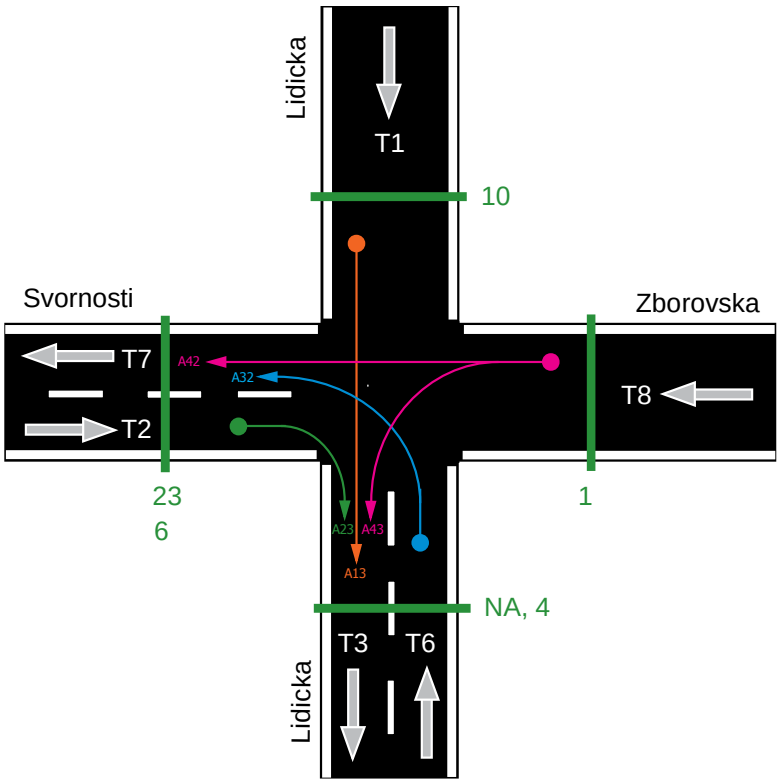
obr. 8-15: křižovatky ulic *Zborovská – Lesnická* a *Zborovská - Plecháčkova*

8.6. Křižovatka S06 (*Zborovská – Svornosti – Lidická*)

Nákres křižovatky ulic *Zborovská*, *Svornosti* a *Lidická* je uveden na **obr. 8-16**. Dva střední dopravní pruhy ulice *Lidická* reprezentují tramvajový pás, který byl zanedbán. Zjednodušené schéma je pak na **obr. 8-17**.

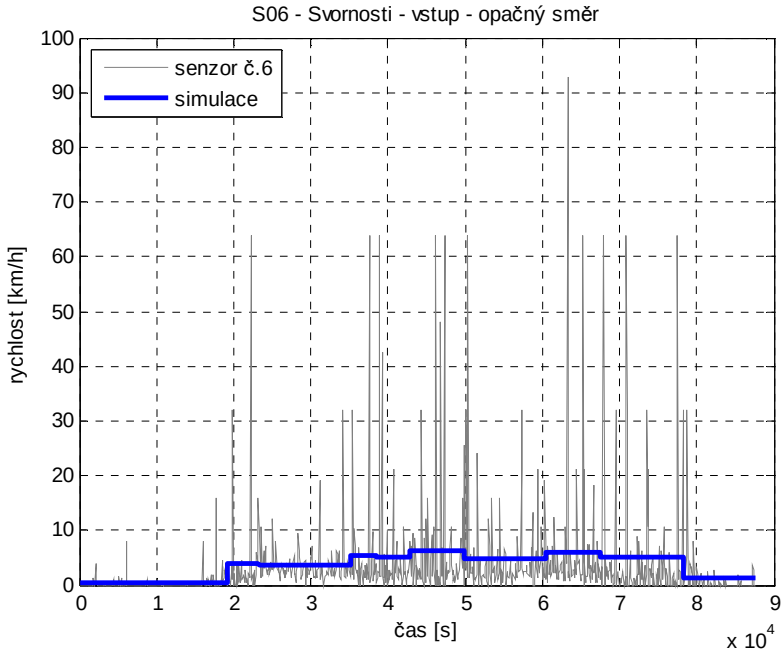


obr. 8-16: křižovatka S06

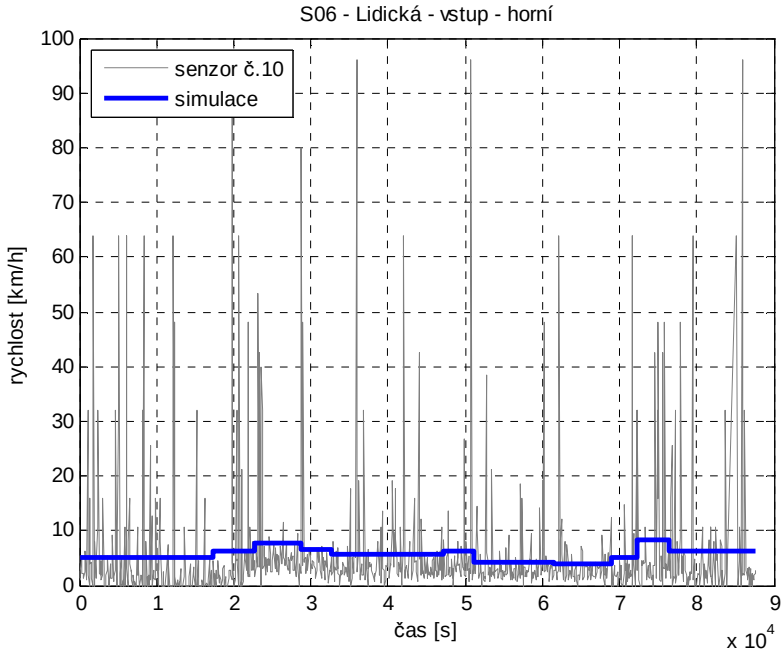


obr. 8-17: schéma křižovatky S06

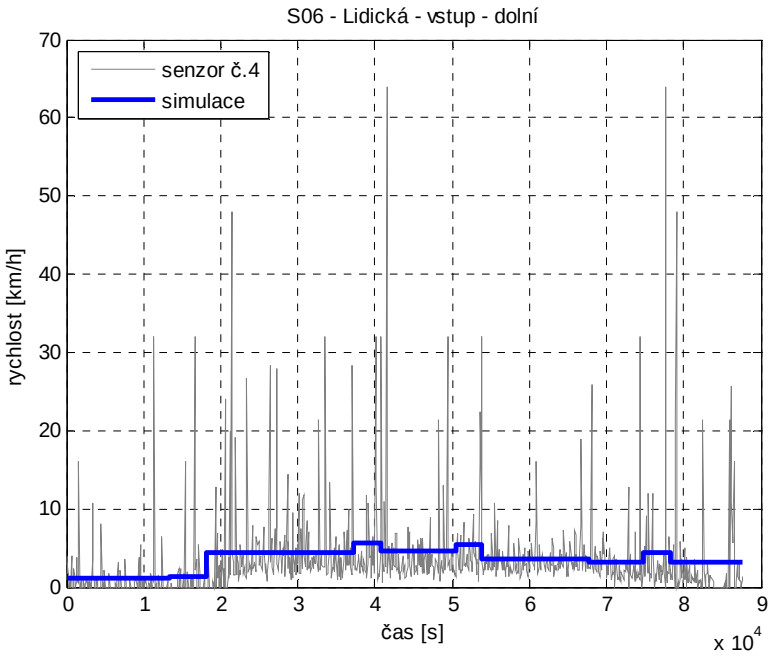
Vstupní signál z ulice *Zborovská* tvoří výstupní signál z křižovatky S03 a je tedy zobrazen na **obr. 8-14**. Další tři vstupy z ulic *Svornosti*, *Lidická* a *Lidická* (protisměr) jsou uvedeny na **obr. 8-18**, **obr. 8-19** a **obr. 8-20**. Odbočovací poměry z ulice *Zborovská* jsou 0.75 (A42) do ulice *Svornosti* a 0.25 (A43) do ulice *Lidická*. Ostatní poměry (A32, A23, A13) jsou rovny jedné.



obr. 8-18: vstup z ulice *Svornosti*

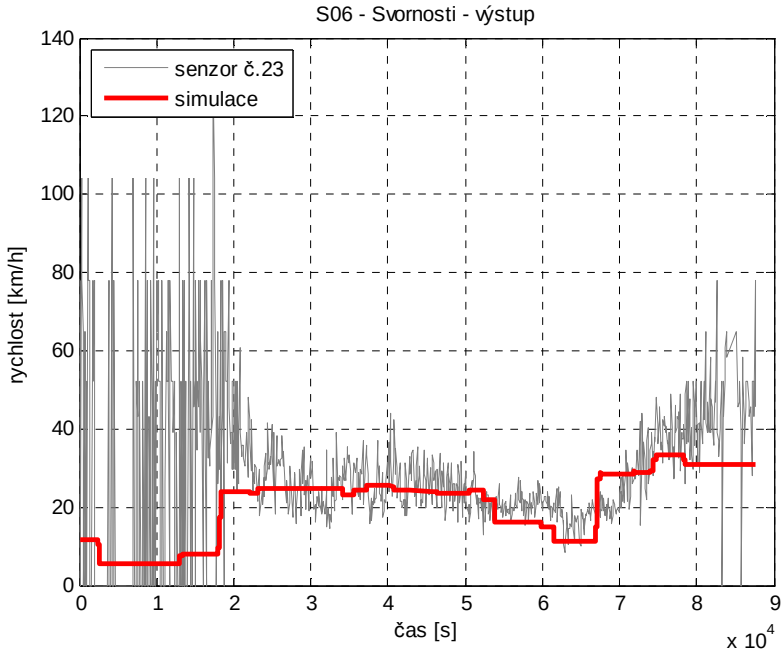


obr. 8-19: vstup z ulice *Lidická*



obr. 8-20: vstup z ulice *Lidická* (protisměr)

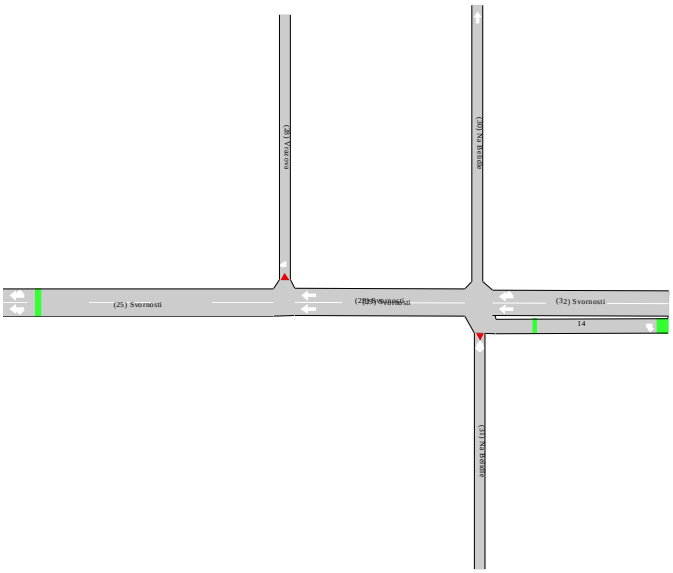
Výstup do ulice *Lidická* nebudeme uvádět, neboť nejsou dostupná data, se kterými by bylo možné výsledky simulace porovnat a měl by tak malou vypovídací hodnotu. Výstup do ulice *Svornosti* je zobrazen na **obr. 8-21**. Přestože je stále patrná korelace mezi reálnými daty a výsledky simulace, dochází zde oproti předchozím výsledkům k mírným odchylkám, zejména od času 7.10^4 *s* dále. Tuto skutečnost lze právě připsat vynechání křižovatek S04 a S05, po kterém simulace pracuje s neúplnými vstupními daty.



obr. 8-21: výstup do ulice *Svornosti*

8.7. Křižovatky S07 (*Svornosti – Na Bělidle*) a S08 (*Svornosti – Vrázova*)

V případě křižovatek ulic *Svornosti − Na Bělidle* a *Svornosti − Vrázova* se jedná o stejnou situaci jako u křižovatek S04 a S05. Pro tyto křižovatky nejsou dostupná reálná data, dle kterých by bylo možné do simulace zavést vstupní průběhy nebo realizovat porovnání výstupních průběhů a proto je nutné tyto křižovatky z oblasti vyřadit. Křižovatky S06 a S09 jsou tedy spojeny modelem ulice o jednom dopravním pruhu a kapacitě vycházející z délky všech mezilehlých ulic, tedy *171m*. Situace je uvedena na **obr. 8-22**.

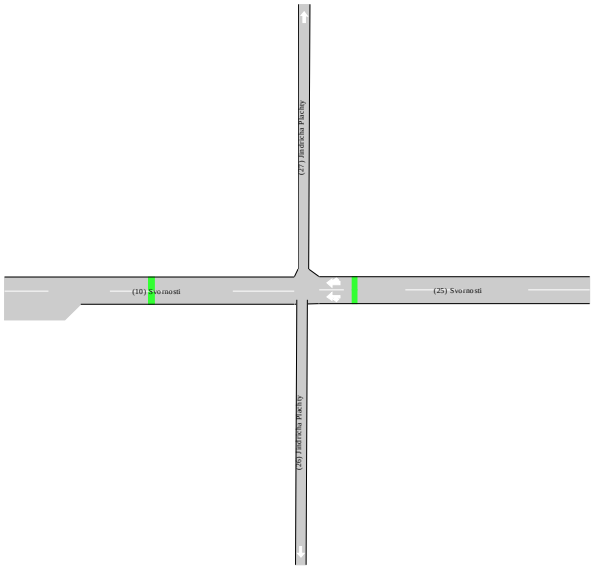


obr. 8-22: křižovatky S07 a S08

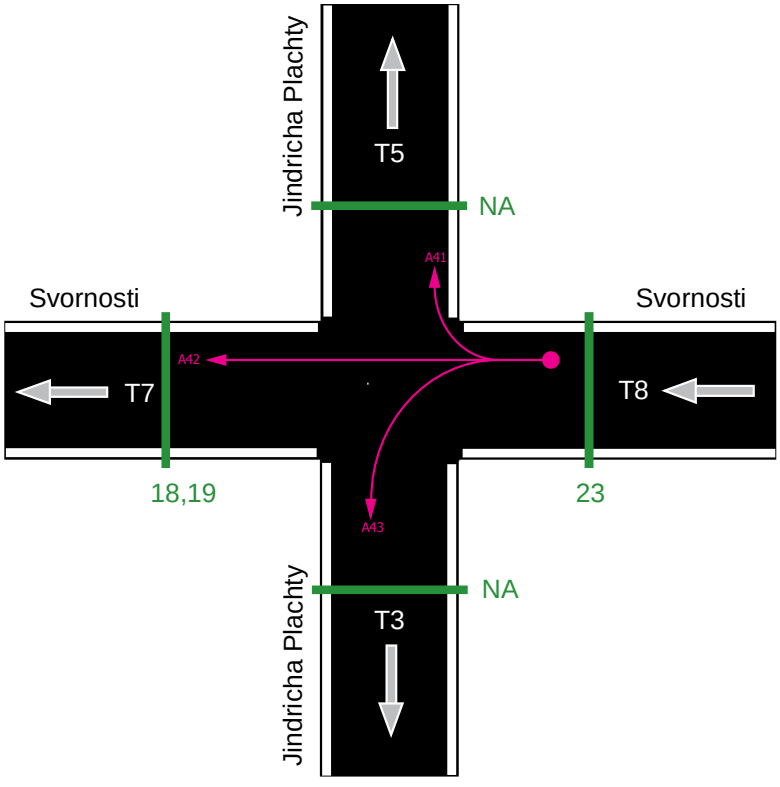
V případě ulice *Na Bělidle* jsme již použili alespoň část celkového vstupu z této ulice pro křižovatku S06, v podobě průběhu ze senzoru č.6 (viz. **obr. 8-18**). Dále lze předpokládat, že zbytek vstupu zhruba odpovídá i výstupu vozidel touto ulicí. V případě ulice *Vrázova* však již tento předpoklad nelze použít, jelikož je tato ulice pouze vstupní. Je tedy zřejmé, že v tomto místě je do výsledků simulace zanesena další chyba.

8.8. Křižovatka S09 (*Svornosti – Jindřicha Plachty*)

Nákres křižovatky ulic *Svornosti* a *Jindřicha Plachty* je uveden na **obr. 8-23**. Zjednodušené schéma pak na **obr. 8-24**. Jedná se o jedinou neřízenou křižovatku v celé zkoumané oblasti, jenž byla v simulaci uvažována. Ulice *Jindřicha Plachty* opět nedisponuje žádnými senzory, nicméně v tomto případě je zmíněná ulice pouze výstupní. Na základě známých odbočovacích poměrů ji tedy lze do simulace zařadit, přestože pro výstupy z této ulice nebude možné uvést srovnání simulace s reálnými daty.

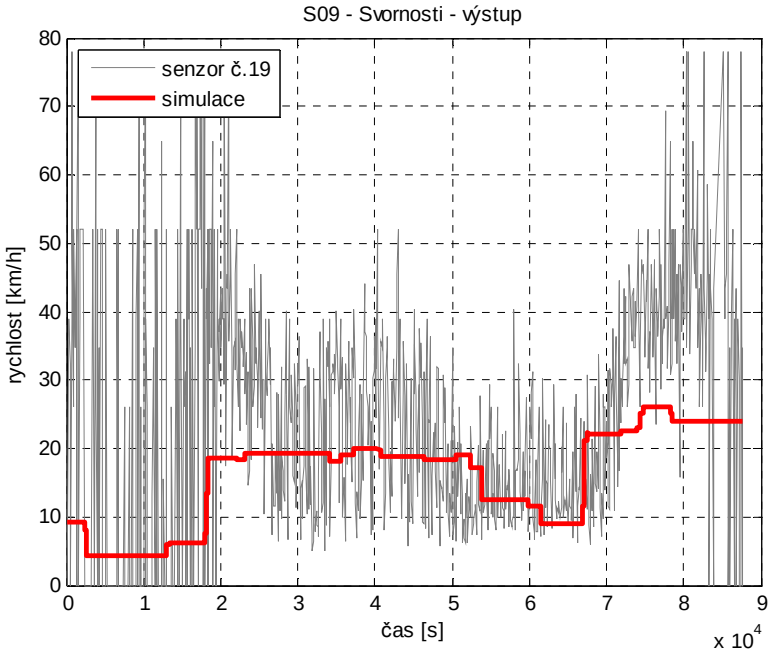


obr. 8-23: křižovatka S09



obr. 8-24: schéma křižovatky S09

Odbočovací poměry z ulice *Svornosti* do ulice *Svornosti* jsou 0.78 (A42), do ulice *Jindřicha Plachty* (nahoru) 0.09 (A41) a 0.13 (A43) do ulice *Jindřicha Plachty* (dolů). Na **obr. 8-25** je uvedeno srovnání reálných dat výstupu do ulice *Svornosti* s výsledky simulace. Souvislost mezi reálnými daty je stále uspokojující, avšak jsou patrné výraznější odchylky v důsledku chybějících dat z křižovatek S04, S05, S07 a S08, zejména pak v intervalech $1.10^4 – 3.10^4$ *s* a od 7.10^4 *s* dále.



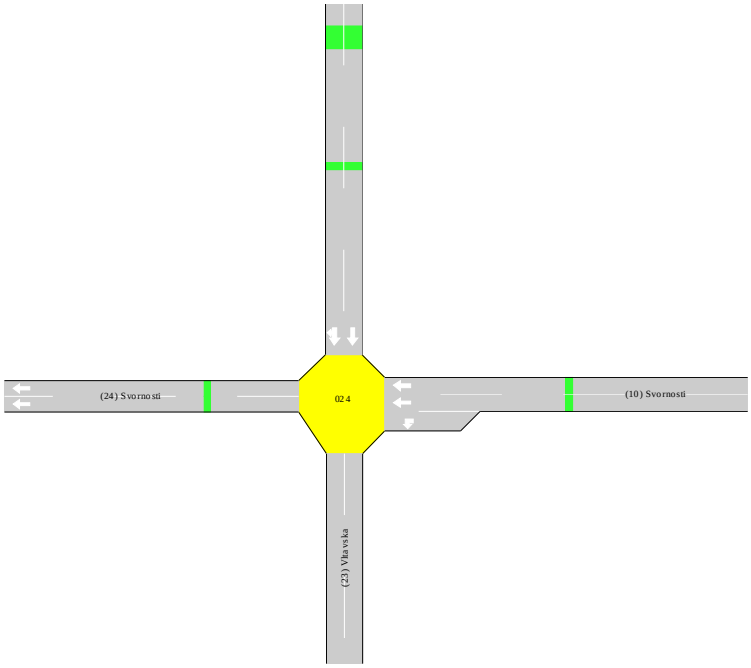
obr. 8-25: výstup do ulice *Svornosti*

Výstupy do ulice *Jindřicha Plachty* nebudeme uvádět, neboť je není možné porovnat s reálnými daty a měly by tak malou vypovídací hodnotu.

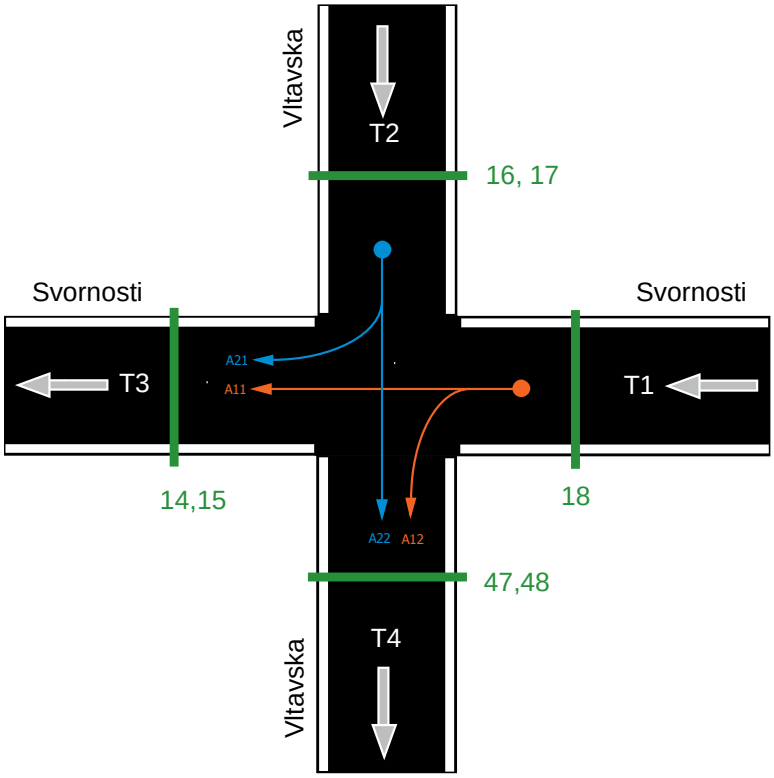
8.9. Křižovatka S10 (*Svornosti – Vltavská*)

Křižovatka ulic *Svornosti* − *Vltavská* je poslední křižovatkou v simulaci. Její nákres je uveden na **obr. 8-26**, zjednodušený model pak na **obr. 8-27**. Odbočovací poměry z ulice *Svornosti* jsou 0.88 (A11) do ulice *Svornosti*, 0.26 (A12) do ulice *Vltavská* a z ulice *Vltavská* pak 0.19 (A21) do ulice *Svornosti* a 0.81 (A22) do ulice *Vltavská*.

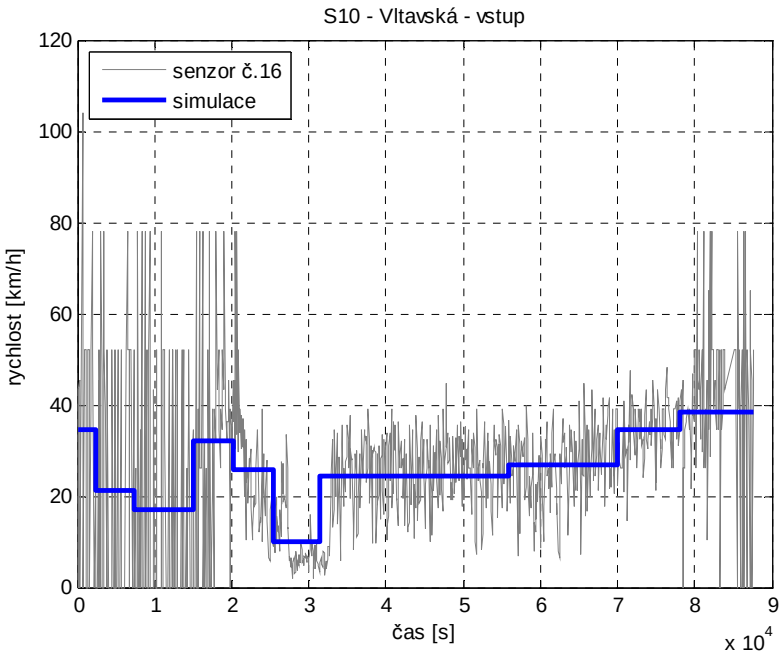
Vstup z ulice *Svornosti* je totožný s výstupem uvedeném na **obr. 8-25**. Vstup z ulice *Vltavská* je znázorněn na **obr. 8.28**. Příslušné výstupní průběhy z ulic *Svornosti* resp. *Vltavská* jsou uvedeny na **obr. 8.29** resp. **obr. 8.30**. Reálná data ze senzorů v těchto ulicích však nejsou dostupná a budeme se tak muset smířit pouze s informativními výsledky simulace.



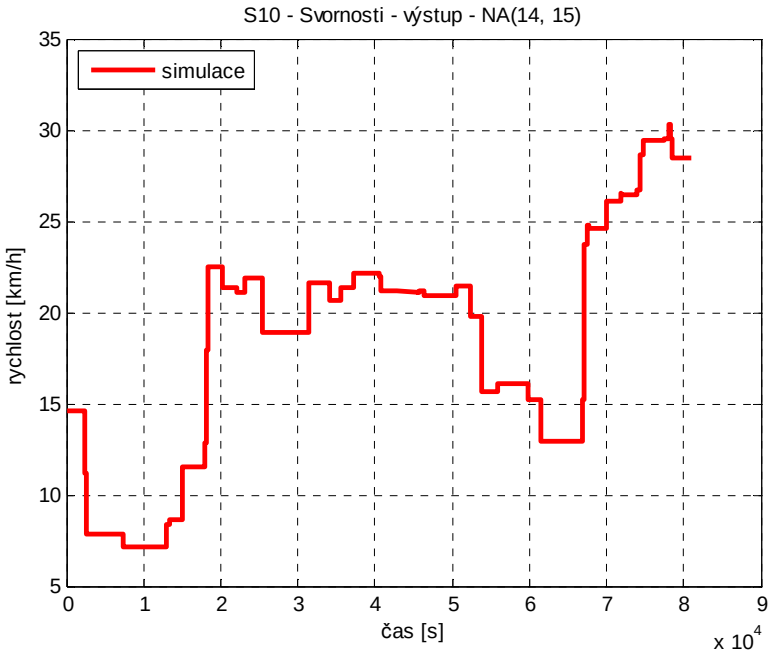
obr. 8-26: křižovatka S10



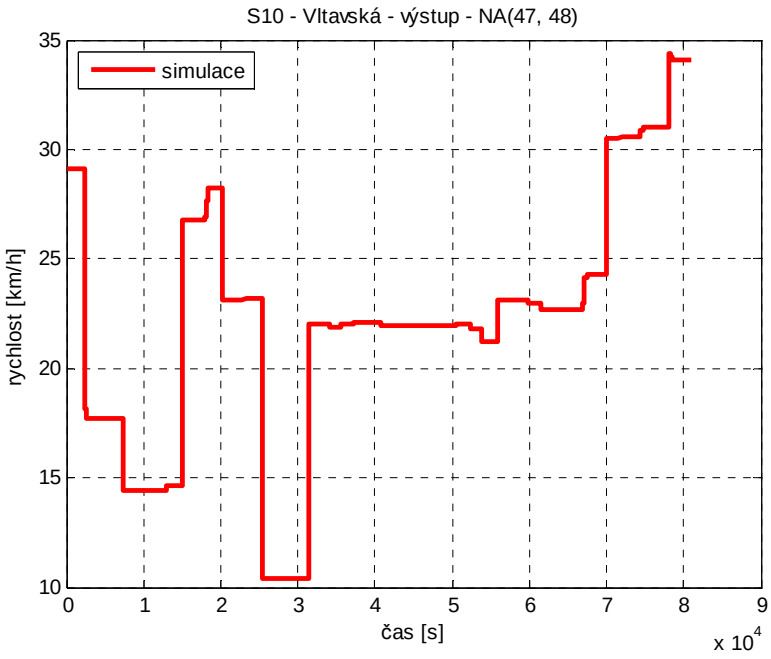
obr. 8-27: schéma křižovatky S10



obr. 8-28: vstup z ulice *Vltavská*



obr. 8-29: výstup do ulice *Svornosti*



obr. 8-30: výstup do ulice *Vltavská*

8.10. Shrnutí výsledků simulace

Z předchozích kapitol je zřejmé, že výsledky simulace jsou velmi blízkou aproximací reálných dat. Použitý přístup simulace transportní sítě pomocí modelů, založených na spojitých Petriho sítích s proměnnou maximální rychlostí lze tedy použít pro statistickou analýzu zkoumané oblasti. Mírné odchylky výsledků simulace od reálných dat, které se vyskytují ve výstupních průbězích od křižovatky S06 dále, lze připsat chybějícím vstupním datům z křižovatek S04, S05, S07 a S08. Dále tedy simulace probíhala na základě neúplných dat.

Výsledný model celé zkoumané oblasti obsahoval padesát devět míst, třicet jedna přechodů a sto dvacet dva hran spojité Petriho sítě. Řízení maximální rychlosti na základě datových souborů podléhalo osm přechodů. Výpočet byl dokončen v sedmdesáti pěti iteracích. Celkový výpočetní čas pak na průměrném PC[4] za použití aplikace *GPNS_console* dosahoval pouhých osmi sekund.

[4] Intel Core2Duo T5500, 1.5GB RAM, Windows XP Professional - SP2

9. Závěr

Tato práce popisuje návrh a implementaci obecného simulátoru transportních sítí, založeného na formalizmu diskrétních a spojitých Petriho sítí. Simulátor je realizován v podobě knihovny tříd a metod objektově orientovaného programovacího jazyku *C++* a poskytuje tak uživateli oporu formalizmu Petriho sítí v tomto prostředí.

Na základě teoretického rozboru a seznámení se s danou problematikou byla implementována knihovna *GPNS*. Ta poskytuje uživateli jazyka *C++* široké spektrum možností pro práci s Petriho sítěmi. Od manipulace se základními elementy, kterými jsou místa, hrany a přechody, přes metody věnované Petriho sítím jako celku až po specializované třídy a metody určené pro výpočet vývoje značení diskrétních i spojitých Petriho sítí. Mezi pokročilé vlastnosti knihovny pak patří zejména možnost řešit efektivní konflikt ve spojité Petriho síti proporcionální metodou, možnost simulací nelineárního chování Petriho sítí a automatické spojování obecného počtu sítí. Při realizaci projektu byl kladen velký důraz na modularitu knihovny, která by v budoucnu zajistila její snadnou rozšiřitelnost a poskytla maximální možné využití pro další, navazující projekty.

Pro účely vlastních simulací pak byla implementována konzolová aplikace, založená na knihovně *GPNS*. Ta mimo jiné umožňuje spolupráci s ostatními nástroji věnovaným Petriho sítím skrze rozšířený *PNML* formát, dávkové spojování modelů realizovaných Petriho sítěmi a výpočty značení Petriho sítí.

Dále byla sestavena knihovna modelů některých dopravních prvků, na jejichž základě bylo možné provádět simulace z oblasti městské dopravy. Vedle modelu vstupu vozidel, výstupu vozidel a ulice o obecném počtu dopravních pruhů byl také představen postup sestavení modelu obecné křižovatky a jeho parametrizace. Zmíněné modely vycházejí s formalizmu spojitých Petriho sítí a k řešení efektivního konfliktu používají proporcionální metodu.

Posledním bodem práce bylo provedení příkladové simulace na reálných datech z oblasti městské dopravy, konkrétně části regionu Praha – Smíchov. Výsledky této simulace, které věrně kopírují reálná data z dané oblasti, vypovídají o vhodnosti navržených modelů a možnostech implementovaného simulátoru. Použitý přístup je pak vhodný zejména pro statistické analýzy vysoce zatížených oblastí.

Práce otevírá prostor pro navazující projekty ať již v oblasti rozšiřování implementované knihovny *GPNS* o metody pro analýzu vlastností Petriho sítí, podporu hybridních Petriho sítí, implementaci koncové aplikace využívající široké spektrum možností této knihovny nebo experimentování s definovanými modely transportních sítí v podobě parametrizace maximálních rychlostí přechodů atp. V neposlední řadě by bylo možné provést experimenty i z jiných oblastí transportních sítí jako například sítě Ethernet nebo průmyslových sítí.

10. Seznam použitých zdrojů

[Aimsun]
TTS − Transport Simulation Systems. *Aimsun NG* [online]. Pg. de Gràcia 12, 08007 Barcelona. 1998-2006. [cit. 2007-05-07]. <http://www.aimsun.com/>.

[Alla]
ALLA, H. – DAVID, R. *A modelling and analysis for discrete events systéme: continuous Petri net*. Laboratorie d'Automatique de Grenoble. Recieved April 14, 1993. Revised December 15, 1997. France. INPG-UJF-CNRS.

[Boost]
RIVIERA, R. *Boost C++ Libraries* [online]. [cit. 2007-04-25]. <http://www.boost.org >.

[David I]
DAVID, R. − ALLA, H. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, November 23, 2004. ISBN 3-540-22480-7.

[David II]
DAVID, R. − ALLA, H. *On Hybrid Petri Nets*. Discrete Event Dynamic Systems: Theory and Applications, 11, 9-40. Kluwer Academic Publishers. Boston, 2001.

[Di Febbraro I]
DI FEBBRARO, A. − GIGLIO, D. − SACCO, N. *Modular Representation of Urban Traffic Systems based on Hybrid Petri Nets*. IEEE Intelligent Systems Conference Proceedings. Oakland (CA) USA, 2001. 0-7803-7194-1/01.

[Di Febbraro II]
DI FEBBRARO, A. − SACONE, S. *Hybrid Modelling of Transportation Systems by means of Petri Nets*. Department of Communications. Computer and System Sciences. University Of Genova. Genova, 1998. IEEE, 0-7803-4778-1/98.

[Doxygen]
DIMITRI van H. *Doxigen manual* [online]. [cit. 2007-04-25].
<http://www.stack.nl/~dimitri/doxygen/manual.html>.

[Eckel]
ECKEL, B. *Myslíme v jazyku C++: knihovna programátora*. Praha: Grada, 2000. ISBN 80-247-9009-2.

[EXPAT]
CLARK, J. The *Expat XML parser* [online]. [cit. 2007-04-20].
<http://expat.sourceforge.net/>.

[Gallego]
GALLEGO, J.L. − FARGES, J.L. − HENRY, J.J. *Design by Petri Nets of an intersection signal controller*. Elsevier Science, 1996. Vol. 4, No. 4. PII: S0968-090X(96)00009-5.

[GLPK]
MAKHORIN, A. *GNU Linear Programming Kit* [online]. Department for Applied Informatics. Moscow Aviation Institute. Russia. <http://www.gnu.org/software/glpk/>.

[GPL]
TIEMANN, M. *The GNU General Public License* [online]. Open Source Initiative OSI. California 501. [cit. 2007-04-25]. <http://www.opensource.org/licenses/gpl-license.php>.

[Graphviz]
ELLSON, J. – NORTH, S. *Graphviz - Graph Visualization Software* [online].
[cit. 2007-04-25]. <http://www.graphviz.org/>.

[Halmo]
HALMO, L. *Numerické algoritmy pro polynomiální matice v jazyce C++*. České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra řídicí techniky. Praha, 2004. Diplomová práce.
<http://dce.felk.cvut.cz/dolezilkova/diplomky/2004/dp_2004_halmo_leos/dp_2004_Halmo_Leos_cz.pdf>.

[Hanzálek I]
HANZÁLEK, Z. *Petriho sítě* [online]. České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra řídicí techniky. Praha, 2006. [cit. 2007-04-25].
<http://dce.felk.cvut.cz/hanzalek/prednasky/petri.pdf>.

[Hanzálek II]
HANZÁLEK, Z. *Petriho sítě a čas*. České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra řídicí techniky. Praha, 2006. Studijní materiál předmětu 35DRS.

[Hanzálek III]
HANZÁLEK, Z. *Continuous Petri Nets and Polytopes*. Czech Technical University in Prague. Falculty of Electrical Engineering. Department of Control Engineering. Centre for Applied Cybernetics. Prague, 2003. 0-7803-7952-7/03/2003 IEEE.

[Júlvez]
JÚVEZ, J. − BOEL, R. *Modelling and Controlling Traffic Behaviour with Continuous Petri Nets*. Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza. Zaragoza, 2005.

[Kutil]
KUTIL, M. – HANZÁLEK Z. *Traffic Microregion Model Based on the Continuous Petri Nets with Maximal Speed Proportion Conflict Resolution*. Czech Technical University in Prague. Falculty of Electrical Engineering. Department of Control Engineering. Prague, 2007. Preprint.

[List]
LIST, G.F. *Modeling Traffic Signal Control Using Petri Nets*. IEEE Transactions on Intelligent Transportation Systems, 2004. Vol. 5, No. 3. 1524-9050/04.

[MIT]
TIEMANN, M. *The MIT License* [online]. Open Source Initiative OSI. California 501. [cit. 2007-05-23]. <http://www.opensource.org/licenses/mit-license.php>.

[OMNeT++]
OMNET++ COMMUNITY. *OMNeT++: Discrete Event Simulation System*. [cit. 2007-04-28]. < http://www.omnetpp.org/>.

[Petri]
PETRI, C. A. *Kommunikation mit Automaten*. Bonn: Institut fur Instrumentelle Mathematik. Schiften des IIM Nr.3, 1962.

[PNML]
WEBER, M. *Petri Net Markup Languege* [online]. [cit. 2007-04-15].
<http://www2.informatik.hu-berlin.de/top/pnml/about.html>.

[Relax NG]
CLARK, J. *RELAX NG home page* [online]. [cit. 2007-04-27]. <http://relaxng.org/>.

[VSPNE]
ŠREMER, M. *Editor pro Stochastické Petriho sítě na WWW pomocí Java technologií*. České vysoké učení technické v Praze. Fakulta elektrotechnická. Katedra řídicí techniky. Praha, 2006. Diplomová práce.
<http://dce.felk.cvut.cz/dolezilkova/diplomky/2006/dp_2006_sremer_martin/dp_2006_Sremer_Martin.pdf>.

A. *Příloha – CD*

Součástí této práce je přiložený CD-ROM obsahující zdrojové kódy implementované knihovny *GPNS*, některé knihovny a nástroje použité při její implementaci, technickou dokumentaci zdrojových kódů, podpůrné definice jazyka *PNML* pro grafický editor *VSPNE*, knihovnu modelů základních dopravních prvků, podklady pro simulaci dopravy v regionu Praha – Smíchov a tento dokument v elektronické podobě.

a. Knihovna *GPNS*, koncová aplikace a technická dokumentace

Zdrojové kódy implementované knihovny *GPNS* jsou uvedeny v souboru *gpns/lib/GPNS.cpp*, hlavičkový soubor této knihovny je pak *gpns/lib/GPNS.h*. Zdrojové kódy koncové aplikace *GPNS_console* jsou uvedeny v souboru *gpns/lib/GPNS_console.cpp*. V adresáři *gpns/bin/* se nachází spustitelná verze koncové aplikace včetně knihoven potřebných pro její funkci. Technická dokumentace knihovny *GPNS* je uvedena v adresáři *gpns/doc/* ve formě *HTML* a jako zdrojové soubory pro formát *LaTeX*. Nástroje a knihovny použité při implementaci knihovny *GPNS* se nacházejí v adresáři *gpns/tools*.

b. Knihovna modelů základních dopravních prvků a podklady pro simulaci

Knihovna modelů základních dopravních prvků je uvedena v adresáři *traffic/lib*. Každý model je reprezentován *XML*/*PNML* souborem s komentářem, schématem v několika grafických formátech a obrázkem modelu.

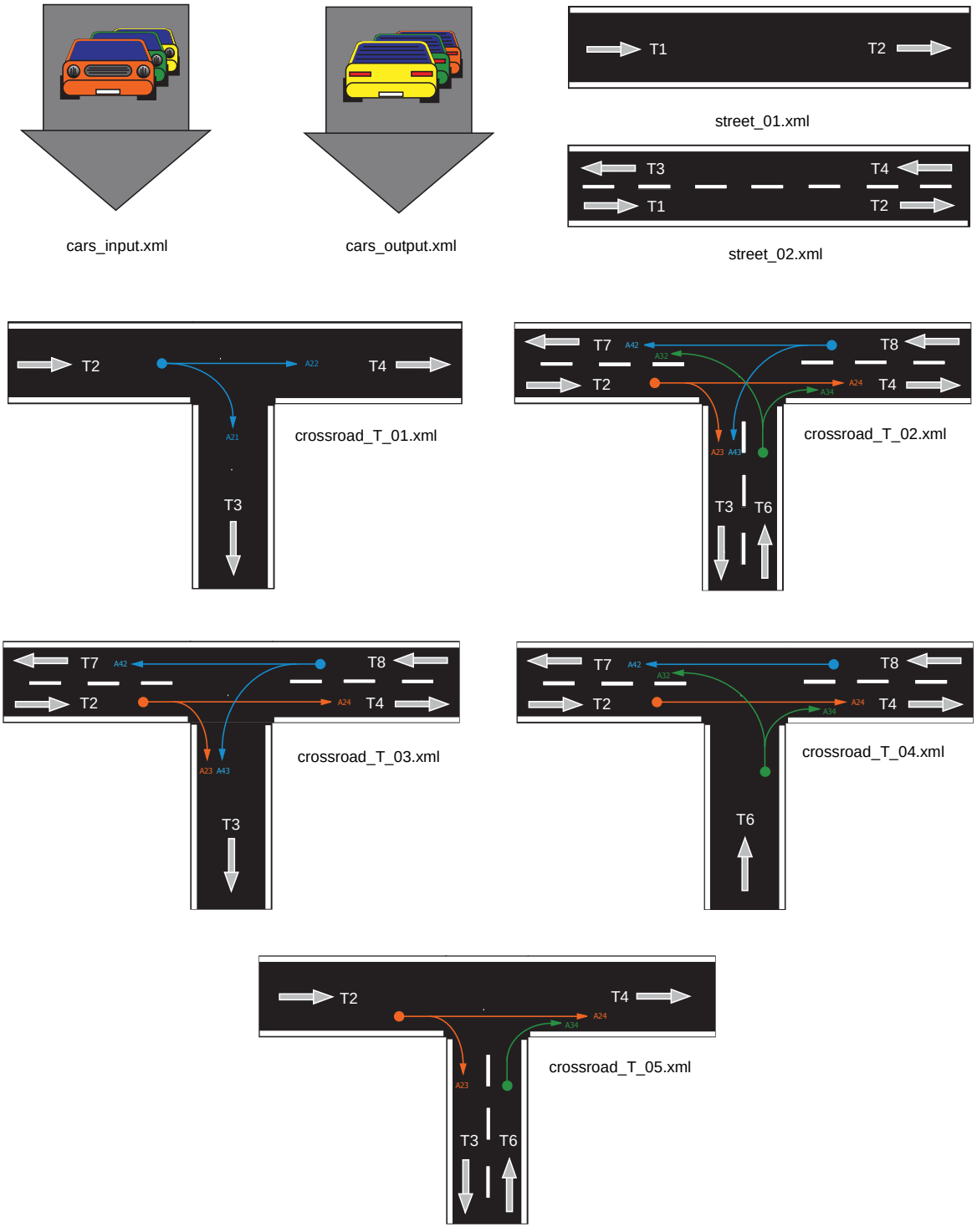
Parametrizované modely pro simulaci oblasti Praha – Smíchov jsou uspořádány ve stejné formě v adresáři *traffic/smichov*. Navíc tento adresář obsahuje konfigurační soubor pro spojení celkového modelu *config_join.txt*, konfigurační soubor pro řízení maximálních rychlostí přechodů tohoto modelu *config_speed.txt* a výstupní soubor simulace *evol.txt*. *MATLAB* script pro grafické znázornění výsledků simulace je pak uložen v souboru *GPNS_evol_data_reader.m*. Data ze senzorů simulované oblasti jsou uvedena v textových souborech v adresáři *traffic/smichov/data/* včetně vypočtených aproximací. Konfigurační a datové soubory slouží jako vstup pro aplikaci *GPNS_console*.

c. *PNML*

Rozšířený *PNML* standard je v podobě podpůrné definice pro obecný grafický editor *VSPNE* uveden v souboru *pnml*/*CCPNTD_sd_01.xml*.

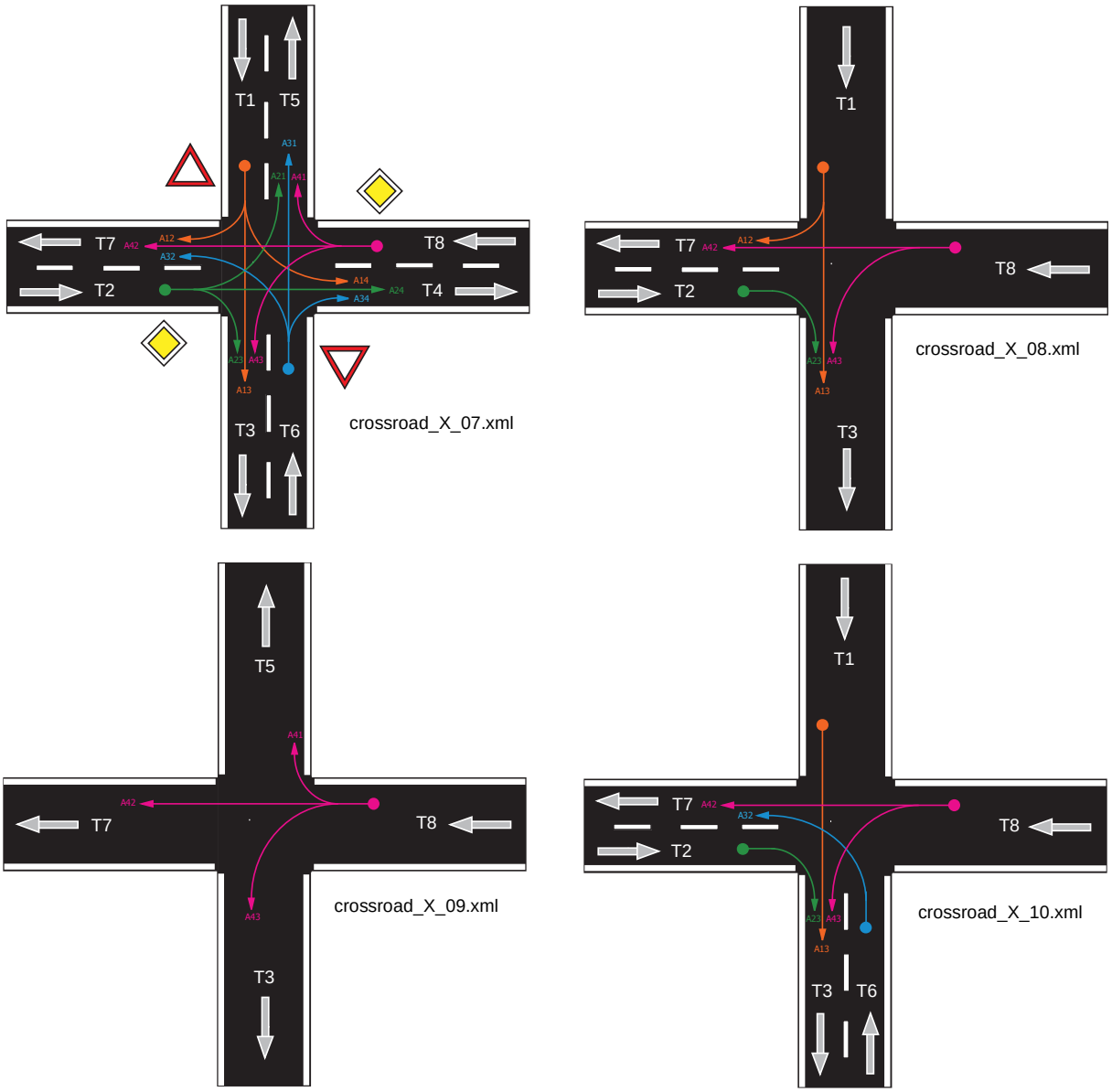
B. Příloha – Knihovna základních dopravních prvků

Na obrázcích **obr. B-1**, **obr. B-2** a **obr. B-3** je uveden celkový náhled na obsah knihovny základních dopravních prvků. Jméno souboru u každého obrázku odkazuje na soubor, ve kterém je dle přílohy ***A*** vždy uveden model prvku ve formalizmu spojitých Petriho sítí jako *PNML* soubor.



obr. B-1: základní dopravní prvky

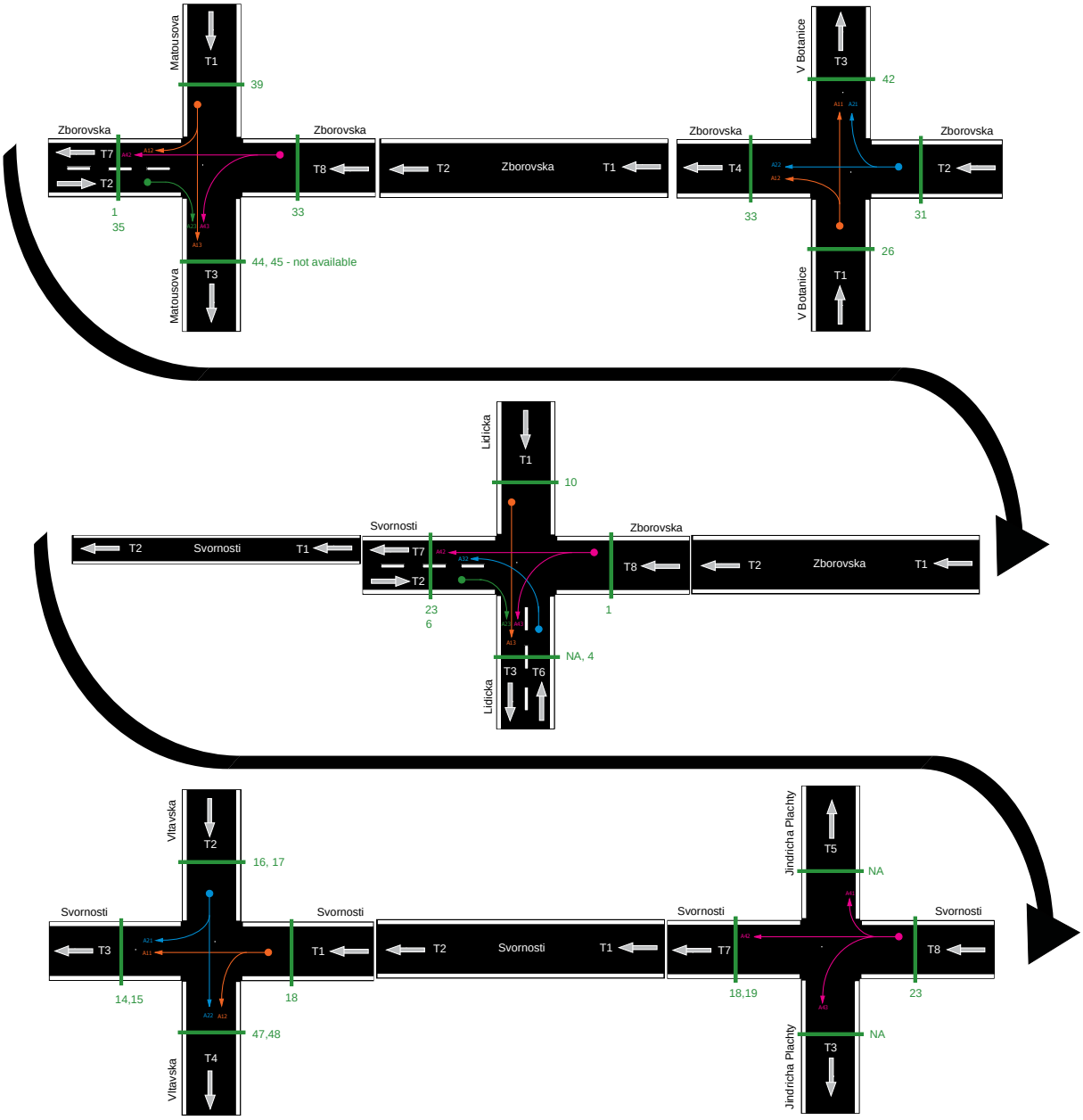
obr. B-2: základní dopravní prvky



obr. B-3: základní dopravní prvky

C. Příloha – Celkové schéma simulované oblasti

Na obrázku **obr. C-1** je uvedeno zjednodušené schéma uvažované oblasti Praha − Smíchov.



obr. C-1: schéma simulované oblasti

D. Příloha – Seznam použitých zkratek

Zkratka	Význam
BLAS	Basic Linear Algebra Subprograms
CCPN	Constant Speed Continuous Petri Net
GLPK	GNU Linear Programming Kit
GNU	GNU's Not Unix (rekurzivní zkratka)
GPL	General Public License
GPNS	General Petri Nets Simulator
HSPN	Hybrid Stochoastic Petri Nets
HTML	HyperText Markup Language
IB-stav	Invariant Behavior state
MIT	Massachusetts Institute of Technology
MSDN AA	Microsoft Development Network Academic Aliance
PC	Personal Computer
PNML	Petri Net Markup Language
VSPNE	Variable Stochastic Petri Net Editor
XML	eXtensible Markup Language

tabulka D-1: seznam použitých zkratek