VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DEMONSTRACE VLASTNOSTÍ ZNAČENÝCH GRAFŮ

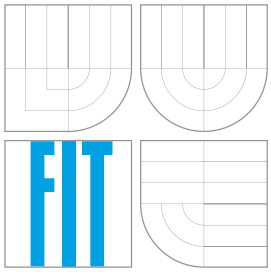
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL HORÁK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DEMONSTRACE VLASTNOSTÍ ZNAČENÝCH GRAFŮ

DEMONSTRATION OF MARKED GRAPHS FEATURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL HORÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. VLADIMÍR JANOUŠEK, Ph.D.

BRNO 2014

Abstrakt

Tato práce prezentuje problematiku Petriho sítí jakožto modelovacího prostředku. Probrány jsou jejich vlastnosti a dělení na podtřídy, jejichž součástí jsou značené grafy. Cílem práce je analýza stavového prostoru Petriho sítí za účelem zkoumání jejich vlastností. Výsledkem práce je nástroj, který umožní tuto analýzu a případně rozšíří využití Petriho sítí.

Abstract

This thesis presents Petri nets as a modelling tool, defines their features and subclasses, including marked graphs. The goal of this paper is to inspect the features of Petri nets by analyzing their state space. As the output of this thesis, a tool for analyzing the features will be designed and implemented, with potential to extend the usability of Petri nets.

Klíčová slova

Petriho sítě, značené grafy, strom dosažitelných značení, modelování systémů, PNML, PNC, PIPE2.

Keywords

Petri nets, marked graphs, reachability tree, systems modelling, PNML, PNC, PIPE2.

Citace

Michal Horák: Demonstrace vlastností značených grafů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Demonstrace vlastností značených grafů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Doc. Ing. Janouška, Ph.D.

.....
Michal Horák
15. května 2014

Poděkování

Děkuji tímto Doc. Ing. Janouškovi, Ph.D. za odbornou pomoc při osobních konzultacích.

Obsah

Kapitola 1

Úvod

Počítačové systémy jsou dnes nedílnou součástí téměř všech odvětví fungování společnosti. Jejich bezchybná a také efektivní funkčnost je ale omezena jejich návrhem, který často postrádá dostatečné prostředky pro modelování a simulaci stavů při jejich vývoji. Jedním z mocných prostředků, který se snaží vývoj systémů usnadnit, jsou Petriho sítě. V této práci si nejdříve formálně představíme Petriho sítě z hlediska jednotlivých prvků a struktury a uvedeme jejich základní dělení. Zaměříme se na vlastnosti, které jsou u nich zkoumány. Dále v kapitole 2.3.1 uvedeme vztah ke značeným grafům a důvody od jejich abstrakce ve zbytku práce.

V Kapitole 3 rozebereme vlastnosti Petriho sítí dopodrobna a navážeme je na analýzu modelovaných systémů. Představíme si techniku pro reprezentaci stavového prostoru, strom dosažitelných značení, a způsob jeho analýzy. V návaznosti na shrnuté poznatky o vlastnostech Petriho sítí, jejich využití při modelování a existujících nástrojích bude v Kapitole 4 shrnut a rozebrán návrh sbírky nástrojů pro demonstraci vlastností Petriho sítí, včetně zohlednění formátů, ve kterých jsou sítě ukládány. Bude představen analyzátor vlastností Petriho sítí a způsob, jakým má dané vlastnosti ověřit. V Kapitole 5 bude následně popsána implementace navržených komponent a jejich funkcí a shrnuty implementační prostředky.

V Závěru se objeví shrnutí poznatků a bude odkázáno na případy užití, které budou součástí této práce. Bude zhodnocen celkový postup a diskutovány další možné směry výzkumu a vývoje v této oblasti.

Kapitola 2

Petriho sítě a značené grafy

Petriho sítěmi označujeme třídu diskrétních matematických nástrojů pro popis řídících toků uvnitř modelovaných systémů. Poprvé je popsal roku 1962 Carl Adam Petri ve své dizertační práci Kommunikation mit Automaten[23], v níž představil nové koncepty popisu systémů s podmínkami a událostmi a využil při tom dekompozici systému na podsystémy popsané konečnými automaty. Dnes jsou Petriho sítě používány hlavně při modelování, simulaci a analýze paralelních a distribuovaných systémů, typicky při popisu komunikačních protokolů, dále paralelních databázových systémů, překladačů nebo počítačových sítí. Jejich využití ale sahá i za hranice počítačových systémů, lze jich využít např. v telekomunikacích, administrativě a dalších oborech.

Díky svému širokému využití vzniklo za posledních několik desítek let řada publikací, praktik, nástrojů a dalších prostředků pracujících s Petriho sítěmi. Všeobecně uznávaným online úložištěm pro další odkazy v této oblasti je web *the Petri Nets World* [8] provozovaný univerzitou v Hamburku.

V této kapitole uvedeme definici Petriho sítě, popíšeme její prvky i změnu stavu v síti a rozdělíme je podle jejich vlastností.

2.1 Formální definice

Nejdříve si uveďme základní definice pro *síť* a *P/T Petriho síť* podle [13]. Pro jejich pochopení předpodkládáme znalost základních matematických pojmů, jako jsou množina, zobrazení, sjednocení, průnik a nebo kartézský součin. Pro více informací o této problematice viz např. [13] nebo [17].

Definice 2.1.1. Síť *je trojice* $N = (P, T, F)$, *kde*

1. $P = p_1, p_2, \ldots, p_n$ *je konečná množina míst,*

2. $T = t_1, t_2, \ldots, t_n$ *je konečná množina přechodů,*

3. $F \subseteq (P \times T) \cup (T \times P)$ *je množina hran, někdy také toková relace,*

4. $P \cap T = \varnothing \wedge P \cup T \neq \varnothing$ *(množiny* P *a* T *jsou disjunktní).*

Z této obecné definice jsou odvozeny definice dalších typů sítí, které jsou probrány v kapitole 2.3. V této práci se zaměříme na P/T Petriho sítě jakožto dominantní typ sítí

určených pro modelování. Pro jejich definici je třeba nejdříve rozšířit množinu přirozených čísel N o symbol ω.

Definice 2.1.2. *Rozšíření* $\mathbb{N}$ *na* $\mathbb{N} \cup \{\omega\}$.
Uspořádání $<$ *a operace* $+$ *a* $-$ *na množině* $\mathbb{N}$ *rozšíříme v množině* $\mathbb{N} \cup \{\omega\}$ *takto:*

1. $\forall n \in \mathbb{N} : n < \omega$

2. $\forall m \in \mathbb{N} \cup \{\omega\} : m + \omega = \omega + m;\ \omega - m = \omega$

3. *Nechť* $A \subseteq \mathbb{N}$, *pak*

$$sup(A) = \begin{cases} a, & \text{jestliže } a \in A \ \wedge \forall a' \in A \ : a' \leq a \\ \omega, & \text{jestliže } \forall n \in N : \exists a \in A : n \leq a \end{cases}$$

Nyní lze zavést definici *P/T Petriho sítě* následovně:

Definice 2.1.3. P/T Petriho síť *(Place/Transition Petri net) je šestice* $N = (P, T, F, W, K, M_0)$, *kde*

1. *trojice* (P, T, F) *je konečná síť,*

2. *zobrazení* $W : F \to \mathbb{N} \setminus \{0\}$ *je ohodnocení hran grafu sítě určující* váhu *každé hrany,*

3. *zobrazení* $K : P \to \mathbb{N} \cup \{\omega\}$ *specifikuje* počáteční kapacitu *(i neomezenou) každého místa,*

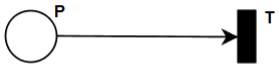
4. *zobrazení* $M_0 : P \to \mathbb{N} \cup \{\omega\}$ *je* počáteční značení *míst sítě respektující kapacity míst, tj.* $\forall p \in P : M_0(p) \leq K(p)$.

Síť N s počátečním značením M_0 je značena (N, M_0). Konkrétní prvky sítě jsou podrobně probrány v další kapitole.

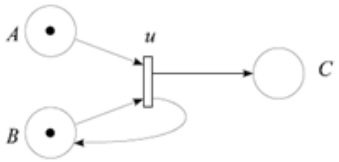
2.2 Charakteristika a prvky Petriho sítě

Zde následuje shrnutí základních informací nutných k pochopení sémantiky Petriho sítí jako celku i jejich jednotlivých prvků. Popíšeme jak jednotlivé prvky, tak situaci změny stavu v síti.

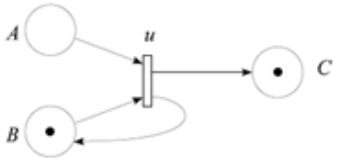
Místa, přechody, hrany Petriho síť pracuje se stavy systému, ty jsou reprezentány kružnicemi a nazývají se *místa* a dále událostmi systému, které jsou reprezentovány obdélníčky nebo také úsečkami, a nazývají se *přechody*. Spojnice těchto dvou prvků znázorněné úsečkou se šipkou udávající směr se nazývají *hrany*. Z Definice 2.1.3 plyne, že hrany v P/T Petriho sítích mohou spojovat pouze elementy různého typu – tedy místo s přechodem nebo obráceně.



Obrázek 2.1: Na obrázku vidíme místo P, přechod T a hranu, která je spojuje.



(a) událost může nastat

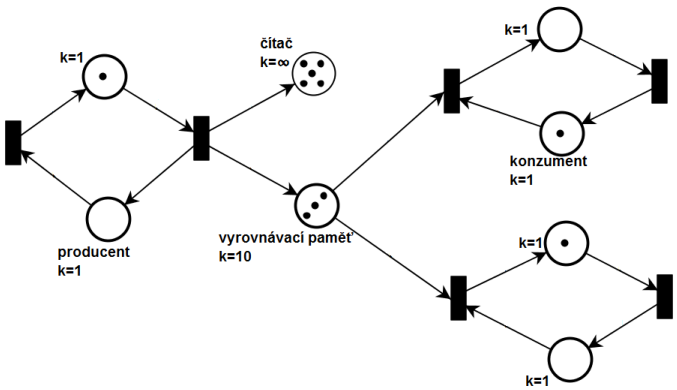


(b) po proběhnutí události

Obrázek 2.2: Značení míst.

Značení místa Abychom dokázali říci, kdy se systém nachází ve stavu, ve kterém platí určitá podmínka, je zaveden pojem *značení místa*. Pro grafickou reprezentaci se používají tečky uvnitř příslušného místa. Pokud je daná podmínka logická (nabývá-li pouze binárních hodnot), místo korespondující této podmínce buď obsahuje tečku uprostřed místa (konkrétní podmínka je platná) nebo je prázdné (podmínka je neplatná). Na Obrázku 2.1 vidíme značení míst a) před a b) po proběhnutí události u.

Obecně mohou Petriho sítě v jakémkoliv místě obsahovat libovolné množství značek za předpokladu, že kapacita k místa není omezena, tedy $k = \omega$. Jinak je množství značek shora omezené kapacitou místa; v případě, že by událostí vzniklo v některém místě vyšší než maximální povolené značení, se přebytečné značky zahodí. Využití kapacity míst je ilustrováno na Obrázku 2.3.

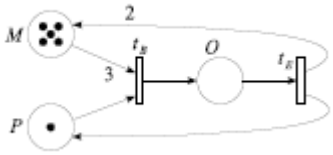


Obrázek 2.3: Ilustrace kapacity míst. Místa modelující čítač a vyrovnávací paměť jí využívají k modelování své omezenosti.

Značení sítě Vezmeme-li v úvahu síť jako celek, můžeme její stav jednoznačně určit celkovým *značením sítě* M, tedy souhrnem značení všech míst sítě. Udává se jako sekvence značení jednotlivých míst, přičemž záleží na jejich pořadí. V síti na Obrázku 2.2 jsou pro posloupnost míst A, B, C znázorněna značení $M = 1, 1, 0$ před a $M = 0, 1, 1$ po proběhnutí události u.

Ohodnocení hrany Místo můžeme obecněji interpretovat jako parciální (tedy částečný) stav systému. Událost pak může být podmíněna minimálním počtem značek konkrétního místa. To je reprezentováno tzv. *ohodnocením hrany*.

Na Obrázku 2.4 vidíme využití takových ohodnocených hran.



Obrázek 2.4: Ilustrace ohodnocení hran.

2.2.1 Pravidlo pro provedení přechodu

Pravidlo pro provedení přechodu definuje, jakým způsobem je modelována změna stavu v systému, tedy proběhnutí události. Je popsáno 3 podmínkami:

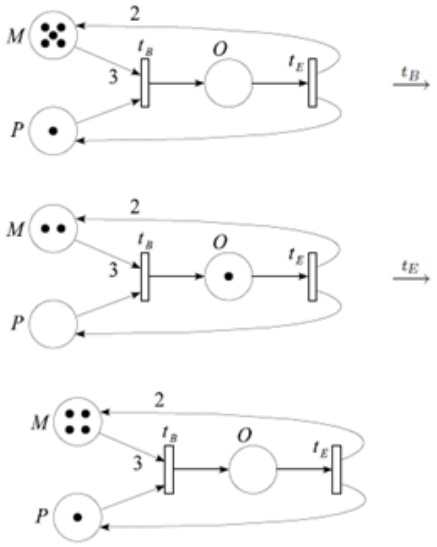
1. Přechod t je aktivní, pokud všechna jeho vstupní místa p mají alespoň značení rovné $w(p, t)$ značek, kde $w(p, t)$ je váha hrany z p do t.
2. Aktivní přechod může a nemusí být proveden, a to v závislosti na tom, zda daná událost opravdu nastane.
3. Při provedení aktivního přechodu t dojde k odebrání $w(p_{IN}, t)$ značek z každého vstupního místa p_{IN} a přidání $w(t, p_{OUT})$ značek do každého výstupního místa p_{OUT}, kde $w(p_{IN}, t)$ značí váhy hran ze vstupních míst do t a $w(t, p_{OUT})$ značí váhy hran z t do výstupních míst.

Příklad: Na Obrázku 2.5 vidíme ukázku systému s místy M, P, O a přechody t_B, t_E. Ohodnocení hran nám říká, že k provedení přechodu t_B je třeba, aby místo M obsahovalo alespoň tři značky, P alespoň jednu. Provedením přechodu t_B daný počet značek ubyde z těchto míst a místu O bude přiřazena jedna značka. Následným provedením přechodu t_E se vrátí jedna značka do místa P a dvě značky do místa M.

2.3 Dělení Petriho sítí

Existují různé podtřídy Petriho sítí. Uvedeme si jejich rozdělení, které je motivováno zjištěním dvou základních aspektů formálních modelů: jejich *modelovací a rozhodovací mocnosti*.

Modelovací mocnost popisuje schopnost reprezentovat systém určený k modelování tak, aby model přesně reprezentoval modelovaný systém. *Rozhodovací mocnost* pak popisuje schopnost analýzy konkrétních konfigurací daného modelu pro účely zjištění vlastností



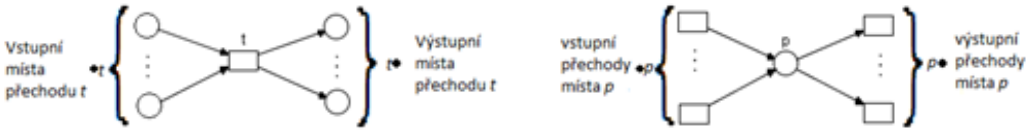
Obrázek 2.5: Tři stavy sítě, k nimž došlo postupným provedením přechodů t_B a t_E.

modelovaného systému. [22] Tyto dva faktory jsou často postaveny proti sobě, jak je také dále nastíněno.

Pro jednodušší popis vlastností, na základě kterých rozdělíme Petriho sítě do podtříd, zavádíme následující notaci pro popis vstupních a výstupních prvků.

$$\bullet t = \{p|(p,t) \in F\} \quad \text{je } \textit{množina vstupních míst přechodu } t \tag{2.1a}$$
$$t\bullet = \{p|(t,p) \in F\} \quad \text{je } \textit{množina výstupních míst přechodu } t \tag{2.1b}$$
$$\bullet p = \{t|(t,p) \in F\} \quad \text{je } \textit{množina vstupních přechodů místa } p \tag{2.1c}$$
$$p\bullet = \{t|(p,t) \in F\} \quad \text{je } \textit{množina výstupních přechodů místa } p \tag{2.1d}$$

Ilustrace této notace je znázorněna na Obrázku 2.6.



(a) Symboly pro množinu vstupních a výstupních míst přechodu t

(b) Symboly pro množinu vstupních a výstupních přechodů místa p

Obrázek 2.6: Notace pro popis vstupních a výstupních prvků.

Na základě struktur vyskytujících se v konkrétních typech sítí jsou P/T sítě rozděleny do 5 kategorií[21]. Uvádíme jak původní anglické termíny pro lepší orientaci v anglické literatuře, tak české ekvivalenty, které ale nejsou všeobecně ustálené.

1. *Stavové stroje* (SM – State Machines)[1] jsou obyčejné Petriho sítě[2], ve kterých každý přechod t má právě jedno vstupní a právě jedno výstupní místo, formálně zapsáno

$$|\bullet t| = |t \bullet| = 1 \quad \text{pro všechna } t \in T.$$

2. *Značené grafy* (MG – Marked Graphs) jsou obyčejné Petriho sítě, ve kterých každé místo p má právě jeden vstupní a právě jeden výstupní přechod, formálně zapsáno

$$|\bullet p| = |p \bullet| = 1 \quad \text{pro všechna } p \in P.$$

3. *Sítě s volným výběrem* (FC – Free-choice nets) jsou obyčejné Petriho sítě, ve kterých každá hrana je buď unikátní výstupní hranou nebo unikátní vstupní hranou nějakého přechodu (nebo také ve které mají všichni předchůdci každého přechodu stejnou množinu následníků), formálně zapsáno

$$|p \bullet| \leq 1 \vee \bullet(p\bullet) = \{p\} \quad \text{pro všechna } p \in P.$$

   nebo ekvivalentně

$$p_1 \bullet \cap p_2\bullet \neq \varnothing \implies |p_1 \bullet| = |p_2 \bullet| = 1 \quad \text{pro všechna } p_1, p_2 \in P.$$

4. *Rozšířené sítě s volným výběrem* (EFC – Extended Free-choice nets) jsou obyčejné Petriho sítě, pro které platí

$$p_1 \bullet \cap p_2\bullet \neq \varnothing \implies p_1\bullet = p_2 \bullet \quad \text{pro všechna } p_1, p_2 \in P.$$

5. *Asymetrické sítě s volným výběrem* (AC – Asymetric Choice nets), také známé jako *jednoduché sítě* (simple nets) jsou obyčejné Petriho sítě, pro které platí

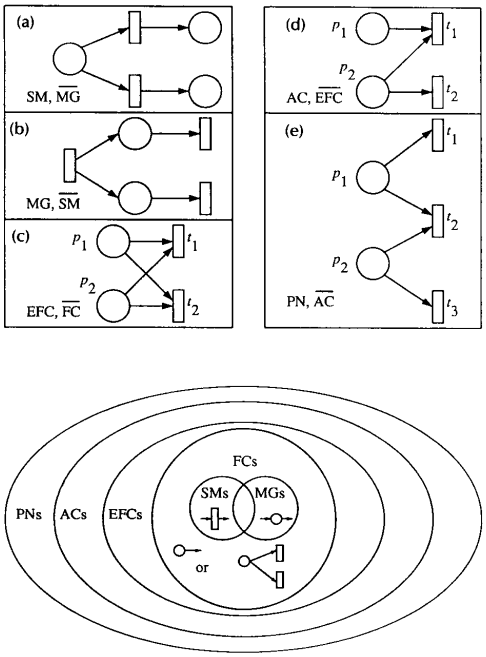
$$p_1 \bullet \cap p_2\bullet \neq \varnothing \implies p_1\bullet \subseteq p_2 \bullet \vee p_1\bullet \supseteq p_2 \bullet \quad \text{pro všechna } p_1, p_2 \in P.$$

Shrnutí poznatků Shrnout poznatky o podtřídách Petriho sítí můžeme následovně:

- Modelovací mocnost obecných P/T sítí je největší, mají tedy největší výpočetní sílu Lze jimi modelovat širokou škálu systémů.
- Neomezenost značení míst obecných P/T sítí implikuje potenciálně nekonečný stavový prostor.
- Značené grafy a stavové stroje mají nejnižší modelovací mocnost pro svá striktní omezení.
- Za předpokladu, že model systému může být modelován značenými grafy nebo stavovými stroji, popř. na ně převeden, jsou nejjednodušším prostředkem pro analýzu vlastností – mají nejvyšší rozhodovací mocnost.

Obrázek 2.7[21] ilustruje uvedené rozdělení ukázkou charakteristických struktur a také také Vennův diagram popisující vzájemný vztah vyjmenovaných podtříd.

[1] FSM (Finite State Machines) – konečný stavový stroj (konečný automat).
[2] *Obyčejné Petriho sítě* jsou takové Petriho sítě, u nichž váha všech hran v síti je rovna 1.



Obrázek 2.7: Struktury charakteristické pro konkrétní podtřídy P/T Petriho sítí a diagram jejich vzájemnného vztahu, kde $\overline{MG}$, $\overline{SM}$ atd. značí, že daná struktura nepatří do této podtřídy.

2.3.1 Značené grafy

Na základě předchozího rozdělení můžeme značené grafy jednoduše definovat[13] takto:

Definice 2.3.1. Značený graf *je Petriho síť* $N = (P, T, F, W, K, M_0)$*, jestliže splňuje tyto podmínky:*

1. $\forall t_1, t_2 \in T : t_1 F * t_2$*, tedy graf je tzv.* silně souvislý*,*

2. $\forall p \in P : |\bullet p| = |p \bullet| = 1 \wedge W(\bullet p, p) = W(p, p\bullet) = 1.$

Kapitola 3

Analýza systémů Petriho sítěmi

V předchozí kapitole bylo uvedeno rozdělení Petriho sítí a definice značených grafů. Bylo ukázáno, že pro značené grafy musí platit silná souvislost a ohodnocení všech jejich hran musí být rovno 1. Tato omezení neumožňují modelovat značenými grafy celou řadu systémů, a tak omezují jejich použitelnost jako modelovacího prostředku. Proto v dalších kapitolách abstrahujeme od značených grafů a i přes zvýšenou komplexnost se zaměříme na analýzu systémů modelovaných obecnými P/T sítěmi. Popíšeme vlastnosti, které jsou u modelovaných systémů nejčastěji zkoumány, hlavní techniku, která se používá při této analýze a také představíme dostupné prostředky a nástroje pro práci s Petriho sítěmi. V závěrečné kapitole bude tato abstrakce zhodnocena.

3.1 Zkoumané vlastnosti

Stav modelovaného systému může být jednoznačně určen značením sítě a úkolem analýzy sítě bývá často zjistit, zda dané značení může v grafu nastat a po jaké sekvenci událostí. Tyto aspekty nazýváme *dosažitelnost značení* a *sekvence přechodů*. Dále nás může zajímat, zda jsou z určitého stavu v systému východiska a jaká. Tyto vlastnosti jsou popsány jako *živost*, popř. *uváznutí* a *aktivní přechody*. Také můžeme analyzovat např. limity systému (*omezenost*) nebo další vlastnosti. Popišme si základní zkoumané vlastnosti P/T sítí se zaměřením na jejich využití.

3.1.1 Problém dosažitelnosti

Dosažitelnost je základní zkoumaná vlastnost jakéhokoliv analyzovaného systému.

Definice 3.1.1. Dosažitelné značení – *Sekvencí provádění aktivních přechodů z počátečního značení M_0 získáváme sekvenci značení $M_1, M_2 \dots M_n$. M_n je* dosažitelné *z M_0, existuje-li sekvence přechodů, která transformuje M_0 na M_n. Tato sekvence se značí $\sigma = t_1 t_2 \dots t_n$.*

Problém dosažitelnosti je obecně rozhodnutelný – je nutné prohledat množinu všech dosažitelných značení a určit, zda značení M_n je podmnožinou této množiny. Základní otázkou tedy je, jakým způsobem reprezentovat stavový prostor dané sítě a všechny její možné transformace. Většina problémů řešených v rámci analýzy Petriho sítí je převoditelná na problém dosažitelnosti, proto bude důležitou součástí při návrhu v další kapitole.

3.1.2 Živost a uváznutí

Uváznutí nebo-li *deadlock* je situace, při které dokončení jedné akce je podmíněno předchozím dokončením druhé akce, přičemž druhá akce je stejně závislá na předchozím dokončení první akce. Definice může být také zobecněna pro více akcí.

Živost Živost sítě se dá jednoduše popsat jako absence uváznutí v síti. Petriho síť N je *živá* (nebo také prvotní značení M_0 je živým značením sítě N), když, bez ohledu na to jaké značení bylo dosaženo z M_0, je možné provést jakýkoliv přechod v síti, po nějaké sekvenci přechodů.

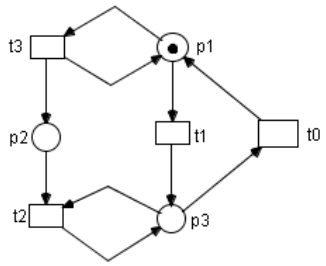
Živost je ideální vlastností mnoha systémů. Jelikož je ale příliš náročné pro některé systémy tuto vlastnost ověřit, zavádí se následující hladiny živosti[13]. Přechod t je

1. *živý na hladině 0* (L_0-živost), není-li proveditelný v žádném značení z $[M_0\rangle$, je tedy neživý.
2. *živý na hladině 1* (L_1-živost), jestliže existuje značení $M \in [M_0\rangle$ takové, že přechod t je M-proveditelný.
3. *živý na hladině 2* (L_2-živost), jestliže pro každé $n \in \mathbb{N}$ Nexistuje výpočetní posloupnost, ve které se přechod tvyskytuje alespoň n-krát.
4. *živý na hladině 3* (L_3-živost), jestliže existuje výpočetní posloupnost, ve které se přechod t vyskytuje nekonečněkrát.
5. *živý na hladině 4* (L_4-živost), jestliže pro každé $M_1 \in [M_0\rangle$ existuje značení $M_2 \in [M\rangle$ takové, že přechod t je M_2-proveditelný.

Petriho síť je potom živá na hladině h, jestliže všechny její přechody jsou živé na hladině h nebo vyšší. Živá Petriho síť podle předchozí definice odpovídá živosti na hladině 4. Můžeme pak odvodit následující implikace:

$$L_4\text{-živost} \implies L_3\text{-živost} \implies L_2\text{-živost} \implies L_1\text{-živost} \implies L_0\text{-živost}$$

Konkrétní příklad na Obrázku 3.1:



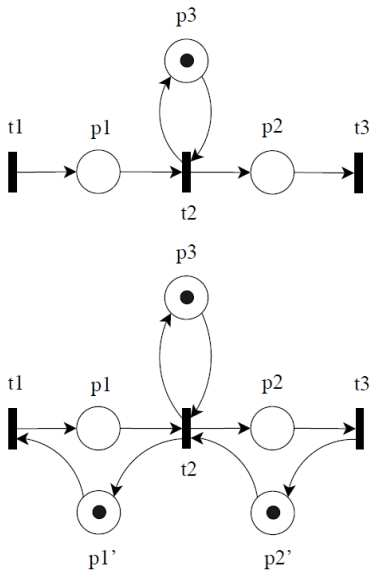
Obrázek 3.1: Přechod t_0, t_1, t_2 a t_3 jsou přesně v tomto pořadí: mrtvý, L_0-živý, L_1-živý, L_2-živý a L_3-živý.

3.1.3 Bezpečnost

Bezpečnost sítě je úzce spojena s vlastností, které říkáme *omezenost*.

Omezenost Petriho síť je *omezená*, pokud počet značek v každém místě sítě nepřesáhne konečnou hodnotu k při jakémkoliv značení dosažitelného z M_0.

Bezpečnost Petriho síť je *bezpečná*, platí-li pro ni podmínka omezenosti s $k = 1$.



Obrázek 3.2: Ukázka sítě, která není bezpečná (nahoře) a jí odpovídající bezpečná síť (dole). (Převzato z [13].

3.1.4 Další vlastnosti

Mezi dalšími vlastnostmi, které můžeme zkoumat, patří např. *konzervativnost* nebo *reverzibilita*.

Konzervativnost je vlastnost systémů, u nichž je požadována kontrola nad zdroji systému. Síť je *konzervativní* tehdy, když součet všech značení v síti zůstává konstatntní.

Reverzibilita je také často zkoumanou vlastností mnoha systémů. Systém je *reverzibilní*, pokud z jakéhokoliv značení dosažitelného z M_0 je po konečné sekvenci přechodů možno dosáhnout zpět tohoto počátečního značení.

3.2 Strom dosažitelnosti

Tyto vlastnosti bývají nejčastěji zjišťovány pomocí tzv. *stromu dosažitelných značení* (dále také *strom dosažitelnosti*). Strom dosažitelnosti je struktura používaná pro reprezentaci možných změn v síti založená na přechodové funkci sítě. Skládá se uzlů popisujících konkrétní značení sítě, větve stromu pak reprezentují možné aktivní přechody. Tvorba stromu je dána následujícími pravidly:

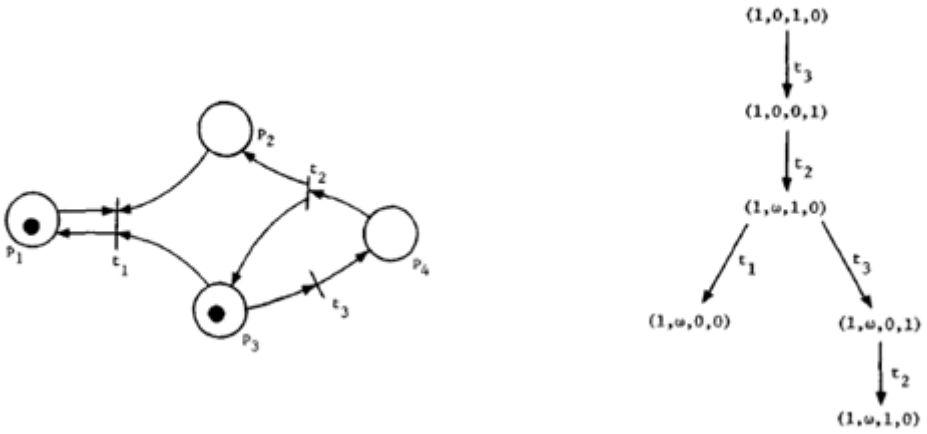
1. Kořenový uzel stromu reprezentuje počáteční značení.

2. Přímý potomek (syn) každého uzlu stromu je přímo dosažitelným značením svého předchůdce. Hrany stromu reprezentují přechody provedené z rodičovského značení do značení potomka.

3. Pokud nově vygenerované značení x je shodné s některým ze značení jeho předchůdců, je uzel s tímto značením označen jako koncový.

4. Pokud nově vygenerované značení x je větší než značení y v kterémkoliv uzlu na cestě ke kořeni, pak ty komponenty značení x, které jsou striktně větší než korespondující komponenty značení y jsou nahrazeny symbolem ω.

Prvními dvěma body lze vytvořit celý stavový prostor. Ten ale může být pro obecnou Petriho síť nekonečný, proto jsou přidány kroky 3 a 4 a je využito zástupného symbolu reprezentujícího jakkoliv vysoké značení v daném místě. Na Obrázku 3.3 vidíme příklad Petriho sítě a jí korespondující strom dosažitelnosti.



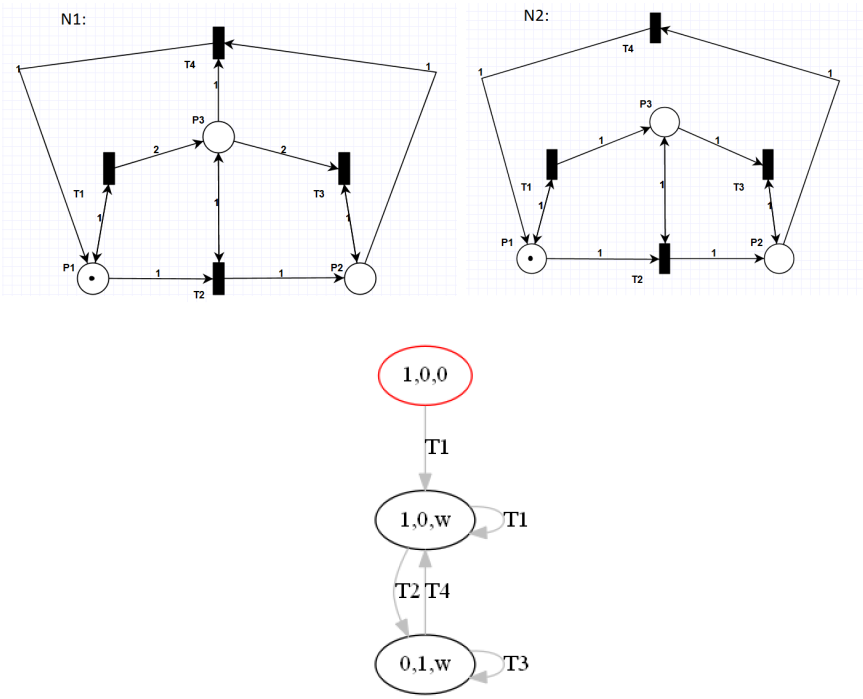
Obrázek 3.3: Příklad Petriho sítě a korespondujícího stromu dosažitelnosti[22]. Všimněme si symbolů ω implikujících nekonečný stavový prostor, ten je přitom jednoduše reprezentovaný konečným stromem dosažitelnosti. Lze jednoduše dokázat, že místo P_2 může hromadit neomezené množství značek v závislosti na počtu provedených přechodů t_2.

Strom dosažitelnosti může mít více podob. Duplikátní uzly např. mohou být odstraněny a ponechána může být pouze šipka s přechodem tvořící ve stromu smyčku. Takovýto strom potom můžeme nazývat také *grafem dosažitelnosti*.

3.2.1 Využití a omezení

Množství problémů může být převedeno právě na problém dosažitelnosti. Konstrukce stromu dosažitelných značení je tedy důležitým krokem při analýze různých vlastností modelovaného systému a jeho využití je tak velmi široké. Použití stromu dosažitelnosti má však svá omezení, která jsou patrná v otázkách živosti sítě a dosažitelnosti značení. Konstrukce stromu probíhá na základě abstrakce přechodové funkce, což způsobuje určitou ztrátu informace. Vztah mezi celou Petriho sítí a stromem dosažitelnosti pak není jednoznačný. Na Obrázku 3.4 vidíme ukázku dvou rozdílných sítí. Přechodová funkce pro obě dvě je však stejná a s ní i strom dosažitelnosti.

Síť $N1$ není živá, např. provedením sekvence přechodů $t_1t_2t_3$ dostaneme značení $(0, 1, 0)$, při kterém již žádný z přechodů není proveditelný. Zatímco síť $N2$ je živá, jelikož nelze dosáhnout značení $(0, 1, 0)$. Vrchol $(0, 1, \omega)$ sice reprezentuje nekonečnou množinu značení lišících se poslední složkou, avšak neklade podmínku, že libovolný prvkek za něj dosazený, je v síti opravdu dosažitelný.



Obrázek 3.4: Dvě rozdílné sítě $N1$ a $N2$ a jejich shodný strom dosažitelnosti.

Strom dosažitelnosti tedy lze použít pro analýzu živosti a dosažitelnosti pouze omezeně:

- Síť můžeme prohlásit za neživou, pokud její strom dosažitelnosti obsahuje nějaký koncový vrchol, tedy vrchol způsobující uvíznutí.
- Není-li značení M pokrytelné žádným značením stromu, pak není dosažitelné.

Složitost Dalším problémem je složitost výpočtu. Byl proveden důkaz[18], že pro sítě, jejichž komplexita (počet elementů a značek) roste lineárně, roste náročnost jim odpovídajících generovaných grafů/stromů dosažitelnosti rychleji, než libovolná primitivně rekurzivní funkce.

Právě kvůli složitosti výpočtu stavového prostoru a snaze eliminovat výše zmíněná omezení probíhá v této oblasti hlubší výzkum a zdokonalování procesu. V sekci 3.2.2 pouze odkážeme na referenční literaturu pro další studium tohoto problému. Zkoumání a implementace těchto postupů ale sahá za hranice této práce.

3.2.2 Metoda KLMST a další vývoj

Problém dosažitelnosti se stal hlavním cílem výzkumu v oblasti Petriho sítí (v originální literatuře často označovných jako VAS – Vector Addition Systems). V roce 1977 G.S.Sacerdote a R.L.Tenney uvedli[24] částečný důkaz řešitelnosti tohoto problému. Jejich důkaz pak dokončil E.W.Mayr[20] a dále zjednodušili S.R.Kosarju[16] a J.L.Lambert[18]. *KLMST metoda*, pojmenovaná po těchto čelních představitelích, se dále stala předmětem dalšího zkoumání. V současné době se výzkumem zabývá m.j. J.Leroux, který ve své práci[19] popisuje řešení šetrnější na časové i paměťové nároky a odkazuje se na další zdroje. Přesto ale tyto postupy zůstávají velmi komplexní a jejich důkazy netriviální. Vhodnost použití ve specifických případech také není diskutována.

3.3 Prostředky

V této kapitole popíšeme formát, v němž jsou Petriho sítě ukládány a shrneme funkcionalitu nabízenou existujícími nástroji pro práci s nimi.

3.3.1 Petri Net Markup Langugage (PNML)

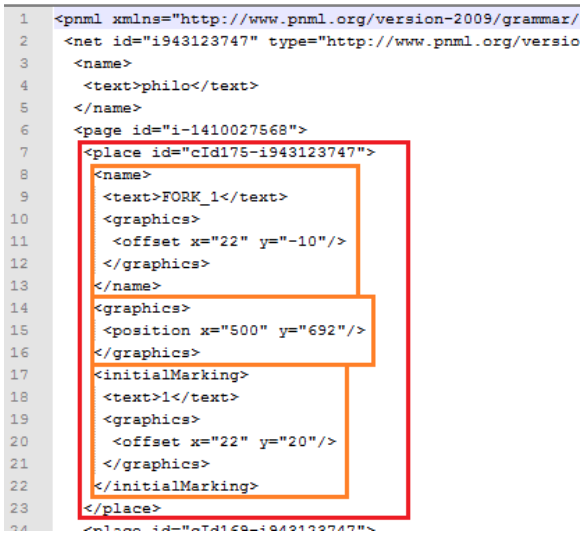
PNML (Petri Net Markup Langugage)[15] je přenositelný formát založený na XML sloužící k popisu Petriho sítí. Nabízí techniku pro definici nových typů Petriho sítí, čímž se stává univerzálním a flexibilním prostředkem pro podporu různých verzí Petriho sítí. V současnosti je PNML součástí standardu ISO/IEC 15909 jako syntax pro podporu tří hlavních verzí Petriho sítí: P/T sítí, vysokoúrovňovňových Petriho sítí a symetrických Petriho sítí.

V příloze A je k nalezení diagram znázorňující jádro PNML modelu a dále jeho konkretizaci pro podporu P/T Petriho sítí PT-net balíčkem. Mezi hlavní principy PNML patří:

- Typ obsažené Petriho sítě je zadán jako URL odkaz na jméno konkrétního balíčku s definicí.
- Petriho síť je brána jako označený orientovaný graf, v němž všechny specifické informace jsou reprezentovány pomocí značek (*labels*), které mohou být spojeny s konktrétními uzly sítě, hranami nebo sítí samotnou. Typicky se jedná o jméno uzlu, značení místa, grafické informace a další.
- Jádro PNML modelu konkrétní značky nedefinuje. Jedinou výjimkou je značka `name`, která je dostupná ve všech balíčcích.

Příklad PNML formátu za použití PT-net balíčku je ilustrován Obrázkem 3.5:

Rozšířenost Mezi nástroje podporující jednotný PNML formát patří např.[6] ePNK, OWLS2, PNML Framework, Coloane (Université et Marie Curie), Tina (LAAS/CNRS), Petri Web (Technische Universiteit Eindhoven), ProM (TU/e). Další reference lze také najít na webu *the Petri Nets World* ([7] nebo [9]). Přestože je PNML formátem standardizovaným, není zdaleka všeobecně rozšířený. Některé nástroje používají vlastní formát, ať už založený na XML (např. PIPE – viz Kapitola 3.3.3), nebo v jiné notaci (např. ARP[1]).



Obrázek 3.5: Část PNML reprezentace sítě, v níž je zachyceno jedno konkrétní místo sítě – patrné jsou části pro informace o názvu místa, obecné grafické informaci a značení.

3.3.2 Existující nástroje

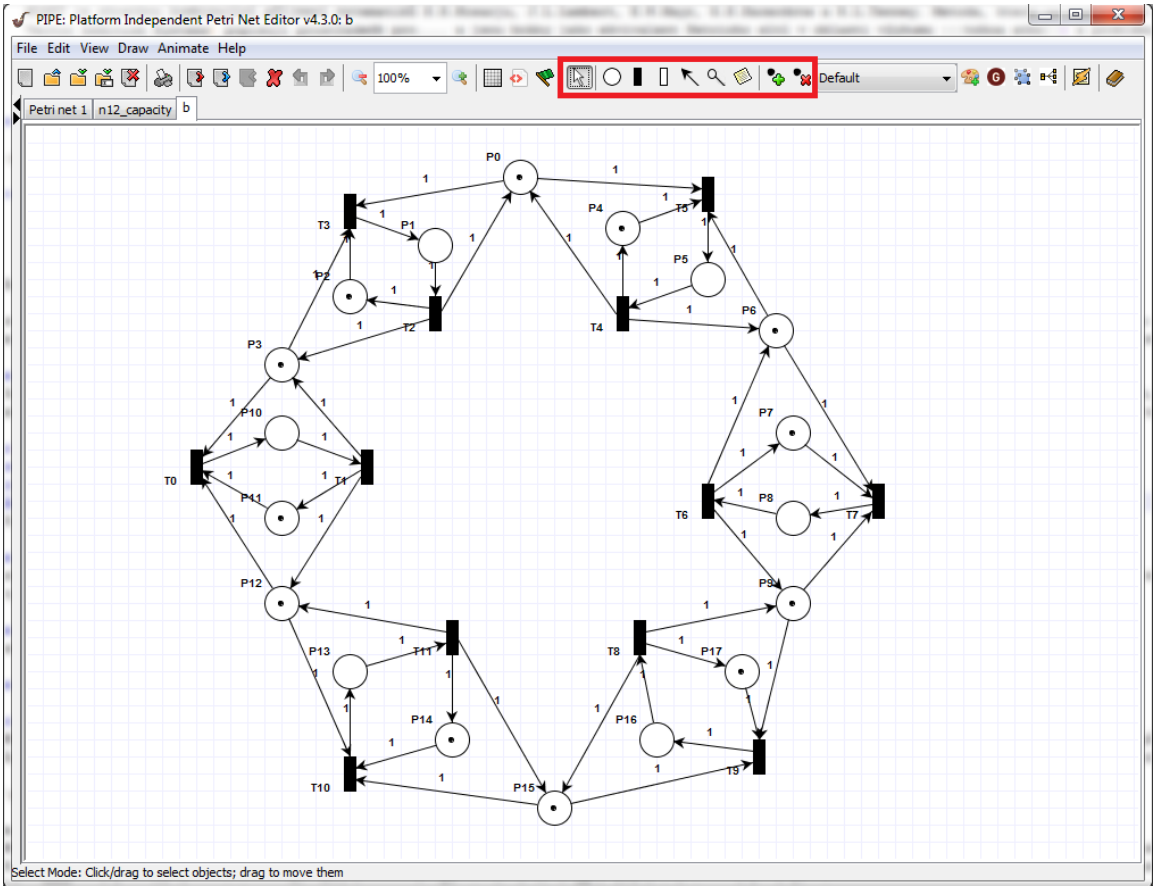
Editory Existuje celá řada volně dostupných editorů Petriho sítí. Od jednoduché editace základních komponent a automatického lomení/ohybu hran, nabízí některé nástroje i pokročilé funkce, jako např. animace, 3D pohled, abstrakce částí sítě, časované přechody, prvky vysokoúrovňových Petriho sítí nebo jiné. Ukázka prostředí takového editoru s ilustrací jeho možností je vidět na Obrázku 3.6a.

Analýza, simulace Některé editory nabízejí např. validaci struktury sítě z hlediska formalismu P/T nebo jiného typu sítě. Můžeme také najít funkce pro zobrazení aktivních přechodů, krokovou simulaci nebo dokonce simulaci složitějších sítí. (Ukázka modulu takové analýzy viz Obrázek 3.6b. Ne vždy však jsou tyto funkce volně dostupné nebo kombinovatelné.

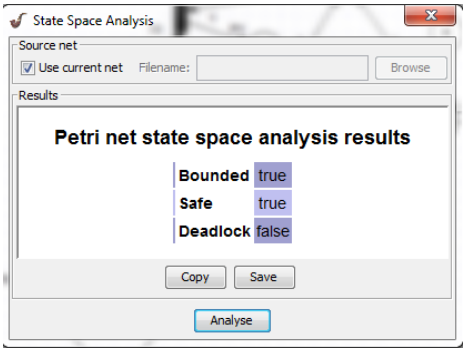
3.3.3 PIPE2

PIPE2 (Platform Independent Petri Net Editor)[10] je platformě nezávislý editor Petriho sítí. Nabízí grafické rozhraní pro pokročilou editaci P/T sítí, simulační funkci a moduly pro analýzu sítě, včetně základní klasifikace sítě, analýzy stavového prostoru a generování grafu dosažitelnosti. Grafickou reprezentaci sítě lze serializovat do XML formátu, který ale není standardním PNML formátem. Prezentace výsledků modulů pro analýzu probíhá graficky, s možným manuálním uložením do HTML formátu. Graf dosažitelných značení je generován pouze graficky, bez možnosti jej nějak uložit a dále zpracovat. Tyto vlastnosti snižují využitelnost programu na vyšší úrovni, např. hromadným zpracováním.

Pro účely této práce byl tento program vybrán jako jeden z referenčních nástrojů pro svou přenositelnost, dobré grafické rozhraní a svou modulovou funkcionalitu. Validní P/T sítě v něm navržené budou brány jako možný vstup analýzy vyvíjeným nástrojem a výsledky analýz mohou být porovnány s výstupy programu PIPE2. Prostředí nástroje PIPE2 je ilustrováno Obrázky 3.6.



(a) Grafický editor. Zvýrazněna je hlavní funkcionalita obsahující mj. vkládání míst, přechodů, hran, a přidávání/odebírání značek v místech.



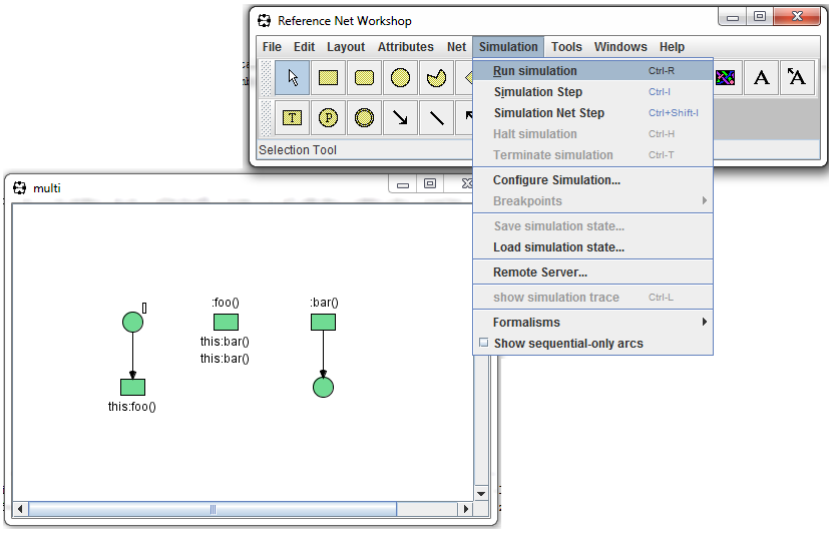
(b) Modul analýzy stavového prostoru sítě z předchozího obrázku. Vidíme, že síť je omezená, dokonce bezpečně omezená a neobsahuje uváznutí.

Obrázek 3.6: Prostředí nástroje PIPE2.

3.3.4 ReNeW

ReNeW (The Reference Net Workshop)[11] je jeden z velmi rozšířených nástrojů pro editaci a simulaci. Je zaměřen na referenční sítě, v nichž jednotlivé značky mohou být referencemi na libovolné typy objektů, např. jiné sítě. Vestavěnými typy značek jako n-tice a seznam, nabízenými typy hran a jinými charakteristikami nabízí velmi silnou základnu pro obecnou simulaci systémů. Také podporuje výstup sítě v PNML.

Přestože se svým zaměřením vzdalují P/T Petriho sítím, je možné je v tomto nástroji modelovat. Pro svou rozšířenost byl tedy zařazen jako jeden z referenčních nástrojů a validní P/T sítě v něm navržené budou brány jako možný vstup analýzy vyvíjeným nástrojem.



Obrázek 3.7: Ukázka prostředí nástroje ReNeW.

Kapitola 4

Analýza a návrh aplikace

V předchozí kapitole bylo popsáno využití Petriho sítí. Byly uvedeny hlavní vlastnosti, které jsou zkoumány při jejich analýze, rozebrána problematika stromu dosažitelnosti a diskutovány dostupné existující nástroje v této oblasti. Blíže byly uvedeny nástroje ReNeW a PIPE2, se kterými bude také dále pracováno. Tato kapitola má za úkol identifikovat základní požadavky na nově vyvíjený nástroj s ohledem na jeho maximálním využitelnost. Hlavním úkolem nástroje má být demonstrovat principy a vlastnosti značených grafů. Základním předpokladem pro další návrh bude již dříve zmíněná abstrakce od značených grafů k obecnějším P/T Petriho sítím odůvodněná v kapitole 2.3.1.

Byly definovány následující komponenty a funkcionalita:

- Načtení sítě a její analýza
  - Konstrukce stromu dosažitelnosti využívající zástupný symbol ω
  - Možnost vizualizace stromu dosažitelnosti
  - Analýza dosažitelných značení
  - Základní vlastnosti sítě: živost, omezenost, bezpečnost
  - Identifikace a výpis sekvencí přechodů k dosažitelnému značení
  - Identifikace a výpis aktivních přechodů při určitém značení
  - Rozpoznání nepodporovaného typu sítě
- Podpora XML formátu sítě nástroje PIPE2
- Podpora standardizovaného PNML formátu
  - Plná podpora P/T-net balíčku
  - Podpora P/T sítí nástroje Renew
  - Abstrakce nad PNML zjednodušeným, jednotným formátem obsahujícím pouze základní informace o síti
- Příkazové rozhraní
  - Jednotné rozhraní pro celou nástrojovou sadu
  - Podpora dávkového zpracování

4.1 Formát PNC

Pro potřeby analýzy sítě je PNML formát příliš obecný a obsahuje všeobecně velké množství nadbytečných informací. Kvůli zpracování sítě byla tedy navrhnuta abstrakce nad tímto formátem PNML, a to PNC (Petri Net CSV). Hlavními důvody pro zavedení tohoto minimalistického formátu jsou:

- Zjednodušit formát, ve kterém jsou sítě zpracovány pro účel analýzy.
- Oprostit se od grafických informací a informací specifických pro konkrétní nástroj, ve kterém je PNML vygenerováno.
- Konzolidace pouze těch dat, která jsou důležitá pro vytvoření objektu P/T sítě, nad kterou lze pak provádět analýzu (viz Kapitola 5). Zrychlení této analýzy.
- Neintegrací tohoto formátu přímo v analyzátoru nýbrž jeho samostatným vyčleněním navíc umožníme uživateli rychlou manuální tvorbu struktury sítě v PNC, aniž by ji musel graficky navrhovat.

Syntax Dokument *.PNC se skládá z jednoho či více elementů $< NET\_ELEM >$ oddělených koncem řádku $< EOL >$. Element může být buď místo, přechod nebo hrana a každý obsahuje atributy oddělené středníkem. $< ID >$ a $< NAME >$ popisují ID a jméno elementu, $< SRCID >$ a $< TGTID >$ u hran udávají ID objektu odkud a kam hrana směřuje, $< INIT\_MARKING >$ popisuje počáteční značení místa a $< INSCR >$ váhu hrany. Formát PNC může být zjednodušeně popsán pomocí terminálních a neterminálních symbolů takto:

$$< PNCDOCUMENT > \rightarrow (< NET\_ELEM > EOL)+ \tag{4.1}$$

$$< NET\_ELEM > \rightarrow P; < ID >; < NAME >; < INIT\_MARKING >; \tag{4.2}$$

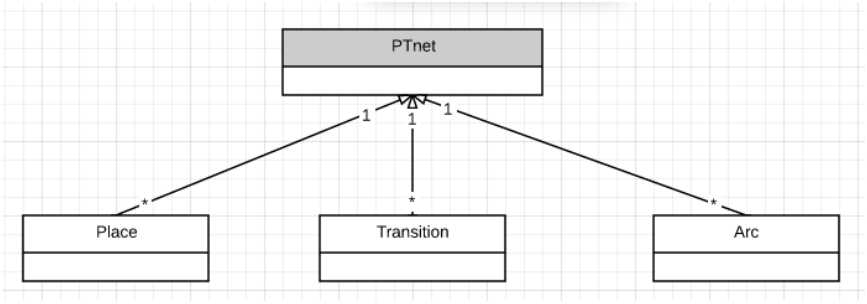
$$< NET\_ELEM > \rightarrow T; < ID >; < NAME >; \tag{4.3}$$

$$< NET\_ELEM > \rightarrow A; < ID >; < SRCID >; < TGTID >; < INSCR >; \tag{4.4}$$

Pozn.: Uživatelem zadané jméno místa/přechodu je nepovinným atributem a může být tedy použit prázdný řetězec. To samé platí pro počáteční značení místa a váhu hrany, u nichž se použije defaultních hodnot (0 pro počáteční značení a 1 pro váhu hrany). Oddělovače však musejí být zachovány.

4.2 Objekt Petriho sítě

Síť pro analýzu je nutno vhodně navrhnout z objektového hlediska, aby práce s ní byla co nejsnazší. Tento krok je výrazně zjednodušený tím, že síť bude načítána ve formátu PNC definovaném v předchozí kapitole. Jednotlivé navržené třídy `Place`, `Transition` a `Arc` návrhově kopírují formát PNC a v podobě seznamů jsou zastřešeny jednotným objektem třídy `PTnet`, jak ilustruje jednoduchý diagram na Obrázku 4.1.



Obrázek 4.1: Diagram závislosti základních tříd objektů pro P/T Petriho síť.

4.3 Strom dosažitelných značení

Reprezentace Na strom dosažitelnosti je kladeno hned několik požadavků. Každý uzel může mít $0 - n$ potomků, dále je nutno počítat se zpětnými odkazy pro možnost tzv. *backtrackingu*[1] a samozřejmě ukládat značení sítě a přechody, k nimž došlo. Konkrétní návrh složek jednoho uzlu stromu dosažitelnosti pak vypadá následovně:

- type – typ uzlu (viz kapitola 3.2)
- parent – odkaz na bezprostřední rodičovský uzel
- marking – značení sítě v tomto uzlu
- children – seznam potomků obsahující jak odkazy na synovské uzly, tak identifikaci k nim provedených přechodů

4.3.1 Vizualizace

Stavový prostor popsaný stromem dosažitelnosti lze také stromově zobrazit. Existuje množství volně dostupných nástrojů nabízejících vizualizaci grafových a stromových struktur a s nimi i řada různých formátů reprezentace. Pro účely této práce byl vybrán formát DOT[2] pro svou jednoduchost a rošířenost. Jedním z nástrojů, který jej podporuje, je GraphViz[5], se kterým budeme pracovat dále. (Jako webovou alternativu můžeme uvést např. Erdos[4].)

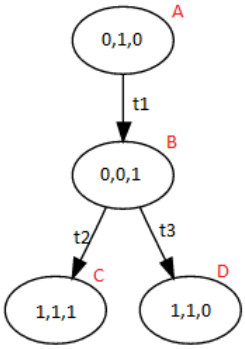
Formát DOT Na obrázku 4.2 je ukázka jednoduchého stomu značení popsaného tímto jednoduchým jazykem. Formát DOT nabízí také rozšířené možnosti, včetně podpory různých typů spojnic, grafického formátování a další. Pro účely našeho nástroje si však vystačíme se základním nastavením.

4.4 Propojení komponent

Na základě definované funkcionality a nastínění použitých prostředků byly navrženy tři základní komponenty:

1. komponenta XML2PNML – pro zpracování PIPE2 formátu Petriho sítě

[1] backtracking – zpětné navracení při průchodu stromovou strukturou



(a) Vizualizace stromu nástrojem GraphViz. Značky A, B, C, D byly přidány dodatečně pro další odkaz na konkrétní uzly.

```
1  digraph MyGraph{
2      "0,1,0" -> "0,0,1" [label="t1"];
3      "0,0,1" -> "1,1,1" [label="t2"];
4      "0,0,1" -> "1,1,0" [label="t3"];
5  }
```

(b) Odpovídající zápis v jazyce DOT.

Obrázek 4.2: Reprezentace stromu dosažitelnosti sítě s třemi místy.

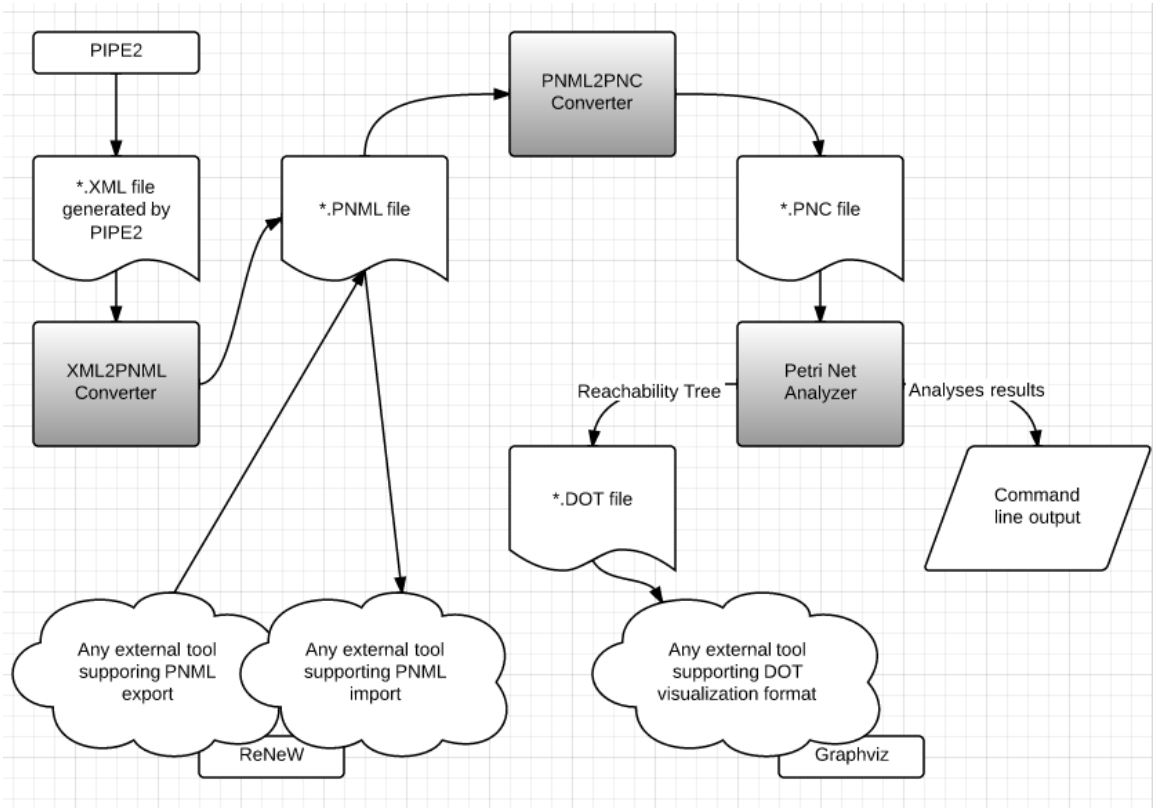
2. komponenta PNML2PNC – pro jednotné zpracování obecného PNML formátu
3. komponenta PNA – pro analýzu sítě

XML2PNML konvertor má za úkol zpracování formátu XML, jenž je výstupním exportem nástroje PIPE2. Konvertor převádí tuto konkrétní reprezentaci do standardizovaného PNML formátu a rozšiřuje tak možnosti programu PIPE2.

PNML2PNC konvertor má za úkol zpracování sítí ve standardizovaném PNML formátu. Neomezuje se pouze na výstup předchozího konvertoru, ale zajišťuje všeobecnou podporu PT-net balíčku jádra modelu PNML. Převádí síť z obecného PNML do zjednodušeného formátu PNC, který je dále použit pro analýzu sítě.

PNA (Petri Net Analyzer) – Analyzátor Petriho sítí je považován za výpočení jádro této sady nástrojů. Zpracovává P/T sítě v předpřipraveném formátu PNC a nabízí algoritmickou funkcionalitu pro analýzu těchto sítí. Mimo jiné je schopen zkonstruovat strom dosažitelnosti a reprezentovat v přenositelném formátu DOT.

Na Obrázku 4.3 je znárorněn diagram propojení těchto tří komponent (šedě zbarvené), vstupních/výstupních formátů a také rozšířená návaznost na jiné nástroje. Tato sbírka nástrojů bude dále odkazována jako *PNA toolbox* (Petri Net Analyzer toolbox – sbírka nástrojů pro analýzu Petriho sítí) a bude tvořit jednotné rozhraní pro dílčí nástroje.



Obrázek 4.3: Diagram propojení komponent sbírky nástrojů PNA toolbox (šedě zbarvené).

Kapitola 5

Implementace

Tato kapitola popisuje implementaci nástroje PNA toolbox popsaného v Kapitole 4. Stručně představuje implementační prostředky a pak dále implementaci jednotlivých částí. Na závěr je uveden popis rozhraní.

5.1 Implementační prostředky

Při návrhu bude využito mých vlastních zdrojových souborů použitých v rámci projektu ICP12[14]. Použita bude hlavně implementace tříd Petriho sítě a jejích prvků a dále některé pomocné funkce.

Pro implementaci algoritmů a zpracování sítě jakožto objektu byl vybrán objektový jazyk C++98 pro svou rozšířenost, čitelnost a vhodnost pro algoritmizaci. Pro práci se soubory, konverzi mezi formáty a tvorbu rozhraní bude použit jazyk Python verze 3. Ten zaručuje jednoduchost implementace základních systémových funkcí a nabízí vestavěné rozhraní pro práci s formáty typu XML. Použitím zmíněných technologií je zaručena multiplatformní přenositelnost vyvíjených nástrojů a možné širší využití.

5.2 XML2PNML

Pro načtení a zpracování sítě v XML a dále pro stavění výstupního formátu, jenž je také založen na XML formalismu, využívá tento nástroj rozhraní `xml.etree.cElementTree`. Pro vytvoření validního PNML je třeba vzít v úvahu standard PNML [6] a na základě analýzy PIPE2 formátu XML (vstupní a výstupní formáty jsou si dost podobné) provést tyto adaptační kroky:

1. Párové XML elementy `<value>` konvertovat na standardní elementy `<text>`,
2. Popis defaultních hodnot `"Default"` nahradit konkrétní číselnou hodnotou,
3. Grafické informace přesunout dovnitř elementů `<graphics>` k tomuto účelu určených a
4. Upravit hlavičku XML.

Toto je provedeno strukturálním procházením XML uzlů a tvorbou nového, upraveného XML stromu. PNML výstup je pak možno použít v jakémkoliv jiném nástroji, který tento

formát podporuje. Je ale třeba mít na paměti, že informace o síti byly exportovány ve formátu specifickém pro PIPE2. Některé informace – zejména grafické – tedy mohou být v jiném nástroji zkreslené. Plná přenositelnost informací nástroje PIPE2 na jiné nástroje podporující standard PNML však není smyslem této práce a tak je tato částečná ztráta informace považována za přijatelnou. Základní charakteristika sítě, včetně značení míst, hran a popisků jednotlivých elementů sítě však zůstává zachována.

5.3 PNML2PNC

Minimalistický formát PNC a jeho využití bylo představeno v kapitole 4.1. Následuje seznam informací, které je třeba o síti znát pro její budoucí analýzu a jak tyto informace z PNML souboru dostat:

- Identifikace všech míst, přechodů a hran mezi těmito prvky. V PNML struktuře, uvnitř elementu `<net>`[1] jsou umístěny všechny hlavní prvky sítě; místa jsou reprezentována elementem `<place>`, přechody elementem `<transition>`, hrany elementem `<arc>`.
- Každý z těchto prvků musí mít své unikátní ID. Netřeba žádné generovat, jelikož stejný požadavek klade PNML standard a ID je uvedeno u každého elementu jako tag `id`.
- Místa a přechody mohou mít uživatelem zadané jméno, odlišné od ID. Přestože tato informace není důležitá pro analýzu stavového prostoru, je důležitá pro následnou prezentaci výsledků. Pro snazší identifikaci elementů jsou použita uživatelem zadaná jména, přestože algoritmy PNA analyzátoru pracují pouze s atributem ID. Pokud jméno není zadané, je substituováno hodnotou ID.
- U míst je třeba zjistit počáteční značení. Tato informace bývá uložena v obecném elementu `<initialMarking>`, jenž obsahuje subelement `<text>` s danou hodnotou. Tato informace není povinná a v případě že chybí je použito defaultní značení $x = 0$.
- U hran je třeba zjistit jejich váha. Tato informace bývá uložena v obecném elementu `<inscription>`, jenž obsahuje subelement `<text>` s danou hodnotou. Tato informace není povinná a v případě že chybí je použita defaultní váha $w = 1$.
- PNML obsahuje také jiné informace o síti, jako její název, grafické informace jednotlivých elementů, a další. Pro účely PNC formátu jsou tyto ignorovány.

Formát PNC, přestože je velmi minimalistický, neklade žádná omezení na formát značení míst, ohodnocení hran a jiné (PNML samotné je velmi volně definováno a jádro jeho modelu tyto restrikce neklade). Jiné typy Petriho sítí než P/T mohou obsahovat v elementu pro značení místa a elementu pro váhu hrany jakoukoliv hodnotu, i textovou. Konvertor PNML2PNC se tedy neomezuje pouze na P/T Petriho sítě, musíme však počítat s tím, že PNA analyzátor pracuje pouze nad nimi a jiné nebudou úspěšně analyzovány.

[1] Většina nástrojů řídících se standardem podporují také stránkování sítí a tzv. *zástupná místa/přechody*. U nich je uvnitř elementu `<net>` ještě jeden nebo více elementů `<page>`. Konvertor PNML2PNC bere v úvahu tento element, avšak podporuje pouze klasická místa a přechody. Je-li tedy síť stránkována, načte pouze první stránku. Pro více informací o stránkování sítí viz standard PNML[6].

Integrace sítí ReNeW Jak bylo zmíněno v kapitole 3.3.4, ReNeW pracuje s referenčními sítěmi a také je v tomto formátu ukládá. Kromě podpory PT-net balíčku je tedy implementován parametr `--pnmltype-renew`, který zpracuje síť definovanou balíčkem pro referenční síťě a snaží se ji interpretovat jako P/T síť. Zjednodušeně pouze hledá sekvenci `"[]"` v elementu `<text>` uvnitř elementu `<initialMarking>` a prohlásí ji za černobílou[2] značku v daném místě. Jiná konverze prováděna není a v případě, že PNML exportované z ReNeW obsahuje jiné neočekávané prvky referenčních sítí, je síť prohlášena za nevalidní P/T síť.

5.4 Analyzátor

PNA analyzátor tvoří algoritmické jádro celké sbírky nástrojů PNA toolbox. Komplexní dokumentace k jednotlivým třídám a funkcím je poskytunuta v programové části této práce za pomoci generátoru Doxygen [3]. V dalším textu jsou popsány alespoň hlavní části programu.

Jádrem PNA je funkce `main`, která po zpracování a validaci všech argumentů volá funkci `processFile` s cestou ke zpracovávanému souboru; při zpracování více souborů je tato funkce volána v cyklu. Argumenty, se kterými je program spuštěn, jsou dostupné na globální úrovni napříč celým programem a podle nich je určeno chování analyzátoru. Chybové stavy jsou jednotně ošetřeny formátovanými zprávami za použití implementovaných funkcí `Error` a `Warning`.[3]

Prvotním krokem je načtení sítě z formátu PNC. Toto je zajišťeno funkcí `build_net`, která vrací nově vytvořený objekt sítě třídy `PTnet`. Spolu s metodou `validate`, která kontroluje vlastnosti specifické pro P/T sítě, zajišťují zpracování a validaci vstupů.

5.4.1 Konstrukce stromu dosažitelnosti

Při spuštění s argumentem `--reachtree-generate` je volána metoda `constructReachabilityTree`. Ta má za úkol vytvořit strom dosažitelných značení s kořenovým uzlem reprezentujícím počáteční značení. Konstrukce stromu se řídí algoritmem popsaným v Kapitole 3.2. Výčet `NodeType` pro tento účel definuje 5 typů uzlů:

`NODE_UNDEF`	– nedefinovaný typ uzlu	(5.1)
`NODE_HEAD`	– čelní uzel, k dalšímu zpracování	(5.2)
`NODE_END`	– koncový uzel, neduplikátní	(5.3)
`NODE_DUPLICATE`	– koncový uzel, duplikátní	(5.4)
`NODE_INNER`	– uzel uvnitř stromu	(5.5)

Pro možnost generování stromu do přílišné hloubky je zaveden systémový limit pro hloubku generovaného stromu a také možnost tento limit měnit pomocí parametru `--depth-limit`.

Tato metoda využívá hned několik dalších důležitých metod:

[2] *Černobílou značkou* je myšlena klasická P/T značka.

[3] Rozdíl mezi nimi je pouze ten, že Warning neukončí běh programu.)

- `fireTransitionWithoutNetUpdate` – metoda volaná pro daný objekt třídy PTnet, danou síť nastaví podle zadaného značení (které považuje za počátační) a provede zadaný aktivní přechod. Vrací výsledné značení sítě.

- `getPathFromRoot` – metoda volaná pro daný objekt třídy `ReachabilityTreeNode`, zpětně projde rozgenerovaný strom dosažitelných značení a v objektu třídy `ReachabilityTreePath` vrátí seznam přechodů, jež bylo nutno provést na cestě od počátečního značení.

- `transitionActive` – metoda volaná v dané síti s parametrem konkrétního přechodu. Analyzuje vstupy daného přechodu a informuje, zda daný přechod může být proveden. Spolu s metodami `updateCurrentMarking` a `markActiveTransitions` reflektují změny značení v celé síti.

Export do DOT Strom dosažitelnosti je automaticky exportován v DOT formátu do souboru s příponou `*.dot` a může být zobrazen programem GraphViz.

5.4.2 Dosažitelná značení

Při spuštění s argumentem `--query-marking` vygeneruje analyzátor strom dosažitelnosti a prohledá jej na zadané značení. Program také bere v potaz úmyslné ignorování konkrétních míst za pomoci zástupného symbolu "x".

V kapitole 3.2.1 bylo uvedeno omezení použitelnosti stromu dosažitelných značení na prohledání shody. Při implementaci metodou `searchTreeForMarking` byly použity následující implikace.

1. Pokud kterýkoliv uzel stromu přesně odpovídá hledanému značení, nebo hledané značení obsahuje zástupné symboly v místech kde tomu tak není, je uzel prohlášen za shodu a je zahlášena *přímá dosažitelnost hledaného značení.*

2. Pokud není nalezena přímá dosažitelnost, a značení ve stromu dosažitelnosti obsahuje symboly ω v místech, kde nedošlo ke shodě popsané předchozím bodem, je uzel prohlášen za možnou shodu a je zahlášena *možná (potenciální) dosažitelnost hledaného značení.*

3. Pokud nedojde ani k přímé ani možné shodě popsané předchozími dvěma body, je hledané značení bezpečně prohlášeno za *nedosažitelné značení.*

Příklad: Za předpokladu, že pracujeme se sítí, která má strom dosažitelnosti podle Obrázku 4.2, tak:

1. Pokud ...

   (a) se dotážeme na značení $0, 0, 1$, je nalezena právě jedna shoda a značení je prohlášeno za *přímo dosažitelné* v uzlu B.

   (b) se dotážeme na značení $0, x, 0$, jsou nalezeny dvě shody a značení je prohlášeno za *přímo dosažitelné* v uzlech A a D.

2. Pokud by uzel D obsahoval místo značení $1, 1, 0$ značení $1, 1, w$ a my se dotážeme na značení $1, 1, 5$, bude toto značení *možná dosažitelné* v uzlu D.

3. Pokud se dotážeme na značení $3, 2, 1$, bude toto značení bezpečně prohlášeno za *nedosažitelné*.

- Při zadání přepínače `--marking-path` je také vypsána sekvence přechodů z počátečního značení k nalezenému(ým) značení(m).
- Při zadání přepínače `--enabled-transitions` jsou také vypsány aktivní přechody z nalezeného(ých) značení.

5.4.3 Omezenost a bezpečnost sítě

Po vygenerování stromu dosažitelných značení je analýza sítě na omezenost (spuštění s argumentem `--is-bounded`), potažmo bezpečnost (spuštění s argumentem `--is-safe`) velmi jednoduchá. Absence zástupného symbolu ω ve stromu dosažitelnosti je dostačující podmínkou pro omezenost sítě. Jak moc je síť omezená je počítáno pro každé místo zvlášť, a to výběrem nejvyššího značení v daném místě. Celková omezenost sítě se pak rovná maximu z těchto dílčích hodnot. Pokud je celková celková omezenost $k = 1$, síť je bezpečná.

Při dosažení maximálního limitu hloubky při generování stromu dosažitelných značení je síť prohlášena za *nejspíše neomezenou*, přestože to nemusí být potvrzeno přítomností zástupného symbolu ω v tomto stromu. Předpokládá se, že značení buď není dostupné nebo je příliš vzdálené od značení počátečního.

5.4.4 Konzervativnost

Zachování počtu značek v síti provedením jakéhokoliv přechodu v jakékoliv fázi je kontrolováno analýzou konzervativnosti. Zda-li je síť konzervativní, je zjištěno přepínačem `--is-conservative`. Ten pro zjednodušení nepracuje se stromem dosažitelnosti, ale převede ho na množinu unikátních dosažitelných značení, nad níž jsou prováděny součty značek.

5.4.5 Detekce uváznutí

Absence uváznutí je další důležitou součástí mnoha systémů. Při spuštění s argumentem `--deadlocks` je analyzován strom dosažitelných značení a všechny uzly bez potomků, které jsou označeny jako koncové, a nejsou duplikátní, jsou prohlášeny za *deadlock*. Funkce tedy není zaměřena na identifikaci stupně živosti sítě či jednotlivých míst, místo toho se zaměřuje na detekci možných uváznutí a jejich předcházení výpisem sekvence přechodů jimiž k nim došlo.

Stav, kdy nejsou nalezena žádná uváznutí však neznamená, že v síti nemohou nastat. Zástupné symboly ve stromu dosažitelnosti nedávají totiž žádnou informaci o tom, která konkrétní značení jsou opravdu dosažitelná, a tudíž ani o aktivních přechodech z takových značení.

5.5 Popis rozhraní a případy užití

V této kapitole byla popsána implementace konvertorů XML2PNML, PNML2PNC a analyzátoru PNA. Ty mohou být použity buď samostatně nebo skrz jednotné rozhraní skriptu `PNA_toolbox`. Toto rozhraní konzoliduje veškerou funkcionalitu popsaných nástrojů. Dále

nabízí dávkové zpracování souborů uvnitř daného adresáře (přepínač `--dir`) – a to i rekurzivně (přepínač `--recursive`) nebo načtení různých typů přípon souborů (přepínač `--extension`). Komplexní nápověda může být získána příkazem `pna_toolbox.py --help`.

Výstupy Výstupy jednotlivých analýz jsou vypisovány na standardní výstup. Chyby a varování jsou zapsány na standardní chybový výstup. Konverze mezi formáty probíhá manuálním spuštěním nebo automaticky a výstupy těchto konverzí jsou tvořeny v původním adresáři pod původním jménem a za použití standardních přípon[4]. Více informací o spuštění, funkcích a vstupech/výstupech je možno získat výpisem nápovědy či přečtením souboru *README.txt*. Seznam všech parametrů nástroje PNA toolbox je také dostupná v Příloze B.

Případy užití V příloze C lze nalézt jak konkrétní ukázku při převodu formátů XML, PNML, PNC, tak případy užití analyzátoru včetně generování stromu dosažitelnosti do formátu DOT. Dále ukázky kombinací konkrétních přepínačů, vstupní sítě a popisy jednotlivých výstupů při analýze. Ilustrované případy užití mohou být otestovány na sadě testovacích sítí, která je součástí programové části této práce, a libovolně kombinovány jak na zmíněných sítích, tak na sítích vytvořených jakýmkoliv validním způsobem popsaným v této práci (viz diagram na Obrázku 4.3).

[4]Standardními příponami se myslí .pnml, případně .pnc pro generované PNML a PNC sítě.

Kapitola 6

Závěr

V této práci byly diskutovány Petriho sítě a jejich využití při modelování systémů. Rozdělením na podtřídy a shrnutím poznatků o jejich modelovací a rozhodovací mocnosti bylo rozhodnuto o abstrakci nad konkrétní podmnožinou Petriho sítí, značenými grafy, a dále byly brány v úvahu pouze obecné P/T Petriho sítě. Kapitola 3 měla za úkol shrnout jejich současné využití a identifikovat možná vylepšení. V návaznosti na to byl v Kapitole 4 představen nástroj PNA toolbox pro analýzu P/T Petriho sítí. Byla zdůvodněna návaznost na program PIPE2, částečná podpora referenčních sítí programu ReNeW a představen zjednodušený PNC formát sítě. V Kapitole 5 byla následně popsána implementace jak jednotlivých částí PNA toolboxu, tak jednotného rozhraní. Byla vytvořena sbírka sítí, nad kterou proběhlo testování. Příloha C obsahuje konkrétní případy užití nástroje PNA toolbox, přičemž je zaměřena na funkce analýzy. Popsány jsou vstupní sítě a výstupy testovaných příkazů, jednotlivé případy užití jsou pak doplněny o další poznámky, včetně případných omezení.

Řada problémů řešitelných analýzou modelu Petriho sítě je řešitelná analýzou dosažitelných značení. Posun v oblasti modelování Petriho sítěmi je ale brzděn hlavně nejednoznačností stromu dosažitelnosti ke konkrétní síti, což způsobuje obecnou nerozhodnutelnost problému dosažitelnosti, živosti sítě a otázek, které se o tyto analýzy opírají. Testováním byla tato omezení potvrzena. Značené grafy by měly být jednodušším a vhodnějším prostředkem pro analýzu vlastností. Na druhou stranu, řadu systémů modelovaných obecnou Petriho sítí na ně nelze převést. Pro analýzu takových sítí je tedy PNA toolbox vhodným prostředkem, spokojíme-li se s omezeným použitím pro analýzu živosti (uváznutí) a dosažitelnosti. Problémy omezenosti, bezpečnosti a konzervativnosti jsou řešeny jednoznačně, stejně tak jsou součástí nástroje generování množiny a možnost vizualizace stromu dosažitelnosti se zástupnými symboly ω. Zpracovány jsou jakékoliv validní P/T sítě ve standardizovaném formátu nebo modelované nástrojem PIPE2. V tom také můžeme také některé z funkcí analýzy ověřit, případně rozšířit.

Jako další krok by mohlo být obohacení nástroje PNA toolbox o další funkce analýzy, např. diskutované stupně živosti nebo interaktivní rozhraní nad konkrétní sítí (zadávání změn značení, dotazování atd.). Hlavní oblastí výzkumu v oblasti ověřování vlastností modelů by ale mělo být hledání kompromisu mezi modelovací a rozhodovací mocností a identifikace nejvhodnějšího modelovacího prostředku. Možnost konverze obecné Petriho sítě na značený graf bez ztráty důležité informace by také byla velkým krokem kupředu v této oblasti. V neposlední řadě, větší rozšíření standardizovaného PNML formátu by také mohlo pomoci dalšímu vývoji přenositelností a kombinovatelností různých existujících nástrojů.

Literatura

[1] The ARP tool (Analisador de Redes de Petri) @ONLINE.
URL http://dainf.ct.utfpr.edu.br/~maziero/doku.php/software:arp_tool

[2] The DOT Language @ONLINE.
URL http://www.graphviz.org/doc/info/lang.html

[3] Doxygen - The official website @ONLINE.
URL http://www.stack.nl/~dimitri/doxygen/index.html

[4] Erdos - A simple javascript interface into the Google Chart Graphviz API @ONLINE.
URL http://sandbox.kidstrythisathome.com/erdos/

[5] Graphviz - Graph Visualization Software @ONLINE.
URL http://www.graphviz.org

[6] The Petri Net Markup Language Home Page @ONLINE.
URL http://www.pnml.org/

[7] The Petri Nets World - Coplete Overview of Petri Nets Tools Database @ONLINE.
URL http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/complete_db.html

[8] The Petri Nets World @ONLINE.
URL http://www.informatik.uni-hamburg.de/TGI/PetriNets/

[9] The Petri Nets World @ONLINE.
URL http://www.pnml.org/tools.php

[10] PIPE2: Platform Independent Petri net Editor 2 @ONLINE.
URL http://pipe2.sourceforge.net/

[11] Renew: The Reference New Workshop @ONLINE.
URL http://www.renew.de/

[12] Češka, M.: *Petriho sítě - Úvod do teorie a nástrojů pro aplikaci Petriho sítí*. Brno: Akademické nakladatelství CERM, 1994.

[13] Češka, M.; Marek, V.; Novosad, P.; aj.: *Petriho sítě, Studijní opora FIT VUT*. 2009.

[14] Horák, M.: Editor a simulátor vysokoúrovňových Petriho sítí, 2012, projekt pro účely kurzu ICP - Seminář C++, Fakulta Informačních Technologii VUT v Brně.

[15] Kindler, E.: PNML: Concept, Status and Future Directions. *Entwurf Komplexer Automatisierungssysteme (EKA)*, May 2006: s. 35–55.

[16] Kosarju, S.: Decidability of reachability in vector addition systems (preliminary version). San Francisco, California, USA: ACM, May 1982, s. 267–281.

[17] Kovár, M.: *Diskrétní matematika*. 2002-2003, skriptum k předmětu Diskrétní matematika pro 1. ročník na Fakultě informačních technologií VUT v Brně.

[18] Lambert, J. L.: A Structure to Decide Reachability in Petri Nets. *Theor. Comput. Sci.*, ročník 99, č. 1, Červen 1992: s. 79–104, ISSN 0304-3975, doi:10.1016/0304-3975(92)90173-D.
URL http://dx.doi.org/10.1016/0304-3975(92)90173-D

[19] Leroux, J.: The General Vector Addition System Reachability Problem by Presburger Inductive Invariants. In *LICS*, 2009, s. 4–13.

[20] Mayr, E. W.: An Algorithm for the General Petri Net Reachability Problem. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, STOC '81, New York, NY, USA: ACM, 1981, s. 238–246, doi:10.1145/800076.802477.
URL http://doi.acm.org/10.1145/800076.802477

[21] Murata, T.: Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, ročník 77, č. 4, April 1989: s. 541–580.

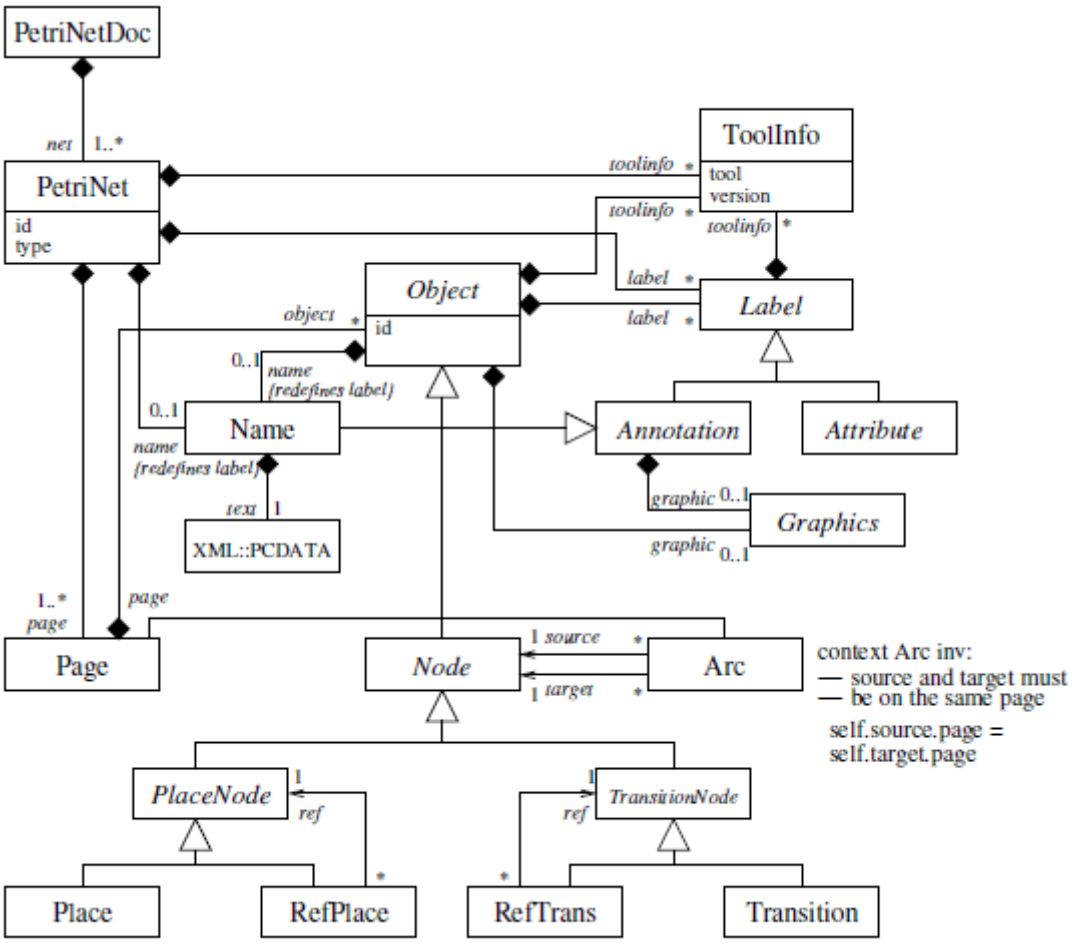
[22] Peterson, J. L.: Petri Nets. *ACM Comput. Surv.*, ročník 9, č. 3, Září 1977: s. 223–252, ISSN 0360-0300, doi:10.1145/356698.356702.
URL http://doi.acm.org/10.1145/356698.356702

[23] Petri, C. A.: *Kommunikation mit Automaten*. Dizertační práce, Universität Hamburg, 1962.

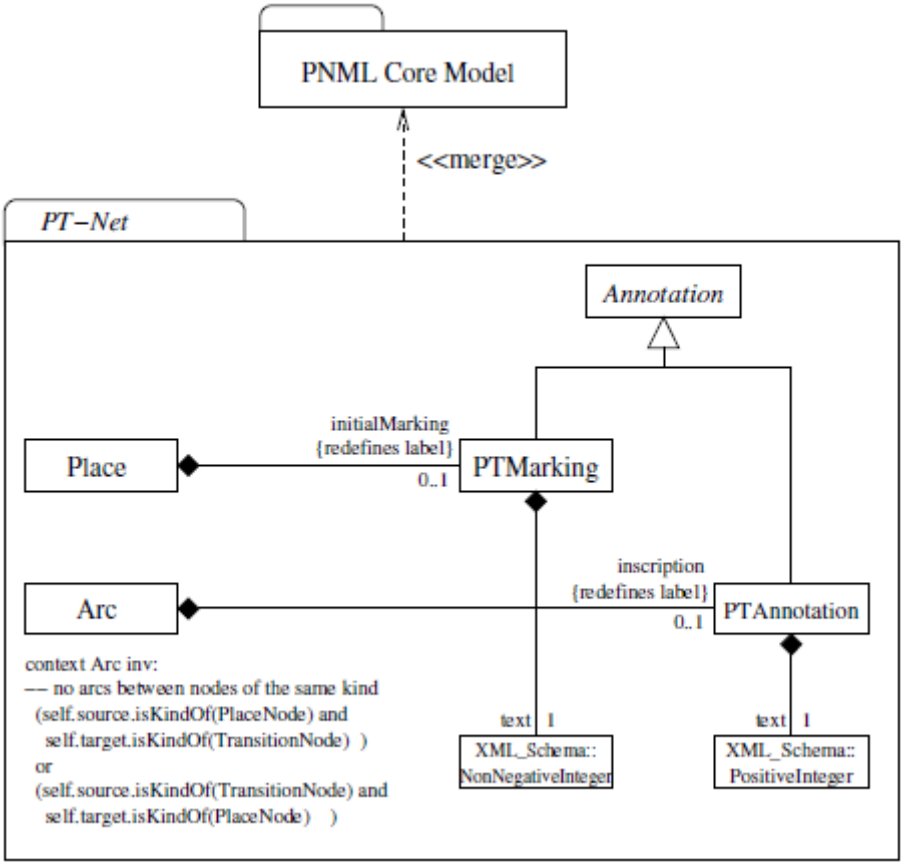
[24] Sacerdote, G. S.; Tenney, R. L.: The Decidability of the Reachability Problem for Vector Addition Systems (Preliminary Version). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, New York, NY, USA: ACM, 1977, s. 61–76, doi:10.1145/800105.803396.
URL http://doi.acm.org/10.1145/800105.803396

Příloha A

PNML



Obrázek A.1: Jádro modelu PNML.



Obrázek A.2: PT-net balíček modelu PNML.

Příloha B

PNA toolbox – parametry

`POUŽITÍ: <pna_toolbox.py> [OPTIONS] [target]`
`OPTIONS:` výběr konkrétního nástroje. Jsou navzájem výlučné:

`--xml2pnml-only`	Konvertuje sítě ve formátu XML uložené programem PIPE2 do standardního PNML formátu.
`--pnml2pnc-only`	Konvertuje P/T sítě ve standardním PNML formátu do minimalistického PNC formatu.
`--xml-analyze`	Analyzuje sítě ve formátu XML uložené programem PIPE2. Kromě analyzátoru pna spouští na pozadí konvertory xml2pnml a pnml2pnc.
`--pnml-analyze`	Analyzuje P/T sítě ve standardním PNML formáru. Kromě analyzátoru pna spouští na pozadí konvertor pnml2pnc.
`--pnc-analyze (defaultní)`	Analyzuje P/T sítě ve formátu PNC. Spouští analýzátor pna.

`OPTIONS:` vztahující se ke konverzi xml2pnml:
Pozn.: Musejí být použity v kombinaci s parametrem `--pnml2pnc-only` nebo `--pnml-analyze` (or `--xml-analyze`).

`--pnmltype-renew`	Vstupním formátem není klasické PT-net PNML ale referenční síť exportovaná z programu ReNeW.

`OPTIONS:` vztahující se k analýze pna:
Pozn.: Musejí být použity v kombinaci s parametrem `--pnc-analyze` (nebo `--pnml-analyze` nebo `--xml-analyze`).

`-b, --is-bounded`	Zjistí, zda je síť omezená a jak. Vypíše meze pro každé místo sítě.
`-c, --is-conservative`	Zjistí, zda je síť striktně konzervativní.
`-e, --transitions-enabled`	Kombinováno s parametrem `-q`. U nalezených značení vypíše všechny aktivní přechody.
`-m, --depth-limit=MAX`	Nastaví mezní hloubku, do které je generován strom dosažitelnosti.
`-f, --is-safe`	Zjistí, zda je síť bezpečná.
`-l, --deadlocks`	Zjistí, zda v síti může dojít k uvíznutí a při jakých značeních.
`-p, --marking-path`	Kombinováno s parametrem `-q`. U nalezených značení vypíše cestu od kořene stromu dosažitelných značení.
`-q, --query-marking=MARKING`	Zjisti, zda je dané značení MARKING dosažitelné. Pokud toto nelze jednoznačně určit ze stromu dosažitelnosti, upozorni na tuto skutečnost. Symbol x může sloužit jako substituce u míst, která nejsou analyzována. Příklad značení u sítě se 4 místy: MARKING=1,0,x,3.
`-s, --reachset-generate`	Vypiš všechna dosažitelná značení podle stormu dosažitelnosti.
`-t, --reachtree-generate`	Zkonstruuj strom dosažitelnosti a ulož jej v .dot formátu.

`OPTIONS:` obecné:

`-d, --dir=DIRECTORY`	Zpracuj všechny soubory v daném adresáři DIRECTORY místo práce s jedním souborem. Pozn.: Pouze soubory se standardními příponami (.xml, .pnml, .pnc) jsou zpracovány.
`-h, --help`	Vytiskni tuto nápovědu.
`-r, --recursive`	Kombinováno s parametrem -d. Zpracuj všechny soubory v adresáři a všech podadresářích, rekurzivně.
`-x, --extension=EXTENSION`	Kombinováno s parametrem -d. Zpracuj všechny soubory v adresáři ale pouze ty, dané příponou EXTENSION. Pozn.: Bez ohledu na příponu souboru musí být tyto soubory validní vůči požadované operaci. Přípona musí být zadána bez uvozovek, nezáleží na velikosti písmen.

Další poznámky:
Povinné argumenty dlouhých tvarů parametrů jsou povinné i u krátkých ekvivalentů.
`target` je jeden konkrétní soubor, validní vůči požadované operaci. Je ignorován, pokud je zadán parametr `-d`.

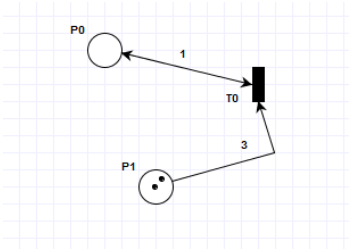
Příloha C

Případy užití

C.1 Případ užití č. 1

Popis: Konverze formátů XML $\longrightarrow$ PNML $\longrightarrow$ PNC

Vstup: Petriho síť net6.xml



Obrázek C.1: Jednoduchá Petriho síť obsahující 2 místa, 1 přechod a 3 hrany (hrana mezi $P0$ a $T0$ je obousměrná).

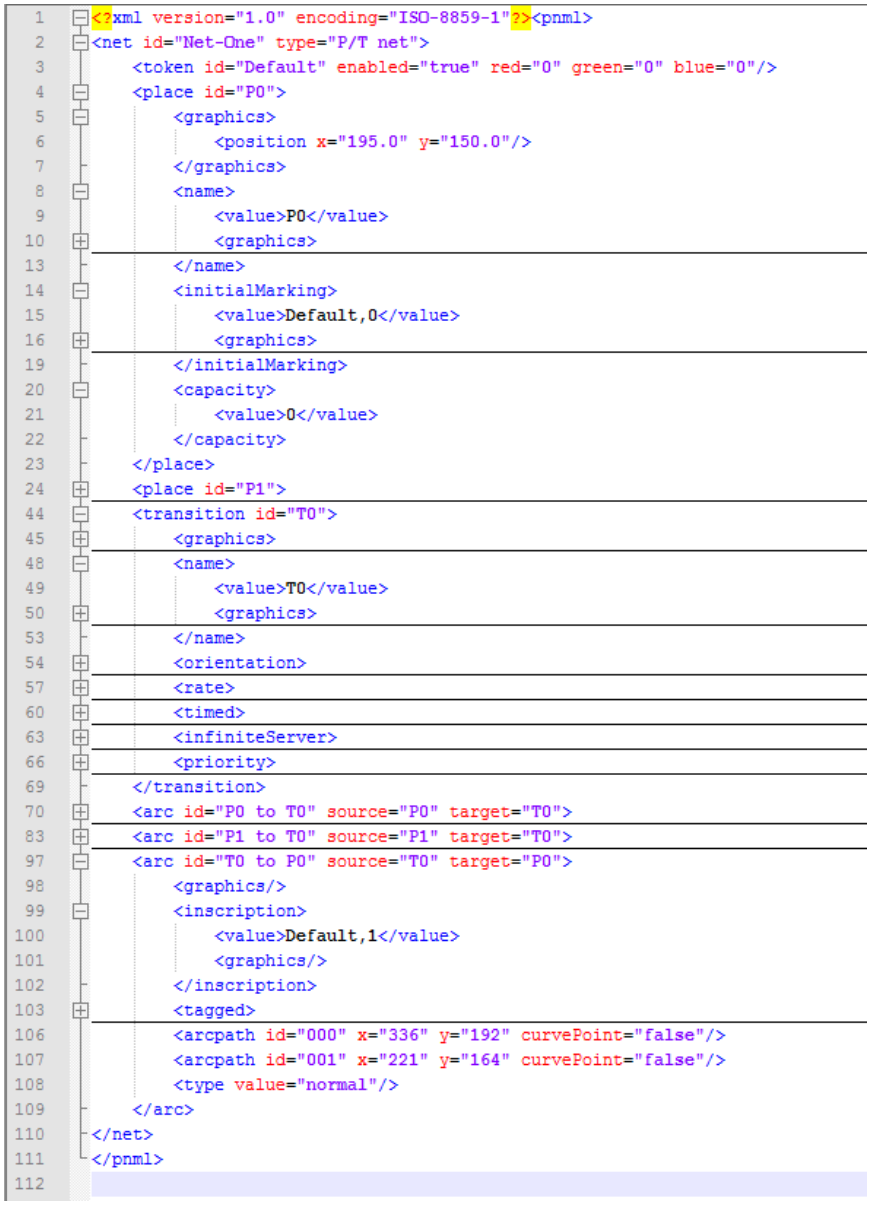
Spuštění:

```
pna_toolbox.py --xml2pnml examples/net6.xml
pna_toolbox.py --pnml2pnc examples/net6.pnml
```

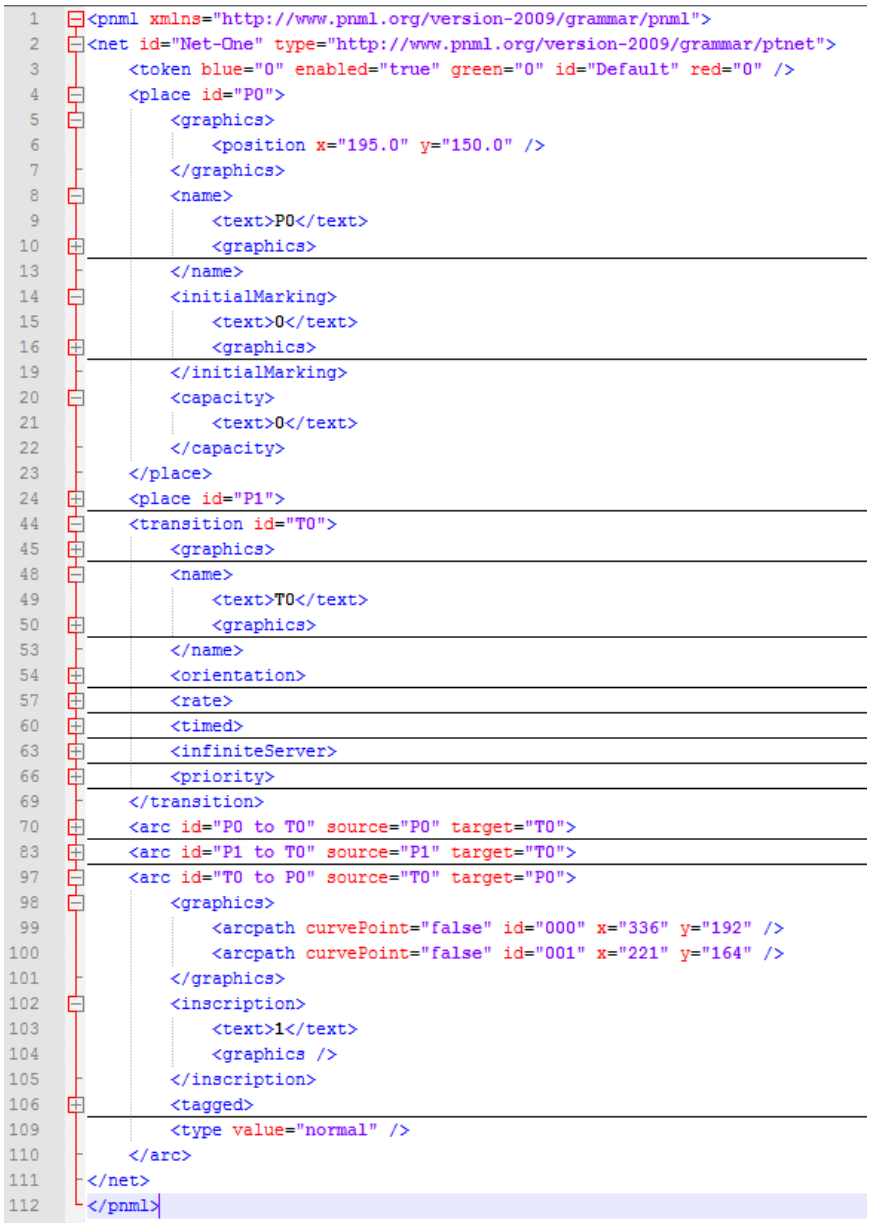
Výstupy:

- Obrázek C.2
- Obrázek C.3
- Obrázek C.4

Komentář: Síť po konverzi z XML do PNML se téměř nezměnila. Všimněme si ale změny umístění elementů `<arcpath>`, přejmenování elementu `<value>` na `<text>` a úpravy hodnot "Default". Minimalistický PNC formát pak již obsahuje pouze informace potřebné k analýze; stále se dají rozeznat jednotlivá místa, přechod a hrany.



Obrázek C.2: Původní XML formát generovaný programem PIPE2.



Obrázek C.3: Stejná síť konvertovaná do PNML.

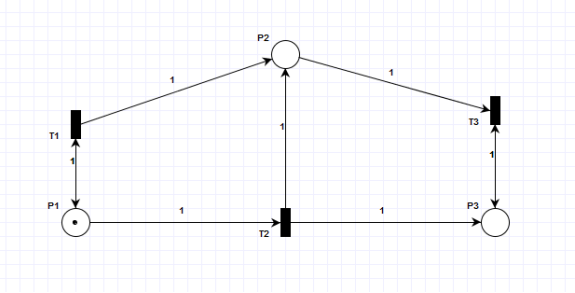
```
P;P0;P0;0;
P;P1;P1;2;
T;T0;T0;
A;P0 to T0;P0;T0;1;
A;P1 to T0;P1;T0;3;
A;T0 to P0;T0;P0;1;
```

Obrázek C.4: Převod do formátu PNC.

C.2 Případ užití č. 2

Popis: Generování stromu dosažitelných značení

Vstup: Petriho síť net1.xml



Obrázek C.5: Síť s 3 místy a 3 přechody.

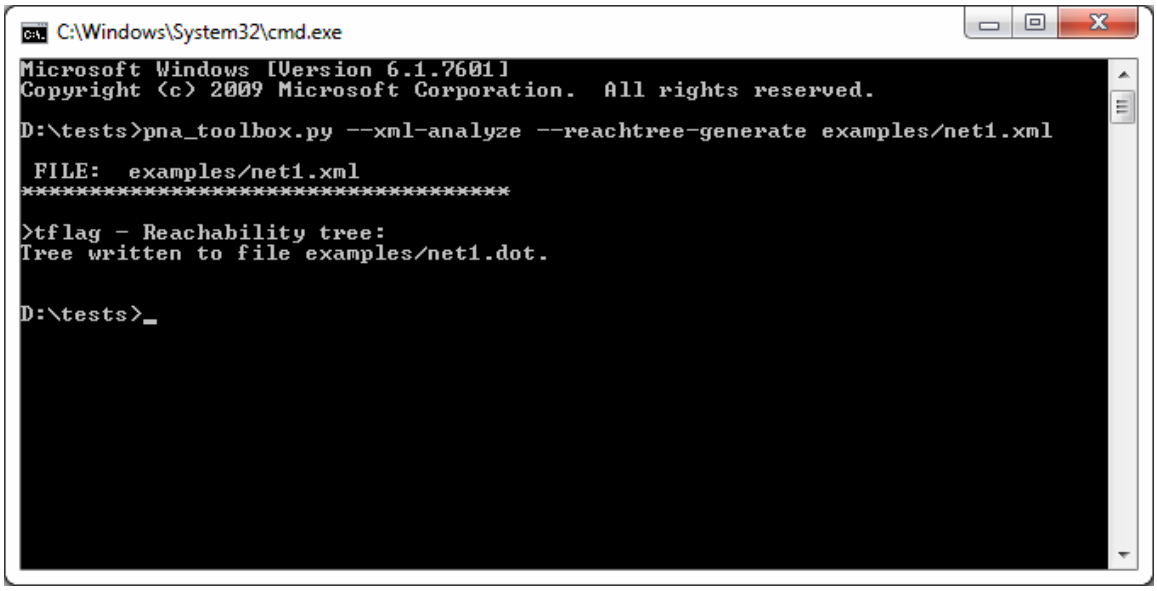
Spuštění:

```
pna_toolbox.py --xml-analyze --reachtree-generate examples/net1.xml
```

Výstupy:

- Obrázek C.6
- Obrázek C.7
- Obrázek C.8

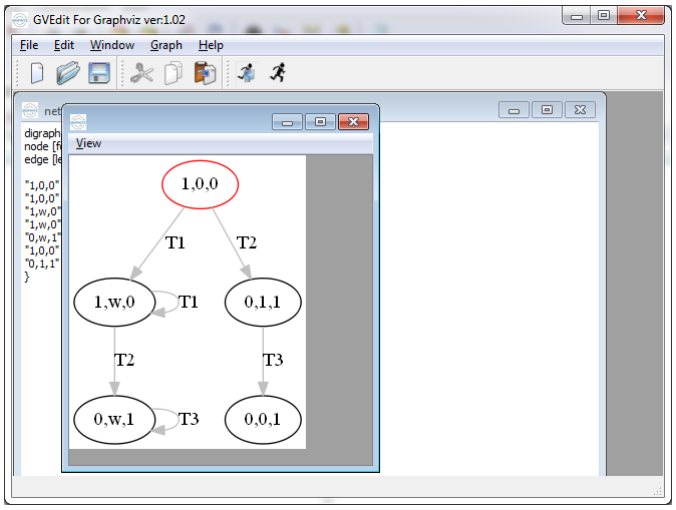
Komentář: Nejdříve byly provedeny konverze z XML do PNML a následně do PNC. Oba formáty jsou uloženy v původním adresáři pro další referenci.). Následně byl vygenerován strom dosažitelnosti a automaticky zapsán do souboru net1.dot. Vidíme, že přesto, že je stavový prostor nekonečný (obsahuje symboly w), je snadno reprezentován konečnou stromovou strukturou. Tento výstup může být zkontrolován oproti [12], str. 33.



Obrázek C.6: Strom dosažitelnosti byl vygenerován do souboru examples/net1.dot.

```
1  digraph name {
2  node [fontSize = 10]
3  edge [length=100, color=gray, fontColor=black]
4
5  "1,0,0" [color=red, fontColor=white]
6  "1,0,0" -> "1,w,0" [label=T1];
7  "1,w,0" -> "1,w,0" [label=T1];
8  "1,w,0" -> "0,w,1" [label=T2];
9  "0,w,1" -> "0,w,1" [label=T3];
10 "1,0,0" -> "0,1,1" [label=T2];
11 "0,1,1" -> "0,0,1" [label=T3];
12 }
13
```

Obrázek C.7: Vygenerovaný soubor net1.dot.



Obrázek C.8: Zobrazení souboru net1.dot programem GraphViz. Vidíme, že GraphViz vytvořil smyčku u uzlů $(1, w, 0)$ a $(0, w, 1)$ místo tvorby duplikátních potomků.

C.3 Případ užití č. 3

Popis: Výpis dosažitelných značení

Vstup: Petriho síť net1.pnc vytvořená předchozím případem užití

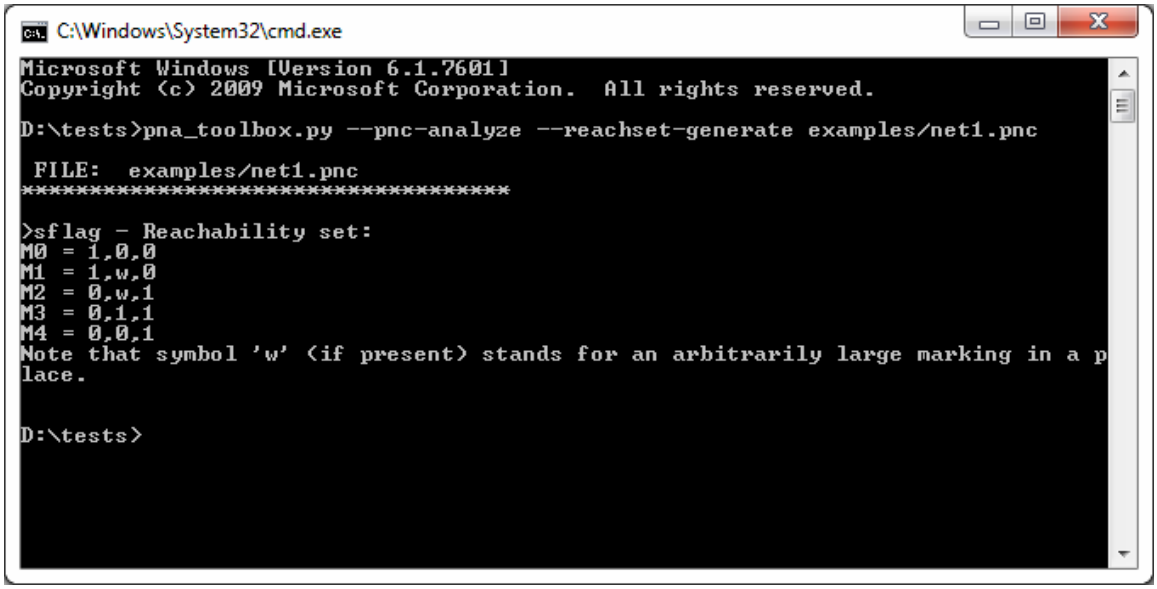
Spuštění:

```
pna_toolbox.py --pnc-analyze --reachset-generate examples/net1.pnc
```

Výstupy:

- Obrázek C.9

Komentář: Je vypsáno 5 unikátních uzlů ze stromu dosažitelných značení ilustrovaného Obrázkem C.8. Značení $M0$ reprezentuje počáteční značení, $M1, M2, \ldots M4$ pak další dosažitelná značení. Díky přítomnosti symbolu w u této sítě dochází ke vzájemnému pokrytí značení $M0$ značením $M1$ a značení $M3$ značením $M2$. Jelikož ale uzly tvoří samostatný neduplikátní uzel ve stromu dosažitelnosti s různými cestami od kořene, jsou považovány za rozdílná značení.

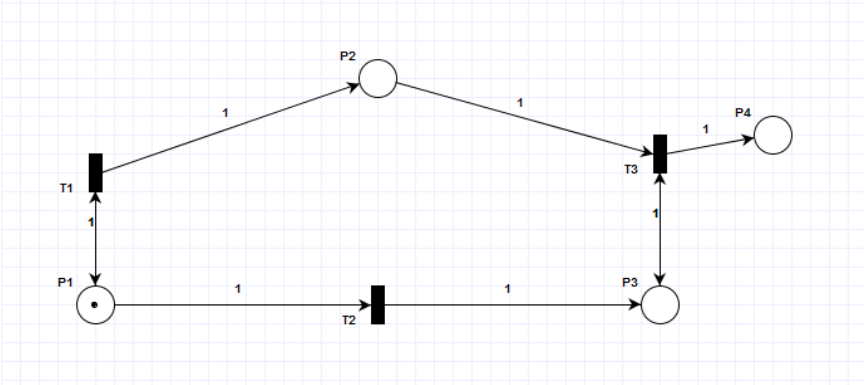


Obrázek C.9: Výpis dosažitelných značení pro síť net1.pnc.

C.4 Případ užití č. 4

Popis: Dosažitelnost

Vstup: Petriho síť net3.xml



Obrázek C.10: Síť s místy $P2$ a $P4$ potenciálně hromadícími značky.

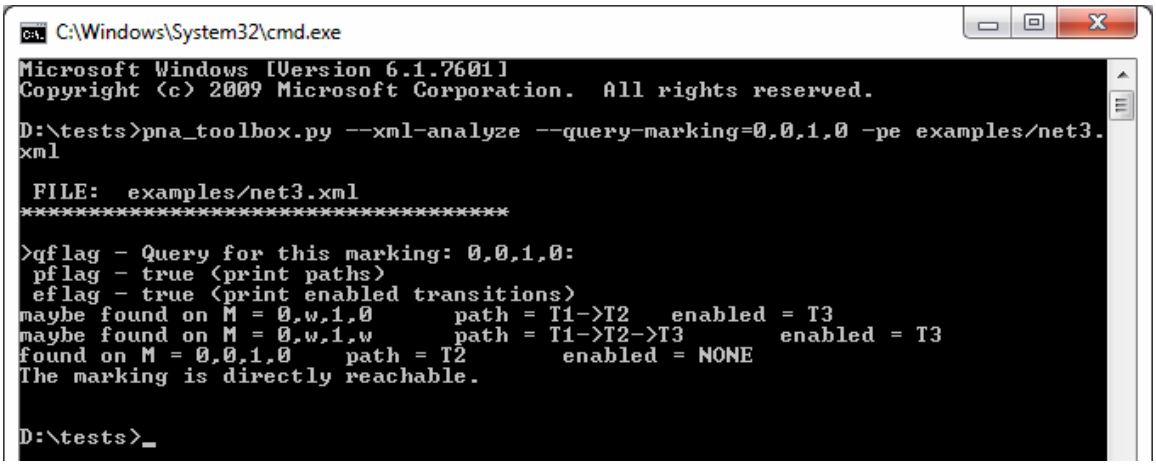
Spuštění:

```
pna_toolbox.py --xml-analyze --query-marking=0,0,1,0 -pe examples/net3.xml
pna_toolbox.py --xml-analyze --query-marking=1,6,0,x -pe examples/net3.xml
pna_toolbox.py --xml-analyze --query-marking=0,x,0,x -pe examples/net3.xml
```

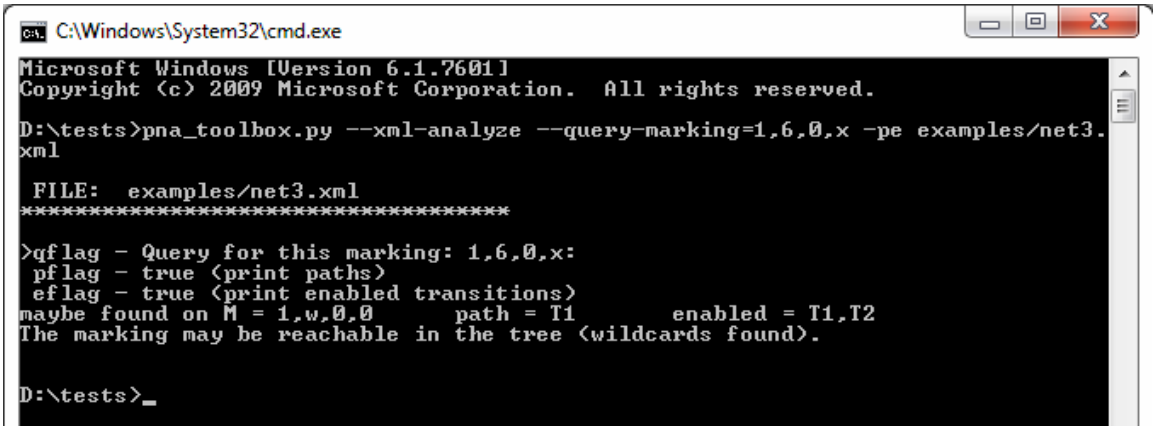
Výstupy:

- Obrázek C.11a
- Obrázek C.11b
- Obrázek C.11c

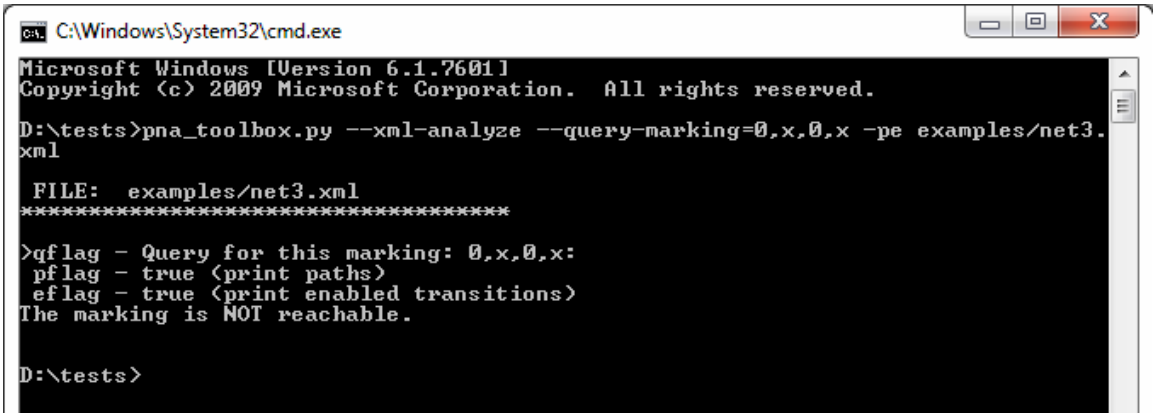
Komentář: Síť je analyzována na 3 různá značení, přičemž se zajímáme také o cesty ke značením (přepínač `-p`) a aktivní přechody v nich (přepínač `-e`). První značení je jednoznačně dosažitelné. Druhé značení je pokryto jiným značením, je tedy potenciálně dosažitelné (všimněme si také nejednoznačnosti výstupu parametru `-p`). Třetí značení je jednoznačně nedosažitelné.



(a) Značení 0,0,1,0 je jednoznačně dosažitelné, a to po provedení přechodu T2. Dále z něj ale nejsou žádné přechody aktivní.



(b) Značení $1, 6, 0, x$ (místo $P4$ nás nezajímá) nalezlo pokrytí v uzlu $1, w, 0, 0$. Kvůli nejednoznačnosti symbolu w je ale zahlášena pouze potenciální dosažitelnost. Z kořenu stromu dosažitelnosti do něj vede přechod $T1$ a dalšími aktivními přechody jsou $T1$ a $T2$. Všimněme si, že strom dosažitelnosti zanedbává počet nutných přechodů $T1$ do hledaného značení, provedením pouze jednoho by v místě $P2$ přibyla pouze jedna značka, přechod bychom tedy museli provést 6krát.



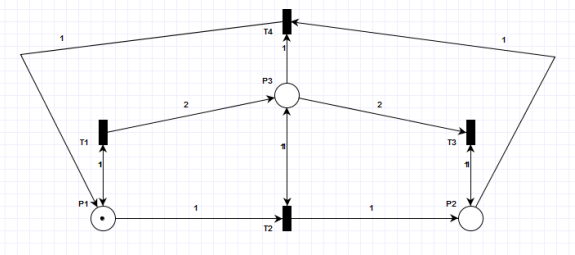
(c) Značení s místy $P1$ a $P3$ neobsahujícími žádnou značku je při zadaném počátečním značení v síti nedosažitelné.

Obrázek C.11: Dosažitelnost různých značení.

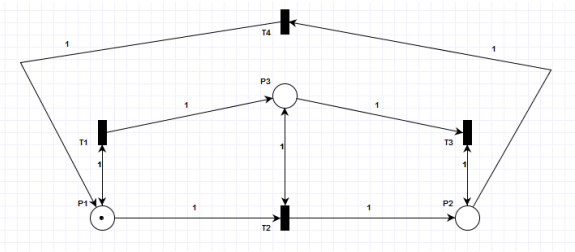
C.5 Případ užití č. 5

Popis: Dosažitelnost a uváznutí – nejednoznačnost

Vstup: Petriho sítě net4a.xml a net4b.xml



(a) Petriho síť $N1$.



(b) Petriho síť $N2$.

Obrázek C.12: Příklad nejednoznačnosti stromu dosažitelnosti převzatý z [12], str. 37.

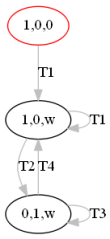
Spuštění:

```
pna_toolbox.py --xml-analyze --reachtree-generate examples/net4a.xml
pna_toolbox.py --xml-analyze --reachtree-generate examples/net4b.xml
pna_toolbox.py --xml-analyze -q 0,1,0 --deadlocks examples/net4a.xml
pna_toolbox.py --xml-analyze -q 0,1,0 --deadlocks examples/net4b.xml
```

Výstupy:

- Obrázek C.13a
- Obrázek C.13b
- Obrázek C.13c

Komentář: Síť $N1$ není živá. Např. posloupnost přechodů $T1T2T3$ vede ke značení $0, 1, 0$, pro který již žádný z přechodů není proveditelný. Síť $N2$ je živá, jelikož v ní toto konkrétní značení není dosažitelné, přestože strom dosažitelnosti obsahuje uzel $0, 1, w$. Proto dotaz na toto značení zahlásí pouze možnou shodu.



(a) Shodný strom dosažitelnosti.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

D:\tests>pna_toolbox.py --xml-analyze --query-marking=0,1,0 --deadlocks examples
/net4a.xml

 FILE:  examples/net4a.xml
**************************************

>qflag - Query for this marking: 0,1,0:
 pflag - false (do not print paths)
 eflag - false (do not print enabled transitions)
maybe found on M = 0,1,w
The marking may be reachable in the tree (wildcards found).

>lflag - Identify deadlocks:
No deadlocks found.


D:\tests>_
```

(b) Výstup pro síť $N1$ (net4a.xml).

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

D:\tests>pna_toolbox.py --xml-analyze --query-marking=0,1,0 --deadlocks examples
/net4b.xml

 FILE:  examples/net4b.xml
**************************************

>qflag - Query for this marking: 0,1,0:
 pflag - false (do not print paths)
 eflag - false (do not print enabled transitions)
maybe found on M = 0,1,w
The marking may be reachable in the tree (wildcards found).

>lflag - Identify deadlocks:
No deadlocks found.


D:\tests>_
```

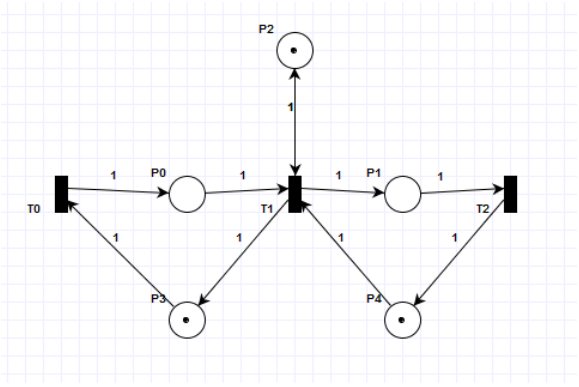
(c) Výstup pro síť $N2$ (net4b.xml).

Obrázek C.13: Shodné výstupy proobě sítě. Hledané značení 0,1,0 je prohlášeno za potenciálně dosažitelné v uzlu 0,1,w. Nebyly identifikovány žádné uzly jednoznačně způsobující uváznutí. Přesto síť $N1$, na rozdíl od sítě $N2$, značení $0, 1, 0$ opravdu obsahuje a způsobí v ní deadlock.

C.6 Případ užití č. 6

Popis: Omezenost, bezpečnonst, konzervativnost

Vstup: Petriho síť net8.xml



Obrázek C.14: Síť, která je bezpečná a konzervativní

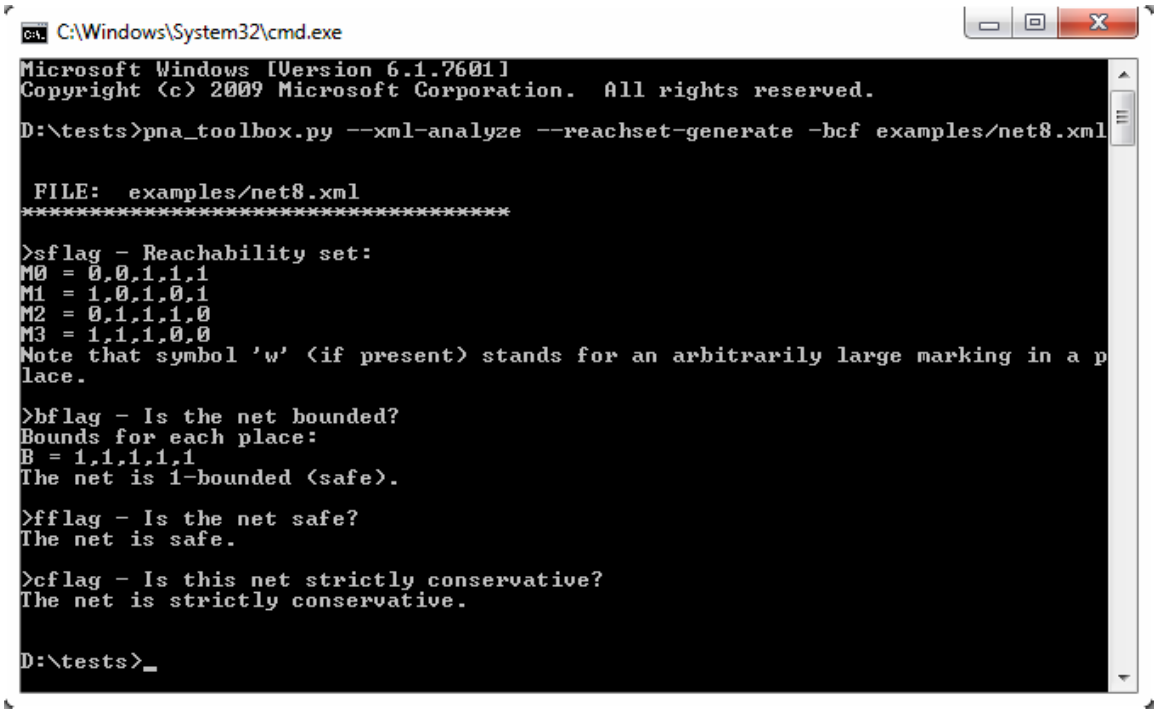
Spuštění:

```
pna_toolbox.py --xml-analyze --reachset-generate -bcf examples/net8.xml
```

Výstupy:

- Obrázek C.15

Komentář: Síť si při jakémkoliv přechodu zachovává celkový počet značek rovný 3 a v žádném místě nikdy nenabyde značení hodnoty vyšší než 1. Omezení jednotlivých míst je tedy stanoveno jako $B = 1, 1, 1, 1, 1$ a síť je prohlášena za bezpečnou a také striktně konzervativní.

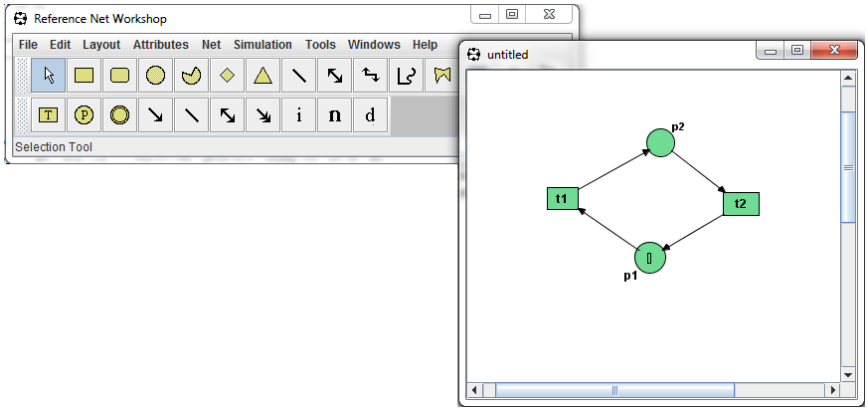


Obrázek C.15: Ověření vlastností sítě výpisem množiny dosažitelných značení a dotazem na omezenost, bezpečnost a konzervativnost.

C.7 Případ užití č. 7

Popis: ReNeW

Vstup: Petriho síť net5.xml



Obrázek C.16: Síť navržená v ReNeW. Místo p_1 obsahuje černou značku v podobě sekvence [].

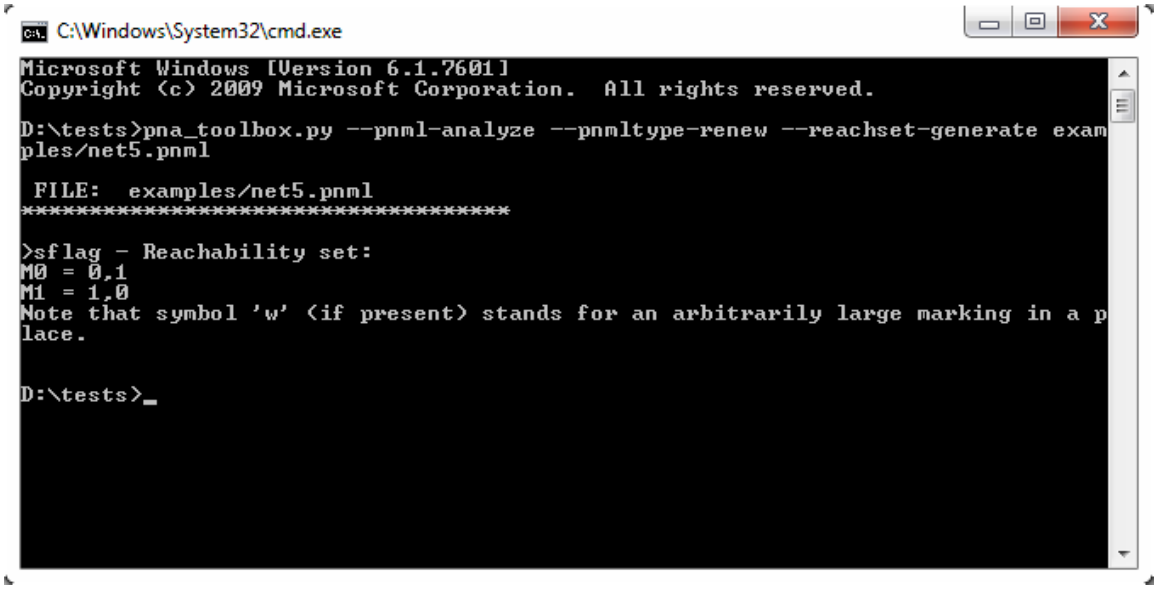
Spuštění:

```
pna_toolbox.py --pnml-analyze --pnmltype-renew --reachset-generate
examples/net5.pnml
```

Výstupy:

- Obrázek C.17

Komentář: Při vynechání přepínače `--pnmltype-renew` by proběhla konverze pnml2pnc a jako počáteční značení místa p_1 by byla zapsána sekvence []. Tu však analyzátor PNA zamítne jako nevalidní značení místa P/T sítě a soubor přeskočí.



Obrázek C.17: Síť má pouze dva stavy.