VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

PETRIHO SÍTĚ PRO EXPERTNÍ SYSTÉMY

PETRI NETS FOR EXPERT SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVEL MILLION
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. VÁCLAV JIRSÍK, CSc.
SUPERVISOR

BRNO 2010

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Pavel Million	***ID:***	85181
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Petriho sítě pro expertní systémy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je implementace a zhodnocení inferenčního mechanismu expertního systému využívajícího Petriho sítě.

Popište základní vlastnosti Petriho sítí a diskutujte možnost jejich využití pro expertní systémy. Navrhněte způsob konstrukce bází znalostí využívaiící barevné synchronizované Petriho sítě s prioritními přechody. Navrhněte algoritmus inferenčního mechanismu. Implementujte v C#. Demonstrujte funkčnost na vhodném příkladu. Diskutujte výsledek s ohledem na praktickou použitelnost pro expertní systémy.

DOPORUČENÁ LITERATURA:

Mařík, Štěpánková, Lažanský: Umělá inteligence, Academia 1997
David, Alla: Discrete, Continuous and Hybrid Petri nets, Springer 2005

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: doc. Ing. Václav Jirsík, CSc.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt:

Cílem této práce je popsat základní atributy expertního systému pomocí Petriho sítí. Pozornost je věnována zejména tvorbě znalostní báze, způsob uchovávání a reprezentace znalostí. Dále je zde popsán zásadní rozdíl při tvorbě báze znalostí pro diagnostický a plánovací expertní systém z hlediska struktury Petriho sítí. Jsou zde popsány podmínky použití Petriho sítí a způsob interpretace znalostí pro inferenční mechanismus jak u diagnostických tak u plánovacích expertních systémů. Rovněž je zde předvedeno použití Petriho sítí vyšší úrovně a jazyk popisující strukturu Petriho sítě.

Klíčová slova:

Petriho sítě, expertní systémy, báze znalostí, synchronizované, časované, prioritní, barvené a rozšířené Petřino sítě, inferenční mechanismus, princip plánování, diagnostika

Abstract:

Purpose of this master thesis is description of base parts of expert system with using Petri nets. Attention is mainly concentrate to knowledge base, way of storing knowledge. Next parts are describing main different between production base knowledge for planning or diagnostic expert system from Petri nets view. In this thesis conditions of using Petri nets and way of interpretation knowledge for inference mechanism in planning and diagnostic expert system are described. Using of high level Petri nets and language describing Petri nets structure and behaviour are demonstrated in next part of this thesis.

Keywords:

Petri nets, expert systems, knowledge base, synchronized, timed, coloured, priority and extended Petri nets, inference mechanism, planning principle, diagnostic

Bibliografická citace mé práce:

MILLION, P. *Petriho sítě pro expertní systémy.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 73 s. Vedoucí diplomové práce doc. Ing. Václav Jirsík, CSc.

Prohlášení o původnosti

„Prohlašuji, že svou diplomovou práci na téma „Petriho sítě pro Expertní systémy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne: **24. května 2010** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Doc. Ing. Václavu Jirsíkovi a konzultantovi diplomové práce Ing. Petru Poláchovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **24. května 2010** ……
podpis autora

Obsah :

1. ÚVOD 10
2. ÚVOD DO EXPERTNÍCH SYSTÉMŮ 11
2.1 Co jsou to expertní systémy 11
2.2 Charakteristické rysy expertních systémů 11
2.3 Struktura expertního systému 12
2.4 Aplikace expertního systému 12
2.5 Rozdíl mezi diagnostickým a plánovacím expertním systémem 13
3. PETRIHO SÍTĚ 14
3.1 Úvod do Petriho sítí 14
3.2 Struktura Petriho sítí 14
3.2.1 Definice sítě 15
3.3 Incidenční matice 17
3.4 Živá, omezená a bezpečná Petriho síť 18
3.4.1 Strom pokrytelnosti 18
3.4.2 Omezená Petriho síť 21
3.5 Jazyky Petriho sítí 21
3.5.1 Jazyk typu L 22
3.5.2 Sjednocení Petriho sítí 22
4. PETRIHO SÍTĚ VYŠŠÍ ÚROVNĚ A MOŽNOSTI JEJICH VYUŽITÍ V EXPERTNÍCH SYSTÉMECH 25
4.1 Petriho sítě s omezenou kapacitou (finite capacity petri nets) 25
4.2 Barvené Petriho sítě (Colored Petri nets) 26
4.3 Rozšířené Petriho sítě (extended petri nets) 29
4.4 Prioritní Petriho sítě (priority petri nets) 30
4.5 Synchronizované Petriho sítě (synchronized petri nets) 31
4.6 Časované Petriho sítě (timed petri nets) 33
5. PETRIHO SÍTĚ JAKO NÁSTROJ PRO TVORBU BÁZE ZNALOSTÍ. 35
5.1 Tvorba báze znalostí expertních systémů pomocí pn 35
5.2 Báze znalostí pro diagnostický Expertní systém 40
5.2.1 Diagnostický expertní systém řešící logickou hru 41

5.2.2 Řešení konfliktu odpalitelnosti 43
5.2.3 Řešení pomocí synchronizovaných PN 43
5.2.4 Báze znalostí 47
5.2.5 Realizace v objektově orientovaném jazyce 47
5.2.6 Jazyk typu L Petriho sítě popisující diagnostický expertní systém 51
5.2.7 Shrnutí použití dané PN 53
5.3 Báze znalostí pro plánovací expertní systém 54
5.3.1 Výběr vhodného pravidla 55
5.3.2 Detekce cílových cest 55
5.3.3 Řešení úlohy umělé inteligence plánovací metodou 55
5.3.4 Báze faktů 58
5.3.5 Báze znalostí 58
5.3.6 Incidenční matice 59
5.3.7 Realizace principu prohledávání stavového prostoru 61
5.3.8 Převod přechodu v bázi znalostí na prvek incidenční matice 63
5.3.9 Příklad pro 2 nádoby o objemech 3 a 5 l 64
5.4 Shrnutí použití Petriho sítí pro tvorbu báze znalostí expertního systému 68
6. ZÁVĚR 70
7. SEZNAM POUŽITÉ LITERATURY 72

1. ÚVOD

V první části této práce jsou popsány základní charakteristiky expertních systémů. Jde o teoretický úvod a popis těch částí, které budou v následujících kapitolách vytvořeny pomocí Petriho sítí.

Další část tvoří popis základních vlastností Petriho sítí. Budou zde vysvětleny charakteristické vlastnosti jako bezpečnost, omezenost, konzervativnost atd. Dále způsoby popisující strukturu a vzájemné vazby Petriho sítě jako strom pokrytelnosti, incidenční matice apod. Rovněž zde budou popsány tzv. Petriho sítě „vyšší úrovně“ jako jsou např. barvené, synchronizované nebo časované Petriho sítě.

Hlavní část této práce je praktické využití Petriho sítí při tvorbě hlavních částí expertního systému - báze faktů, báze znalostí a implementace příslušného inferenčního mechanismusu. Z hlediska inferenčního mechanismu bude demonstrován zásadní rozdíl při popisu znalostí u plánovacího a diagnostického expertního systému.

Závěr této práce je věnován zhodnocení dosažených poznatků a algoritmizace rozhodovacího mechanismu jednoduchého expertního systému z hlediska filozofie Petriho sítí.

2. ÚVOD DO EXPERTNÍCH SYSTÉMŮ

Hlavním úkolem této práce je vytvořit základní části expertního systému pomocí Petriho sítí. Jde především o to, jak pomocí Petriho sítě uchovávat znalosti a implementovat řídící (inferenční) mechanismus, které jsou základními stavebními kameny každého expertního systému. Teoretický popis, rozdělení a příklady použití expertních systémů jsem již provedl v Semestrálním projektu 1 „Netradiční typy expertních systémů“. Nyní bych rád zde jen popsal základní charakteristické rysy expertních systémů a potom přejdu k popisu vlastností Petriho sítí, především těch vlastností, které na závěr této práce využiji k popisu báze znalostí expertního sytému.

2.1 CO JSOU TO EXPERTNÍ SYSTÉMY

Expertní systémy patří mezi nejúspěšnější aplikace umělé inteligence. Od svého komerčního uvedení na začátku osmdesátých let prodělaly významný rozvoj a v současnosti jsou používány v mnoha oblastech lidské činnosti, jako např. ve vědě, technice, obchodě atd.

Expertní systém je vlastně počítačový program, simulující rozhodovací činnost experta při řešení složitých úloh a využívající vhodně zakódovaných, explicitně vyjádřených znalostí, převzatých od experta, s cílem dosáhnout ve zvolené problémové oblasti kvality rozhodování na úrovni experta.[2]. Právě problematika vhodného kódování a uchovávání znalostí je jedním z problémů, které zde budou konkrétněji popsány a řešeny.

2.2 CHARAKTERISTICKÉ RYSY EXPERTNÍCH SYSTÉMŮ

Expertní systémy se vyznačují následujícími charakteristickými rysy

- Oddělení znalostí a mechanismu jejich využívání (tento rys je pro expertní systémy asi nejtypičtější a odlišuje se jím od klasických počítačových programů)
- Schopnost rozhodování za neurčitosti

- Schopnost vysvětlování

2.3 STRUKTURA EXPERTNÍHO SYSTÉMU

Expertní systém obsahuje tyto základní složky

- Báze znalostí
- Inferenční mechanismus
- I/O rozhraní
- Vysvětlovací modul
- Modul pro získávání znalostí

Báze znalostí obsahuje znalosti z určité domény a specifické znalosti o řešení problémů v této doméně. Znalosti mohou být nejrůznějšího charakteru od nejobecnějších znalostí až po vysoce specifické. [1]

Inferenční mechanismus obsahuje obecné algoritmy schopné řešit problémy na základě zadaných faktů pomocí manipulace se znalostmi z báze znalostí. Jedná se vlastně o jakýsi řídící mechanismus využívající znalosti z báze znalostí. Striktní oddělení báze znalostí a řídícího mechanismu je typické pro expertní systémy a právě toto oddělení umožňuje aplikaci kvalitního již vytvořeného inferečního mechanismu na jinou bázi znalostí.

2.4 APLIKACE EXPERTNÍHO SYSTÉMU

Aby mělo smysl použít expertní systém pro řešení nějakého problému, musejí být splněny dvě následující podmínky

1. Musí se jednat o problém složitý rozsahem neurčitostí vztahů, pro nějž exaktní metoda řešení buď není k dispozici, nebo není schopna poskytnout řešení v požadované době [2]
2. Efekty plynoucí z použití expertního systému musejí převyšovat vynaložené náklady. To znamená, že by mělo jít o problém s opakovanou potřebou řešení a značnými finančními dopady, pro nějž jsou lidští experti drazí nebo omezeně dostupní.

2.5 ROZDÍL MEZI DIAGNOSTICKÝM A PLÁNOVACÍM EXPERTNÍM SYSTÉMEM

Z hlediska charakteru řešených úloh lze expertní systémy v podstatě rozdělit do tří skupin. Na systémy diagnostické, plánovací a hybridní. Ve 4. kapitole bude popsán základní rozdíl mezi modelem diagnostického a plánovacího expertního systému pomocí Petriho sítě. Zde uvedu pouze obecné rozdíly.

Úloho diagnostického expertního systému je provádět efektivní interpretaci dat s cílem určit, která z hypotéz z předem stanovené konečné množiny hypotéz nejlépe koresponduje s reálnými daty týkající se konkrétního případu.[8] U diagnostických expertních systému tedy řešení probíhá formou postupného ohodnocování a přehodnocování dílčích a cílových hypotéz v rámci pevně daného vnitřního modelu, který je předem navržen expertem.

Plánovacími expertními systémy jsou obvykle řešeny takové úlohy, kdy je znám počáteční stav a cíl řešení a systém má využitím dat o konkrétním případu nalézt (pokud možno optimální) posloupnost kroků (operátorů), kterými lze cíle dosáhnout. Podstatnou částí takovýchto expertních systémů je generátor možných řešení, který automaticky kombinuje posloupnost operátorů. Výsledkem činnosti plánovacího expertního systému je seznam navrhovaných řešení, obvykle ohodnocený jistou měrou optimality.

3. PETRIHO SÍTĚ

3.1 ÚVOD DO PETRIHO SÍTÍ

Petriho sítě (PN) patří k významným nástrojům pro modelování paralelních systémů a systémů s diskrétním časem. Díky své schopnosti snadno a srozumitelně vyjádřit paralelismus, synchronizaci a kauzalitu událostí jsou vhodným modelem systémů, jakými jsou komunikační protokoly, počítačové či databázové systémy apod. [7]. PN slouží jako modelovací prostředek v celé řadě „nekybernetických disciplín“. Např. v chemii, kde je pohybem tokenu modelován pohyb elektronů nebo molekul, v ekonomii jako modelovací nástroj pro výsledky marketingového průzkumu atd… Petriho sítěmi je označována rozsáhlá třída matematických modelů, které umožňují specifickými prostředky popisovat řídící toky a informační závislosti uvnitř modelovaných systémů. Pojem Petriho sítí byl postupně obohacován a zobecňován tak, aby jeho modelovací schopnost vyhověla praktickým potřebám. V 80. letech začal vývoj Petriho sítí vyšší úrovně (high – level Petri nets), odstraňující některé nedostatky. Petriho sítěmi vyšší úrovně rozumíme barevné PN (Colored Petri nets), časované PN (timed Petri nets) nebo synchronizované PN (synchronized PN). Tyto budou podrobněji popsány v dalších částech této kapitoly.

3.2 STRUKTURA PETRIHO SÍTÍ

Základními elementy PN jsou místa a přechody. Místa (places) jsou graficky zobrazována kružnicemi a přechody (transitions) úsečkami nebo obdélníky. Místa a přechody jsou vzájemně propojeny orientovanými hranami (arcs). Jinými slovy je PN bipartitním orientovaným grafem s dvěma typy uzlů: místy a přechody. [7]

Okamžitý stav systému je definován umístěním značek – tokenů v místech, což je v grafu PN vyjádřeno příslušným počtem teček v místech.

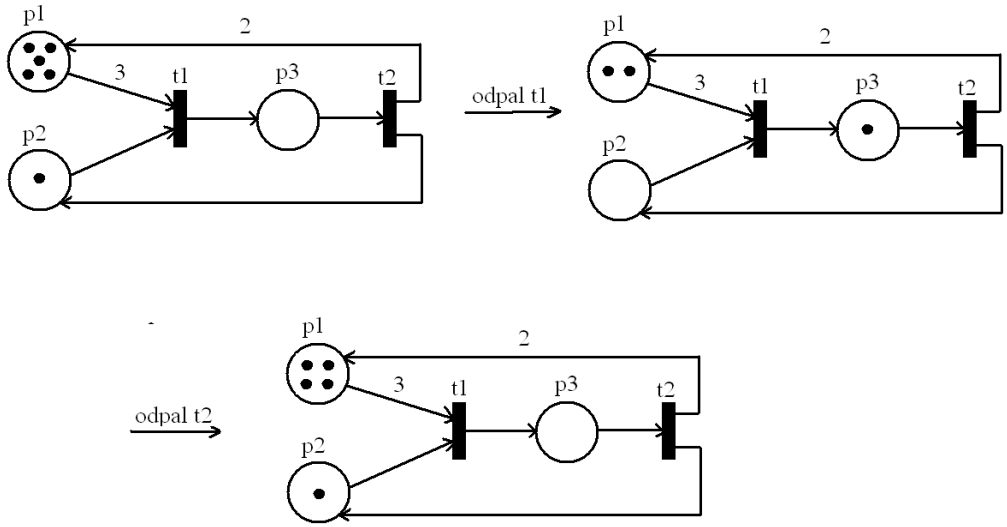
3.2.1 Definice sítě

Trojici $N = (P,T,F)$ nazýváme sítí jestliže

1. P a T jsou disjunktní množiny
2. $F \subseteq (P \times T) \cup (T \times P)$ je binární relace

Množina P se nazývá množinou míst sítě N, množina T množinou přechodů sítě N a relace F tokovou relací sítě N.

Každý přechod má definovánu množinu vstupních a výstupních míst. Vstupní a výstupní podmínky přechodu určují počty odebíraných a umísťovaných tokenů, což je v grafu PN specifikováno ohodnocením orientované hrany. Hrana, která není v grafu explicitně ohodnocena, má implicitní váhu 1. Přechod může být odpálen v případě, že má splněny všechny vstupní podmínky, tzn. všechny jeho vstupní místa obsahují příslušné počty tokenů.



Obr 3.1 Provedení přechodů PN [6]

Na obr. 3.1 je Petriho síť N, pro kterou platí

Trojice $N = (P,T,F)$

$P = \{p_1,p_2,p_3\}$

$T = \{t_1,t_2\}$

$F = \{<p_1,t_1>,<p_2,t_1>,<p_3,t_2>,<t_1,p_3>,<t_2,p_1>,<t_2,p_2>\}$

Ve své práci se budu díle zabývat pouze P/T Petriho sítěmi, které jsou popsány šesticí $N = (P, T, F, W, K, M_0)$, kde

1. trojice (P, T, F) je konečná síť
2. zobrazení $W : F \rightarrow \mathrm{N} \setminus \{0\}$ je ohodnocení hran grafu sítě určující váhu každé hrany
3. zobrazení $K : P \rightarrow \mathrm{N} \cup \{\omega\}$ specifikuje kapacitu (i neomezenou) každého místa
4. zobrazení $M_0 : P \rightarrow \mathrm{N} \cup \{\omega\}$ je počáteční značení míst sítě respektující kapacity míst, tj. $\forall p \in P : M_0(p) \leq K(p)$

Je zcela zřejmé, že K pro síť o p místech bude p – rozměrný vektor, přičemž platí, že kapacita místa P_i je K_i. Značení míst M sítě N bude p – rozměrný vektor, kde na i – tém místě se vyskytuje počet tokenů v místě P_i. Jestliže síť obsahuje f binárních relací neboli orientovaných hran pak W je f – rozměrný vektor, kde w_i určuje váhu i – té relace mezi místem a přechodem.

Jak je vidět na obrázku 3.1 ve výchozí pozici obsahuje místo p_1 5 tokenů, p_2 jeden token a p_3 je prázdné. Výchozí značení sítě je tedy $M_0 = (5,1,0)$. Podmínkou odpalitelnosti přechodu t_1 je přítomnost alespoň 3 tokenů v p_1 (orientovaná hrana p_1-t_1 má váhu 3) a minimálně 1 token v p_2. Přechod t_2 může být odpálen pouze za přítomnosti nejméně 1 tokenu v p_3. Je tedy zcela zřejmé, že jediný přechod, který může být odpálen je t_1. Po jeho provedení zůstanou v p_1 2 tokeny (z 5 jsou 3 odebrány), p_2 bude prázdné a do p_3 bude umístěn 1 token (hrana $t_1 – p_3$ má váhu 1). V tomto stavu není splněna podmínka odpalitelnosti t_1, ale naopak odpalitelným se stal přechod t_2. Odpálíme – li přechod t_2 bude z místa p_3 odebrán 1 token, do p_2 bude 1 token přidán a do p_1 budou přidány 2 tokeny. Je tedy patrno, že během odpalování přechodů v PN se aktuální počet tokenů mění. Např. na výše popsaném obrázku se před odpalem t_2 vyskytují pouze 2 tokeny a po jeho provedení jich je 5. Vliv na počet

tokenů jednotlivých přechodů nejlépe popisuje incidenční matice, která bude popsána v následující podkapitole.

3.3 INCIDENČNÍ MATICE

Incidenční matice v udává vztah mezi počátečním (M_0) a koncovým stavem (M_k) PN. Vektor $M_k = (p_1, p_2, p_3,\ldots)$ je řada čísel udávající počet tokenů v příslušném místě v k-tém stavu. Jestliže podle definice Petriho sítě N je množina míst $P = \{p_1, p_2,\ldots, p_n\}$ a množina přechodů $T = \{t_1, t_2, \ldots, t_m\}$ pak incidenční matice A je matice $m \times n$,

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ \vdots \\ p_n \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & \ldots & \ldots & t_m \end{array} \\ \begin{bmatrix} a_{11} & a_{12} & \ldots & \ldots & a_{1m} \\ a_{21} & & & & \vdots \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ a_{n1} & & \ldots & \ldots & a_{nm} \end{bmatrix} \end{array}$$

Incidenční matice je $m \times n$ celých čísel, kde prvek a_{xy} označuje vliv přechodu t_y na počet tokenů v místě p_x. Této vlastnosti bude využito v kapitole 5 při popisu báze znalostí pro plánovací expertní systém.

Na obr. 3.1 je zobrazena jednoduchá PN jejíž výchozí stav lze zapsat jako $M_0 = (5, 1, 0)$, což značí 5 tokenů v p_1, 1 v p_2, atd. Postupné odpalování přechodů zapisujeme ve tvaru $M_0\ [t_1> M_1\ [t_2> M_2\ [t_1> M_3 \ldots$ Pro příklad uvedený na obr 2.1 je to

$(5, 1, 0)\ [t_1>\ (2, 0, 1)\ [t_2>\ (4, 1, 0)\ [t_1>\ (1, 0, 1)$

Incidenční matice by potom měla tento tvar

$$A = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cc} t_1 & t_2 \end{array} \\ \begin{bmatrix} -3 & 2 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \end{array}$$

Incidenční matice popisuje vliv jednotlivých přechodů na jednotlivá místa. Zde můžeme pozorovat, že odpálení přechodu t_1 způsobí odebrání 3 tokenů z místa p_1 (hodnota -3). Naopak t_1 přidává 1 token do místa p_3 (hodnota 1).

3.4 ŽIVÁ, OMEZENÁ A BEZPEČNÁ PETRIHO SÍŤ

V této kapitole bude vysvětleno co znamená když je PN živá, omezená, nebo bezpečná.

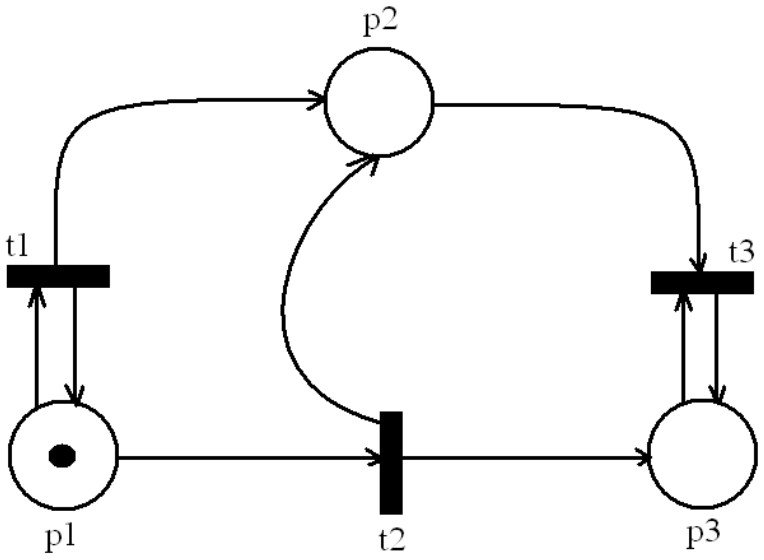
Živost sítě nám popisuje odpalitelnost jednotlivých přechodů. PN je živá, pokud lze odpálit všechny její přechody, ať už přímo nebo přes nějaký další její přechod.

O PN můžeme prohlásit, že je bezpečná, jestliže se v jejích místech p_1 ... p_n vyskytuje maximálně 1 token. PN uvedená na obrázku 3.1 očividně bezpečná není. V praxi se vyskytuje velké množství sítí, které nejsou bezpečné, protože sítě, které mohou v určitém místě mít nanejvýš 1 token jsou velice omezené a pro modelování složitějších systémů prakticky nepoužitelné.

Pro vysvětlení omezenosti sítě bude třeba nejdříve vysvětlit pojem strom pokrytelnosti.

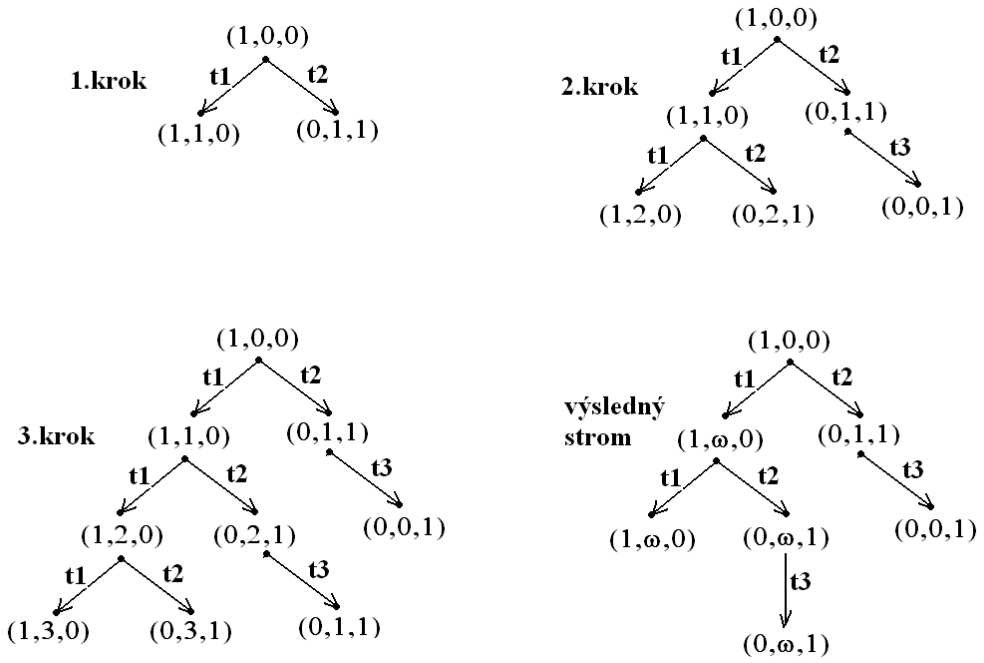
3.4.1 Strom pokrytelnosti

Strom pokrytelnosti nebo také strom dosažitelných řešení nám ilustruje množinu stavů M, kterých může daná PN dosáhnout. Konstrukci stromu pokrytelnosti popíši pro PN na obr. 3.2



Obr. 3.2 Petriho síť pro popis stromu pokrytelnosti

Pro tuto PN by strom pokrytelnosti vypadal následovně.



Obr. 3.3 Konstrukce stromu pokrytelnosti

Obr. 3.3 demonstruje konstrukci stromu pokrytelnosti ve 4. krocích. V kroku 1 zapíšeme výchozí stav (1,0,0), ze kterého se můžeme pomocí odpalu t_1 dostat do stavu (1,1,0) a pomocí odpalu t_2 do (0,1,1). Odpal přechodu t_3 není v této fázi možný. Kroky 2 a 3 popisují další odpaly jednotlivých přechodů. Jak je vidět, odpal t_1 způsobí že v místě p_2 vzroste počet tokenů o 1 a v ostatních zůstane nezměněn. V takovém případě, říkáme, že stav (1, 1, 0) dominuje stavu (1, 0 ,0) a dominující stav zapíšeme jako (1, ω, 0), kde symbol ω značí blíže neurčený počet tokenů. Strom pokrytelnosti má velkou vypovídací hodnotu o celkovém charakteru Petriho sítě, proto bude s úspěchem použit při popisu expertního systému *plánovacího* typu.

3.4.2 Omezená Petriho síť

V předcházející podkapitole jsem vytvořil strom pokrytelnosti pro jednoduchou PN. V tomto stromu se vyskytuje symbol ω, značící blíže neurčený počet tokenů. Pokud se tento objeví, můžeme říci, že příslušná PN není omezená. PN na obr. 3.2 tedy není omezená.

Obecně lze říci, že PN je není omezená pokud počet tokenů v kterémkoliv místě PN nepřekročí předem definovanou hodnotu k.

3.5 JAZYKY PETRIHO SÍTÍ

PN vznikly zobecněním konečných automatů. Tento fakt se projevuje i v definici jazyků PN. V analogii s jazyky popisujícími konečné automaty je vhodné zavést i pro PN pojem vstupní abeceda, počáteční stav a množina koncových stavů. Výpočetní posloupnost reprezentovaná množinou přechodů T je nahrazena abecedou Petriho sítě Σ a zobrazení $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$, které každému přechodu sítě přiřadí symbol Σ nebo prázdný symbol ε. Zobrazení λ budeme nazývat ohodnocením přechodů a příslušnou Petriho síť jako ohodnocenou.

Jak bylo uvedeno již v kapitole 3.1, stav Petriho sítě je popsán množinou M. Počáteční stav odpovídá počátečnímu značení M_0. Koncový stav M_k je naopak vektor popisující počet tokenů v jednotlivých místech v konečném cílovém stavu. Série přechodů, které je potřeba odpálit se pak nazývá jazykem PN. Přičemž se nemusí vždy jednat o nejkratší možnou cestu – tedy o nejmenší možný počet odpálených přechodů od stavu M_0 do stavu M_k.

Existují čtyři různé definice, které vymezují třídy jazyků označované jako jazyky typu L, G, T a P. O jazycích typu G, T a P je možno získat více informací v [5] a [6]. Ve své práci se zaměřím na jazyky typu L. Tato třída jazyků je dostatečně obecná, abych pomocí ní mohl popsat vazby v expertním systému a zároveň byla z hlediska svých vlastností nejvíce studována a analyzována.

3.5.1 Jazyk typu L

Nechť N je Petriho síť s počátečním značením M_0, s ohodnocením přechodů $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$, se (zobecněnou) přechodovou funkcí δ

$$\delta : [M_0 > \times T^* \rightarrow [M_0 > \qquad (1)$$

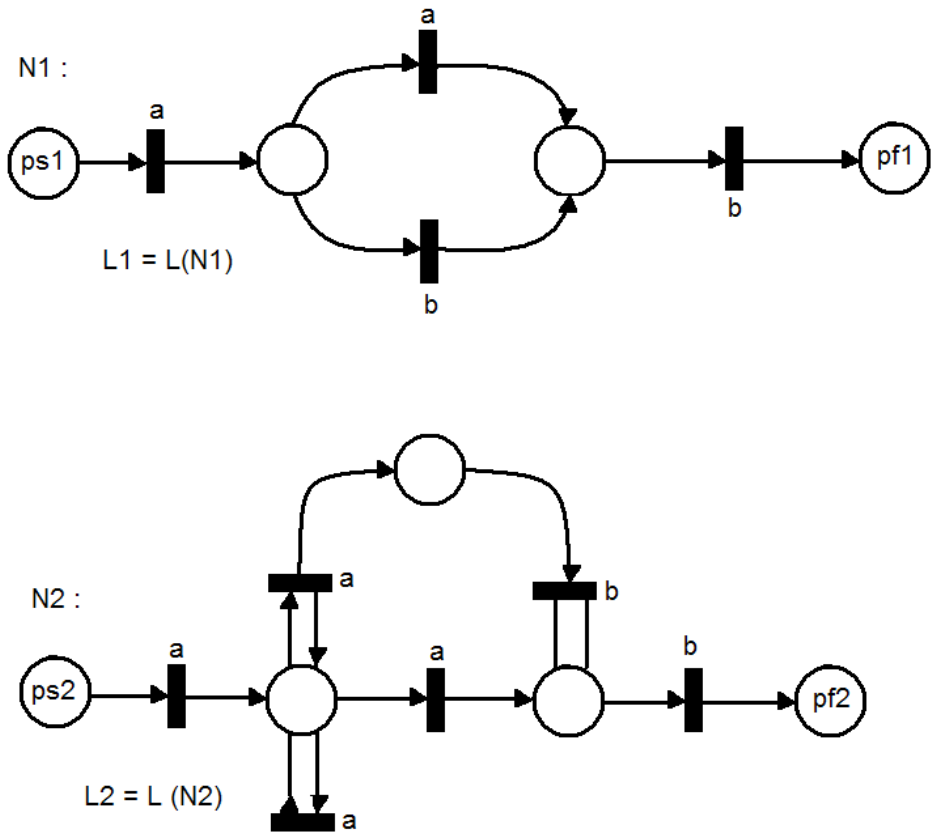
a s množinou koncových stavů $Q_F \in M_0$. Jazyk L(N) Petriho sítě definovaný jako

$$L(N) = \{\lambda(\alpha) \mid \alpha \in T^* \wedge \delta((M_0), \alpha) \in Q_F\} \qquad (2)$$

se nazývá jazykem typu L.

3.5.2 Sjednocení Petriho sítí

Jestliže Petriho síť N_1 generuje jazyk L_1 a Petriho síť N_2 jazyk L_2, pak sloučením sítí N_1 a N_2 (sloučením koncového místa N_1 a počátečního N_2 do jednoho místa) získáme síť N generující jazyk $L = L_1 \cup L_2$.



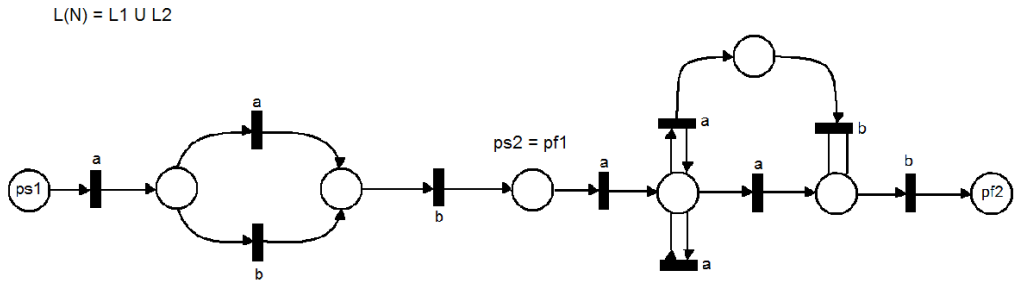
Obr 3.4 Petriho sítě generující jazyky typu L

Petriho síť N_1 generuje jazyk $L_1 = a \cdot (a + b) \cdot b$

N_2 generuje jazyk $L_2 = \{a^m b^n \mid m > n > 1\}$

Jak již bylo řečeno výše, jazyk typu L znázorňuje posloupnost přechodů od počátečního ke koncovému místu. V tomto případě jsou označení přechodů nahrazeny písmeny abecedy, aby bylo možno více přechodům přiřadit stejný symbol. Jedná se pouze o formalní zápis, z hlediska funkčnosti celé sítě se nic nemění. V případě sítě N_1 je generovaný jazyk typu L $-L_1 = a \cdot (a + b) \cdot b$, kde stejně jako v logických funkcích součin znázorňuje seriové spojení – za sebou (přechodu a , a + b, b) zatímco znaménko + označuje paralelismus (v logické sféře "OR"). Do cílového stavu se dostanu buď přes a nebo b. Naproti tomu a^m – napovídá, že přechod označený a – může být odpálen až m – krát.

Jestliže spojím dvě Petriho sítě generující jazyk L_1 popř. L_2, výsledná síť bude rovněž generovat jazyk typu L.



Obr. 3.5 Spojení 2 Petriho sítí generující jazyk L

Výsledná Petriho síť $L = L_1 \cup L_2$ tedy generuje jazyk

$$L = \{a \cdot (a+b) \cdot b \cdot a^m \cdot b^n \mid m > n > 1\}$$

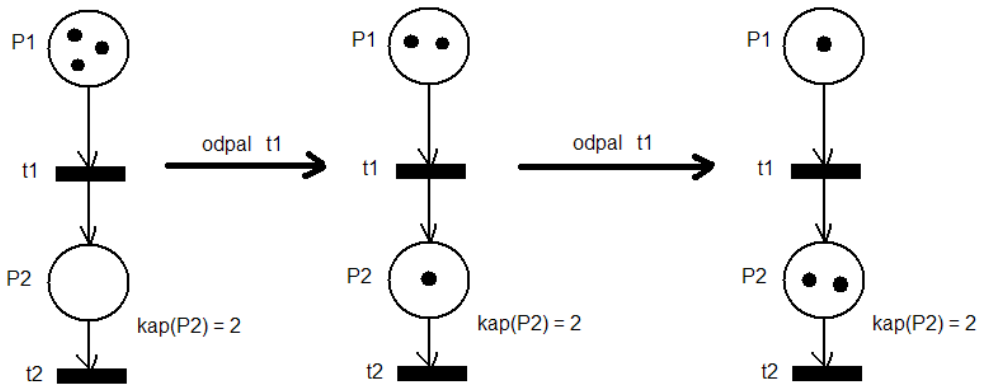
Zachování vlastností (včetně syntaxe daného jazyka) po spojení 2 nebo více PN bude výhodné zejména při popisu spojení báze znalostí a inferenčního mechanismu expertního systému, což bude popsáno v kapitole 5.

4. PETRIHO SÍTĚ VYŠŠÍ ÚROVNĚ A MOŽNOSTI JEJICH VYUŽITÍ V EXPERTNÍCH SYSTÉMECH

Jak jsem již bylo napsáno v úvodu, PN jsou vhodným modelovacím nástrojem pro zobrazení systémů a procesů v mnoha oblastech. V této kapitole se zaměřím na možnosti uplatnění některých vlastností PN u expertních systémů. Základní vlastnosti PN jsou podle mě nedostačující pro zachycení rozhodovacího mechanismu i poměrně jednoduchých expertních systémů. Jedná se především o nejednoznačnost, který přechod mám v dané chvíli odpálit, jestliže jsou splněny vstupní podmínky odpalitelnosti pro více přechodů. Taková síť se poté nazývá Petriho síť s volným výběrem. Tento problém řeší především časované PN (timed Petri nets), prioritní PN (Priority Petri nets), Synchronizované PN (sychronized Petri nets). Přesto zde budou popsány i další typy Petriho sítí „vyšší úrovně“ jako např barvené.

4.1 PETRIHO SÍTĚ S OMEZENOU KAPACITOU (FINITE CAPACITY PETRI NETS)

PN s omezenou kapacitou (angl. Finite capacity PN) se vyznačují tím, že obsahují alespoň jedno místo ve své struktuře, které má omezenou kapacitu tokenů, které může příjmout. Podstatný rozdíl oproti jednoduchým PN (Simple PN) je v tom, že zde pro odpalitelnost přechodu nejsou testovány jen vstupní podmínky, ale také podmínky výstupní. Přechod nemůže být odpálen, jestliže všechny jeho vstupní místa obsahují dostatečný počet tokenů, ale jeho odpalem by byla překročena kapacita výstupního místa.



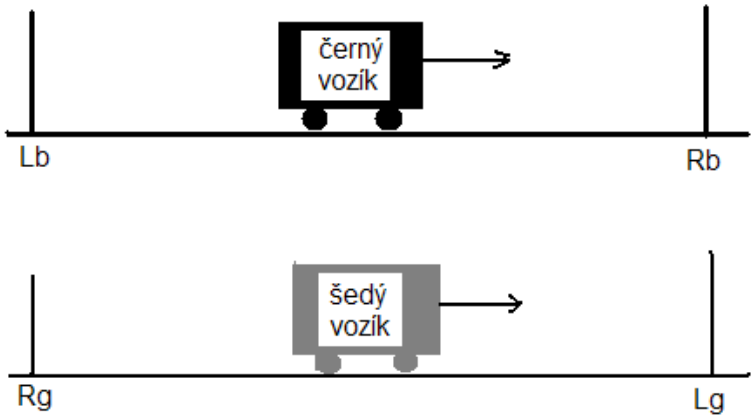
Obr 4.1 PN s omezenou kapacitou místa

Na obrázku 4.1 je velmi jednoduchá PN, jejíž místo p_2 má kapacitu omezenou na 2 tokeny, což způsobuje, že přechod t_2 je odpalitelný pouze dvakrát, přesto, že v místě p_1 je stále potřebný počet tokenů, odpal přechodu t_2 by způsobil překročení kapacity v místě p_2. Jestliže je ve vstupních místech dostatečný počet tokenů pro odpal přechodů a zároveň ve výstupních místech dostatečný prostor, aby nebyla překročena předem definovaná konečná kapacita místa, můžeme danou síť prohlásit za bezkontaktní.

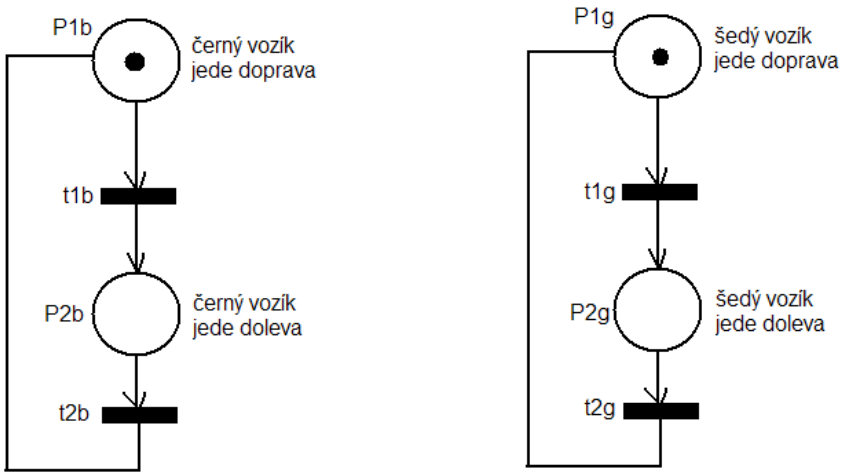
4.2 BARVENÉ PETRIHO SÍTĚ (COLORED PETRI NETS)

Až doposud jsem se zabýval Petriho sítěmi jejíž tokeny a přechody měli jednu blíže nedefinovanou barvu. V schématech byly samozřejmě tyto PN kresleny černou barvou pro nejlepší viditelnost. Tokeny a přechody mohou být tzv. barvené. Což znamená, že se v určitém místě mohou objevovat třeba žluté nebo červené tokeny a stejně tak mějme například přechod, který může odpalovat pouze žluté nebo červené tokeny. Další podstatnou výhodou je, že pokud přechod může odpalovat více barev, nemusí to být v jeden okamžik. Mám- li přechod, který přemisťuje červené a žluté tokeny mohu v jeden okamžik pomocí jeho odpalu přemístit žlutý token z místa

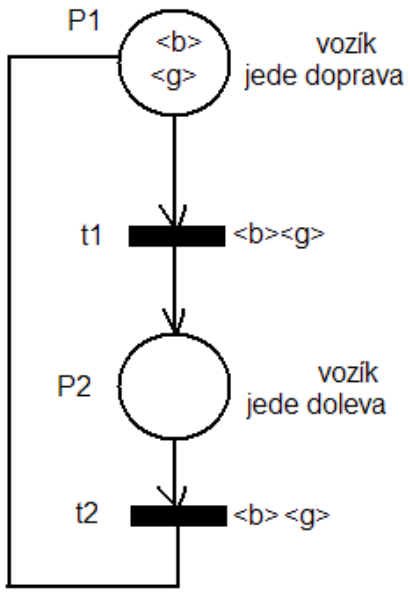
p_1 do místa p_2 a v další okamžik přemístit červený token z místa p_1 do místa p_2, samozřejmě za předpokladu splnění všech vstupních/výstupních podmínek nutných k provedení odpalu. Tato vlastnost je využitelná například chceme-li modelovat více identických vzájemně se neovlivňujících systémů, které se mohou nacházet ve stejných stavech a mají stejné podmínky přechodu mezi těmito stavy.



Obr 4.2 a systém 2 vozíků [4]



Obr 4.2 b modelování pomocí 2 jednoduchých PN

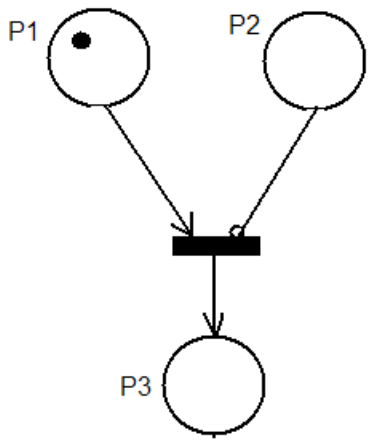


Obr 4.2 c model pomocí barvené PN

Na obrázku 4.2 a jsou zobrazeny 2 vozíky (šedý a černý) pohybující se zleva doprava a zpět. Podmínkou změny směru je dosažení okrajové hranice. Na obr. 4.2b je tento proces modelován pomocí dvou jednoduchých PN kde místo p_{1b} (p_{1g}) symbolizuje pohyb černého (šedého) vozíku doprava, odpal přechodu t_{1b} (t_{1g}) značí dosažení pravého okraje R_b popř. R_g a místo p_{2b} popř. p_{2g} značí pohyb doleva. Jak je vidět obě PN jsou naprosto identické, co se týče míst a podmínek odpalitelnosti přechodů. Proto je vhodné celý model zjednodušit na jednu barvenou PN kde černý token symbolizuje stav černého vozíku a šedý token stav šedého vozíku. Je tedy možné přechodem t_1 odpálit černý token do místa p_2 a šedý ponechat v místě p_1, což symbolizuje, že černý vozík narazil na pravý okraj a jede směrem doleva, zatímco šedý stále pokračuje v pohybu doprava. Celá situace je zobrazena na obr. 4.2c

4.3 ROZŠÍŘENÉ PETRIHO SÍTĚ (EXTENDED PETRI NETS)

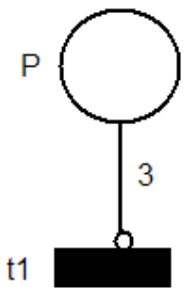
Místa a přechody jsou spojeny orientovanými hranami, které jsou graficky znázorněny jako úsečky, které mají na svém konci šipku směřující do místa, kam směřuje orientovaná hrana. Rozšířené PN obsahují speciální orientované hrany označené jako inhibitory. Graficky jsou tyto hrany zobrazeny tak, že místo šipky mají na svém konci kolečko. V logických obvodech se tento symbol používá pro označení negace (invertor). Jeho význam v PN je analogický. Pokud máme spojen přechod s určitým místem pomocí hrany s inhibitorem, pak tento přechod může být odpálen pouze neobsahuje – li příslušné místo žádný token.



Obr. 4.3 PN s inhibitorem [4]

Přidání inhibitoru na konec orientované hrany způsobí, že přechod t_1 je odpalitelný pouze v případě, že místo p_1 bude obsahovat nejméně 1 token a místo p_2 žádný token. Po jeho provedení se do místa *p3* dostane 1 token. PN s inhibitorem slouží k provádění tzv. zero testů nebo též threshold testů, což znamená testování, zda je dané místo prázdné. Je – li orientovaná hrana s inhibitorem váhována váhou *w*, v případě, že by inhibitor neobsahovala, po odpalu příslušného přechodu by z místa bylo odebráno *w* tokenů, tak *w* váhovaná hrana s inhibitorem, umožňuje odpálit

přechod, se kterým je spojena pouze za podmínek, že místo na jejím vstupu obsahuje nanejvýš $w - 1$ tokenů.



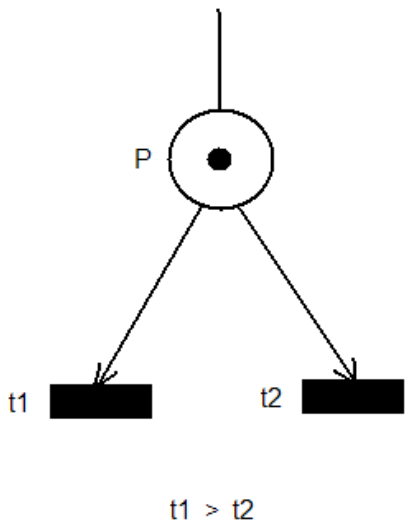
Obr. 4.4 Orientovaná hrana s váhou 3

Přechod t_1 na obr. 4.4 je odpalitelný pouze v případě, že se v místě p_1 vyskytují nanejvýš 2 tokeny.

4.4 PRIORITNÍ PETRIHO SÍTĚ (PRIORITY PETRI NETS)

Prioritní PN jsou s úspěchem využívány tam, kde potřebujeme jasně určit, který přechod má být odpálen. Značná část PN v určitých momentech obsahuje více než jeden odpalitelný přechod. Tato situace nastává hlavně tehdy, když z jednoho místa vede více orientovaných hran k více přechodům. Zde nastává tzv. konflikt odpalitelnosti. Petriho sítě u kterých nastává konflikt odpalitelnosti se nazývají Petriho sítě s volným výběrem. Pokud využíváme PN pro popis chování nějakého sytému, tak fakt, který přechod máme zrovna odpálit, je dán vnějšími podmínkami. Na příkladě s posunem barevných vozíků bylo dáno, který přechod bude odpálen dosažením příslušného okraje. Tehdy se samozřejmě nejednalo o konflikt odpalitelnosti. U expertních systémů je zvláště nežádoucí jakákoliv neurčitost v kterékoliv části rozhodovacího (inferenčního) procesu. Proto je velice výhodná vlastnost prioritních PN, které poskytují možnost různým přechodům nastavit různé

priority. Pokud jsou tedy splněny vstupní podmínky odpalitelnosti u více přechodů a odpálení jednoho znemožní odpal dalších, pak je odpálen přechod s nejvyšší prioritou.



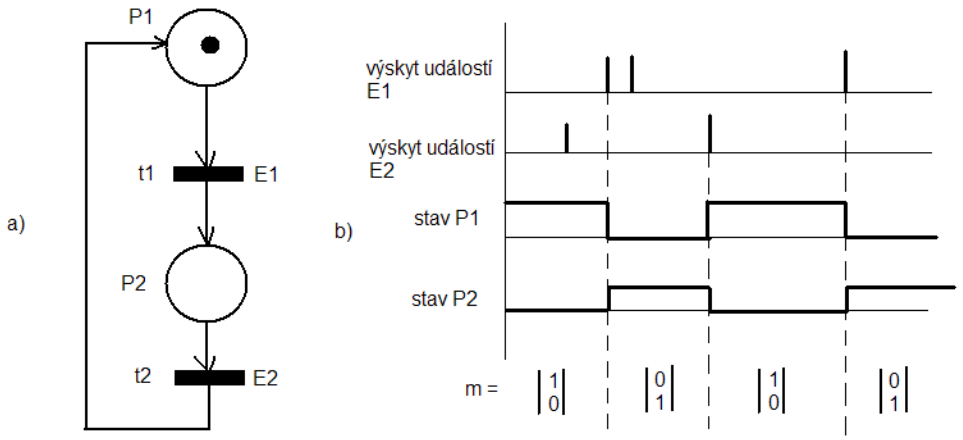
Obr 4.5 PN s nastavenými prioritami odpalu jednotlivých hran.

Místo p_1 obsahuje token, tím jsou splněny vstupní podmínky odpalitelnosti, jak pro přechod t_1 tak pro t_2 a za normálního stavu by nastal typický konflikt odpalitelnosti. Nastavíme – li přechodu t_1 vyšší prioritu než t_2, pak bude t_1 odpálen. Přechod t_2 už potom odpálen nebude, protože odebráním tokenu z místa p_1 dojde k nesplnění vstupních podmínek odpalitelnosti.

4.5 SYNCHRONIZOVANÉ PETRIHO SÍTĚ (SYNCHRONIZED PETRI NETS)

V předcházející podkapitole jsem řešil konflikt odpalitelnosti nastavením různých priorit přechodů u PN. Budu – li v dalších kapitolách demonstrovat popis

expertního systému pomocí PN, můžu dojít do fáze, kdy budu mít u více přechodů splněny podmínky odpalitelnosti a např. u přechodů t_1, t_2, t_3 a v jeden okamžik budu potřebovat odpálit přechod t_2 a jakmile se znovu splní podmínky odpalitelnosti pro všechny přechody, budu chtít odpálit např. t_1. Konstantním nastavením priorit přechodů by při splnění stejných podmínek byl odpálen vždy jeden a tentýž přechod a ostatní by se stali mrtvými (tzn. nikdy neodpalitelnými.). Synchronizované PN tento konflikt řeší zavedením tzv. vnějších událostí (external events). Taková vnější událost je vlastně signál závislý na čase, který má hodnotu 0. Ovšem obsahuje krátké časové jednotkové pulsy. Jakmile se takový puls dostane na vstup konkrétního přechodu, je testováno, zda je možno tento přechod odpálit, pokud ano je odpálen, pokud ne - odpálen není. To znamená, že konkrétní přechod může být odpálen, pouze v případě, že jsou splněny podmínky odpalitelnosti a že se na přechodu objeví jednotkový puls. V případě konfliktu odpalitelnosti, kdy má každý přechod svoji sadu vnějších událostí (set of external event) bude odpálen ten přechod, u kterého se objeví jednotkový puls jako první.



Obr 4.6 ukázka chování synchronizované PN

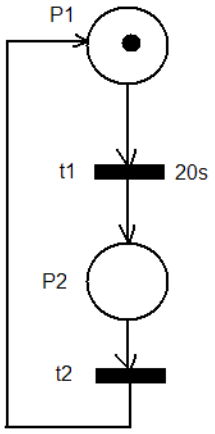
Na obrázku 4.6 a je synchronizovaná PN. Přechod t_1 je synchronizován signálem E_1 a přechod t_2 signálem E_2. Jak je vidět z grafu *4.6 b*, objeví – li se na přechodu t_2 „spouštěcí“ puls, nic se nestane protože přechod t_2 je v tuto chvíli

neodpalitelný. Jakmile je však tento puls zaznamenán na t_1, tak při splněných podmínkách odplaitelnosti dojde k odpalu t_1 a token je předán do t_2, kde se opět čeká na příchod pulsu v signálu E_2. Je logické, že puls signálu E_1 nebude mít v tomto stavu systému na nic vliv.

4.6 ČASOVANÉ PETRIHO SÍTĚ (TIMED PETRI NETS)

V mnoha systémech je důležitá funkce času. Pomocí PN lze modelovat např. systém řízení křižovatky, ovládání semaforů, světelných návěstidel apod. Tato problematika je mimojité podrobněji popsána v [10]. Pro tyto systémy je charakteristické, že se jejich stav mění podle času (např. červená svítí 20 s pak se rozsvítí oranžová apod.) Řídící systémy realizující tyto funkce obsahují časovače užívané k odměření předem definovaných časových úseků.

Časované PN se vyznačují tím, že daný přechod je odpálen až po určitém definovaném časovém úseku. Dalo by se říci, že se jedná o odpaly se zpožděním. V systému modelující řízení křižovatky je žádoucí, pokud chceme co nejvěrohodněji modelovat reálné chování systému, musí být dodrženy časové úseky, kdy systém zůstává v konkrétních stavech (u PN to odpovídá místům ve kterých je token).



Obr 4.7 Příklad časované PN

Je – li token v místě p_1 (což by v ekvivalenci s křižovatkou odpovídalo, že na semaforu svítí červená) je přechod odpálen až po 20s a poté je token přemístěn do stavu p_2. Analogicky to může symbolizovat např. 20 s svítí červená (p_1) pak se rozsvítí oranžová *(p_2)*.

Pomocí časovaných PN je možné také modelovat např. chování operačního systému (stav WAIT – čekající, po určitém časovém úseku READY atd…) apod. Chování různých dějů závislých na čase můžeme popsat buď deterministicky nebo stochasticky. Deterministicky znamená konstantní doba trvání a stochasticky náhodná doba trvání. Stochastické doby trvání přechodů lze využít již v několikrát zmíněných konfliktech odpalitelnosti, kdy potřebuji rozhodnout, který přechod má být odpálen. Tím, že dobu trvání nastavím u těchto přechodů stochasticky, tak se náhodně mění a tudíž je zřejmé, že odpálen bude ten přechod, u kterého je náhodný čas odměřen jako nejkratší.

5. PETRIHO SÍTĚ JAKO NÁSTROJ PRO TVORBU BÁZE ZNALOSTÍ

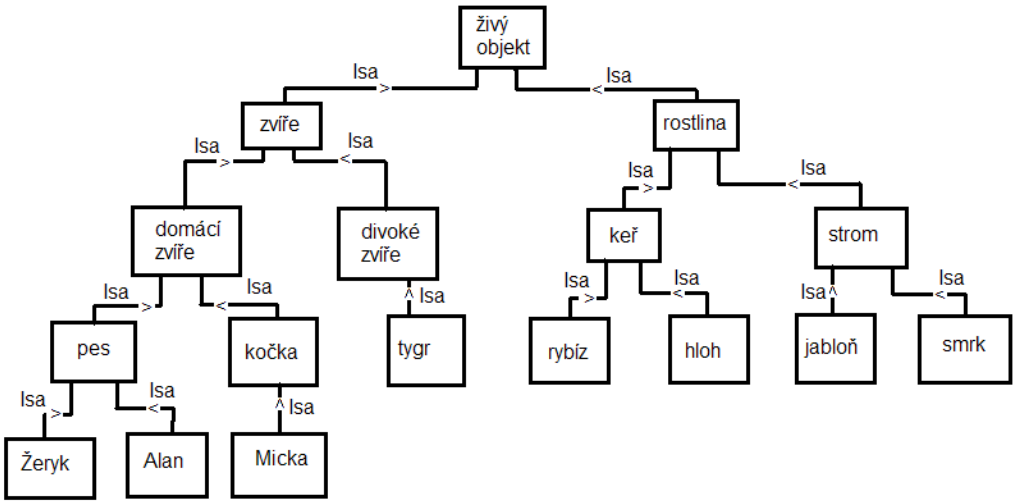
V této kapitole bude popsán proces tvorby expertního systému pomocí PN. Jak jsem již zmínil v úvodní kapitole, věnované teoretickému popisu ES a jejich charakteristických vlastností, mezi nejzákladnější části ES patří báze znalostí, báze faktů a inferenční (rozhodovací) mechanismus. Zaměřím se tedy nejdříve na reprezentaci znalostní báze expertního systému, poté demonstruji inferenční mechanismus. Rovněž předvedu rozdíl mezi bází znalostí diagnostického a plánovacího expertního systému z hlediska přístupu inferečního mechanismu. Faktický rozdíl mezi oběma typy systémů byl popsán v kapitole 2. Ačkoliv existují i další typy expertních systémů, než jen diagnostické a plánovací, ve své práci jsem se zaměřil pouze na tyto 2 typy, protože jsou mezi expertními systémy nejčastěji využívané. Diagnostické expertní systémy zaujímají většinu použitých expertních systémů. Plánovací systémy nacházejí rovněž své uplatnění, proto se na ně zaměřím ve třetí části této kapitoly. Tak jak se u obou typů liší způsob vyhodnocování znalostí ze znalostní báze, bude patrný i rozdíl ve struktuře, popisující příslušné inferenční mechanismy.

5.1 TVORBA BÁZE ZNALOSTÍ EXPERTNÍCH SYSTÉMŮ POMOCÍ PN

Problém reprezentace znalostí je klíčovým problémem umělé inteligence, protože znalosti jsou základem každého inteligentního systému. Tedy i expertních systémů. Existují samozřejmě i prázdné expertní systémy, tedy systémy bez znalostní báze, avšak jejich použití je značně omezující a proto zde budou popsány systémy s bází znalostí. Do problematiky návrhu báze znalostí patří i výběr relevantních znalostí a jejich transformace do vhodných datových struktur a uložení těchto struktur do počítače. Zřejmě platí, že čím specifičtější bude struktura pro daný problém, tím specifičtější a výkonnější může být příslušný inferenční mechanismus

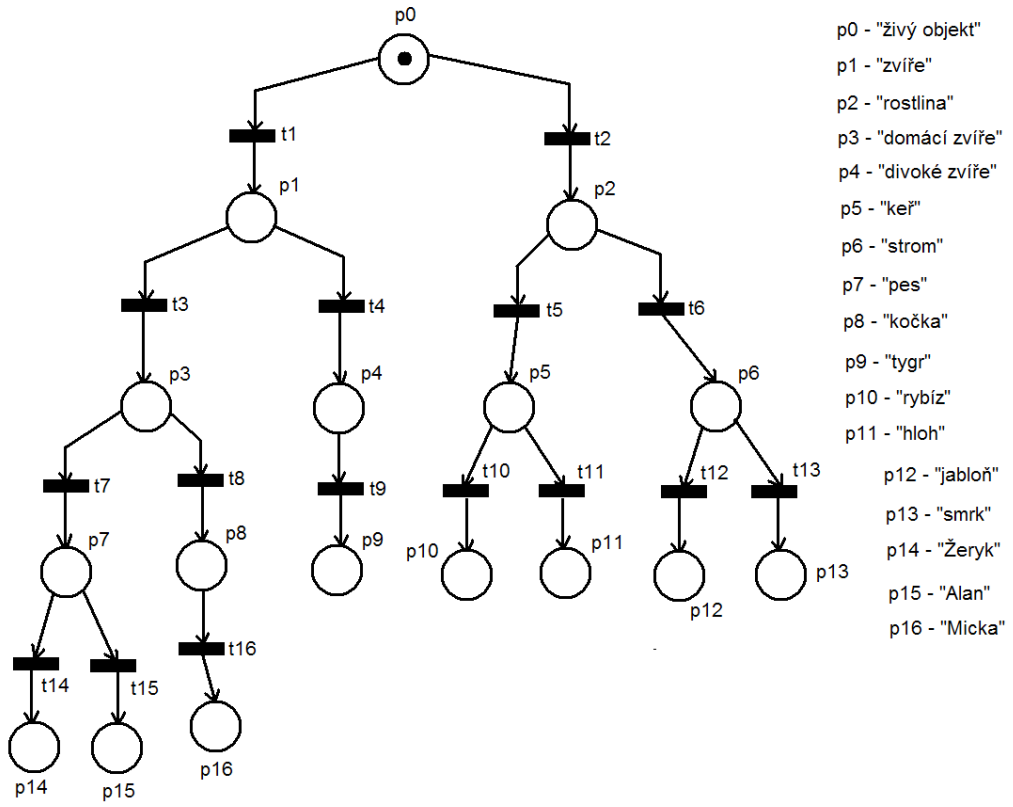
[13]. Struktura báze znalostí vytvořená pomocí Petriho sítí je hlavní náplní této kapitoly.

Mezi nejrozšířenější struktury pro reprezentaci znalostí patří sémantické sítě. Ačkoli sémantické sítě nejsou tématem této práce, bude zde popsán rozdíl v popisu znalostí pomocí sémantických a Petriho sítí. Východiskem pro sémantické sítě byly tzv. ISA (*is a*…= je) a ISPART (*is a part of*…. = je částí). Příklad jednoduché sémantické sítě můžeme vidět na obr. 4.1



Obr. 5.1 Sémantická síť ISA [13]

Na obr. 5.1 se sémantická síť popisující bázi znalostí nějakého pravděpodobně diagnostického expertního systému, který má k dispozici nějaký živý objekt (výchozí bod) a postupně (pomocí inferenčního mechanismu) specifikuje zda se jedná o zvíře či rostlinu a informace o daném objektu dále specifikuje. Na následujícím obrázku 5.2 bude prezentována tatáž báze znalostí s tím rozdílem, že tentokrát budou data zaznamenána ve struktuře Petriho sítě. Výhodou popisu znalostí pomocí Petriho sítě je možnost aplikace různých operací (řízení, prohledávání, porovnávání atd.), strukturovanost, zpřístupnitelnost (přístupné jednoduchým způsobem), modulárnost (možnost přidat i odebrat znalosti), přirozenost a srozumitelnost.



Obr. 5.2 Petriho síť popisující bázi znalostí

Přítomnost tokenu v místě p_0, které je (podle seznamu na pravé straně obrázku) interpretován jako “živý objekt”, značí, že zkoumaný objekt je označen jako “živý objekt”. Odpal přechodu t_1 odpovídá faktu, že daný živý objekt byl diagnostikován jako zvíře. Tato informace je poté v síti uložena přítomností tokenu v místě p_1 (zvíře).

Jak již bylo řečeno v kapitole 3 v Úvodu do Petriho sítí, stav Petriho sítě popisuje množina M, jejíž prvky jsou počty tokenů na jednotlivých místech. V tomto případě má Petriho síť popisující vektor obecně $M = (p_0, p_1, \ldots, p_{16})$. Pro počáteční stav sítě M_0 platí $M_0 = (1, 0, 0, 0, \ldots 0)$. Vektor jehož první prvek je 1 a ostatní 0 v této bázi znalostí označuje fakt, že se jedná o živý objekt.

Představme si situaci, kdy bude například daný živý objekt diagnostikován jako hloh. Nejdříve je potřeba určit, že se jedná o rostlinu (odpal *t2*), rostlina je poté specifikována jako keř (odpal t_5) a keř je určen jako hloh (odpal t_{11}). Je tedy vidět, že určení, že se jedná o hloh je symbolizováno řadou odpalů t_2-t_5-t_{11}. Samotná informace, že daný objekt je hloh je uložena ve vektoru Petriho sítě

M_3 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0).

Trojka proto, že se jedná o stav po 3 odpalech. V úvodu této kapitoly bylo řečeno, že jedním ze zásadních problémů při návrhu a realizaci báze znalostí je problém reprezentace dat a forma jejích uložení v počítači. Pomocí Petriho sítě lze tedy uložit nejen konkrétní informaci (že se jedná např. o hloh) ale rovněž cestu dosažení tohoto faktu formou posloupnosti odpálených přechodů (např. t_2, t_5, t_{11}) Posloupnost odpálených přechodů odpovídá jazyku Petriho sítě typu L (jak bylo popsáno v kapitole 2 – Jazyky Petriho sítí). Jazyk typu L, který je generován výše uvedenou sítí je ve tvaru

$$L = t_1 \cdot \left[t_3 \cdot \left(t_7 \cdot \left(t_{14} + t_{15}\right) + t_8 \cdot t_{16}\right) + t_{14} \cdot t_9\right] + t_2 \cdot \left[t_5 + \left(t_{10} + t_{11}\right) + t_6 \cdot \left(t_{12} + t_{13}\right)\right]$$

Součet v tomto případě vyznačuje 1 z n možností. Konkrétně tedy 1 ze 2. Prakticky to odpovídá faktu, že např. tentýž živý objekt nemůže být diagnostikován jako zvíře a rostlina současně.

Je důležité chápat rozdíl mezi bází faktů a bází znalostí. Báze faktů je u této sítě vyjádřena počátečním značením M_0. Tedy jedná se o živý objekt. Báze znalostí je reprezentována vektory Petriho sítě P, T, F.

$P = \{p_0, p_1, \ldots, p_{16}\}$ $T = \{t_1, t_2, \ldots, t_{16}\}$

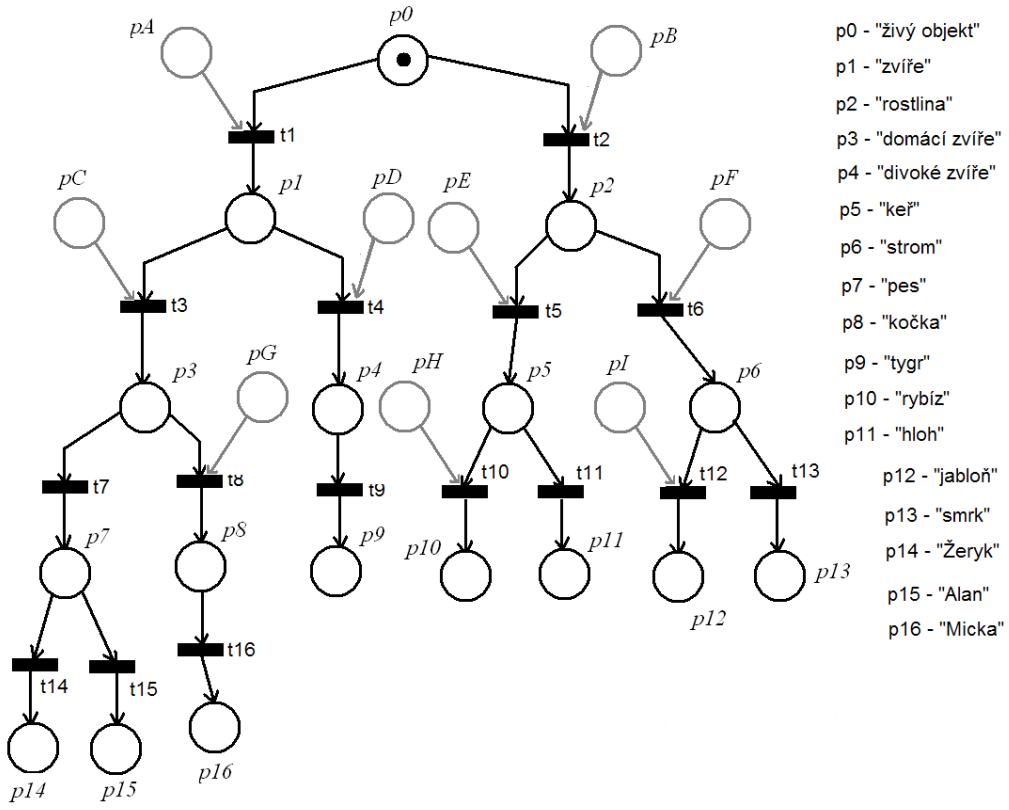
$F = \{<p_o,t_1>,<p_0,t_2>,<t_1,p_1>\ldots<t_{16},p_{16}>\}$

Místa určují aktuální stav (případ) M_k Petriho sítě N. Tedy konkrétní diagnostiku. Při dotazování na takovouto bázi je odpověď např. zkoumaný keř je označen jako hloh vyjádřena odpalem přechodu t_{11}. Cesta k tomuto poznatku je pak uložena ve formě přechodů t_2-t_5-t_{11}

Další výhodou popisu báze znalostí pomocí Petriho sítě je “spojitelnost” s inferenčním mechanismem. V dalších částech této kapitoly budou popsány způsoby

modelování inferenčního mechanismu pomocí Petriho sítí. V kapitole zabývající se jazyky PN bylo uvedeno, že pokud je L_1 jazykem PN a L_2 rovněž jazykem PN potom $L_1 + L_2$, stejně tak $L_1 * L_2$ jsou taktéž jazyky Petriho sítě. Tudíž spojení modelu báze znalostí v PN a inferenčního mechanismu rovněž v PN zachovává všechny podmínky dané Petriho sítěmi. Vytvoření báze znalostí a způsob vyhodnocování inferečním mechanismem bude demonstrován v následujících kapitolách.

O Petriho sítí na obr 5.2 lze říci, že je bezpečná, konzervativní, omezená, ale především se jedná o Petriho síť s volným výběrem. Vniká zde tedy spousta konfliktů odpalitelnosti. Proto by výsledná Petriho síť reprezentující znalosti měla vypadat následovně.



Obr 5.3 Petriho síť popisující bázi znalostí

Problém, který přechod bude odpálen je zde řešen pomocí pomocných míst $p_A \ldots p_I$ Postupná zpřesňování o jaký druh živého objektu se jedná, probíhá poté

formou dotazů. Na začátku je token jen v místě p_0 , zde je položen dotaz „Jedná se o rostlinu nebo zvíře.“ Podle odpovědi uživatele, (odpověď *ano* na otázku *Je to zvíře* je vyjádřena tokenem v p_A) je rozhodnuto jak budu postupovat dále. Tedy zda bude odpálen přechod t_1 nebo t_2. Po zodpovězení dotazu a odpalu příslušného přechodu pokračuje dotazování upřesňujícími dotazy.

Báze znalostí na předcházejících stránkách popisující určení druhu živého objektu, byla zjednodušena a sloužila pro popis způsobu modelování znalostí v Petriho sítích. Ovšem diagnostika živých organismů s využitím expertních systémů byla realizována např. v [11]. Jedná se o práci na Univerzitě v Canterbury, kde byl navržen expertní systém pro antropologickou analýzu, provádějící identifikaci popř. zařazení do příslušné třídy, čeledi, rodu, druhu, neznámých živočichů na základě druhotných znaků. Podle dostupných informací o daném organismu se zde využívá několikero algoritmů inference avšak báze znalostí by ve zjednodušené podobě mohla odpovídat výše uvedené. Odpovědi na výše zmíněné dotazování, symbolizované umístěním tokenu do pomocných míst jsou rovněž určovány na základě poznatků o daném živém objektu.

5.2 BÁZE ZNALOSTÍ PRO DIAGNOSTICKÝ EXPERTNÍ SYSTÉM

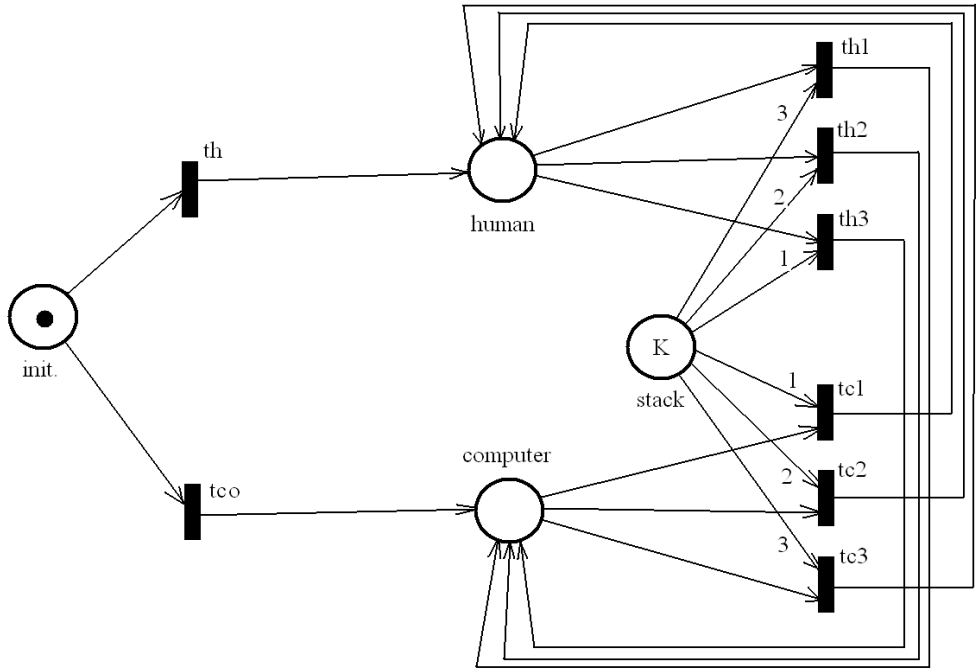
Většina v praxi nasazených expertních systémů je diagnostického charakteru. Úlohou diagnostických expertních systémů je provádět efektivní interpretaci dat s cílem určit, která z hypotéz z předem definované konečné množiny cílových hypotéz nejlépe koresponduje s reálnými daty týkajícími se daného konkrétního případu. U diagnostických systémů tedy řešení případu probíhá formou postupného ohodnocování a přehodnocování dílčích a cílových hypotéz v rámci víceméně pevně daného vnitřního strojového modelu řešeného problému, který je předem navržen expertem. [9]

Jádrem diagnostického systému je řídící mechanismus, který využitím báze znalostí v každém kroku upřesňuje aktuální model konzultovaného případu. Řídící

mechanismus je odpovědný za výběr dotazu a za úpravu aktuálního modelu po obdržení odpovědi. Báze dat může být obecně tvořena jak přímými odpověďmi uživatele, tak i hodnotami automaticky odečtenými z měřicích přístrojů nebo parametrické výstupy řízeného systému. Expertní systémy se v některých svých podobách používají též pro řízení. Uplatnění umělé inteligence je i v procesech, kde nemůžeme exaktně naměřit požadovaná data a poté je předepsaným způsob algoritmizovat.

5.2.1 Diagnostický expertní systém řešící logickou hru

V [3] jsem objevil zajímavý příklad možností jednoduchého expertního systému. Jeho činnost je demonstrována na strategické hře, ve které soupeří počítač a člověk ve hře "zápalky". Principem této hry je střídavé odebírání zápalek oběma hráči z hromádky. Hromádka může obsahovat libovolný počet zápalek. Hráč může odebrat ve svém tahu vždy jednu, dvě nebo tři zápalky. Prohrává ten, kdo odebere poslední zápalku. Jak bude popsáno dále, expertní systém slouží jako "poradce" pro člověka, zatímco "hrací" strategie počítače je náhodná, ale dá se vhodným způsobem upravovat. Je dokazatelné, že vítězná strategie u této hry se řídí velikostí zbytku po celočíselném dělení. Tzn. před každým tahem si spočítám počet zbývajících zápalek a vydělím ho 4. Je jasné, že zbytek po tomto celočíselném dělení čtyřmi bude 0, 1, 2 nebo 3. V případě, že zbytek po dělení 4 je 0, odeberu 3 sirky. Zůstanou – li mi 1 nebo 2 sirky – odeberu 1 a v případě, že po dělení zůstanou 3 sirky odeberu z hromádky 2 zápalky. Následující obrázek popisuje celou situaci pomocí PN.



Obr 5.4 Petriho síť znázorňující průběh strategické hry

PN zobrazená na obr 5.4 pouze modeluje průběh celé strategické hry. Na začátku je token v místě *init*. Místa *human* a *computer* odpovídají hráči, který je právě na tahu – tudíž odebírá sirky. Místo *stack* symbolizuje hromádku sirek. Jak je vidět, obsahuje K tokenů, kde K značí příslušný počet zápalek. Jak již bylo řečeno výše, počet zápalek na začátku hry není pevně dán a proto i místo *stack* neobsahuje pevně daný, konstatní počet tokenů.

Vstupní podmínky odpalitelnosti jsou v tomto případě splněny pouze u přechodů t_h a t_{co} a proto mohu odpálit pouze jeden z těchto 2 přechodů. Odpálení *th* symbolizuje, že začínající hráč je člověk, zatímco t_{co} odpovídá prvnímu tahu provedém počítačem. Zde se jedná o klasický konflikt proveditelnosti, který bude vyřešen v další části této kapitoly. Zde chci zatím jen popsat analogii celé problematiky zobrazenou v příslušné PN.

V případě, že odpálím přechod t_h, dostane se 1 token do místa *human* a já tedy mohu odpálit jeden z přechodů t_{h1}, t_{h2} nebo t_{h3}. Jak je vidět, tyto přechody odebírají ze zásobníku *stack* 1, 2 nebo 3 tokeny, což odpovídá příslušném počtu zápalek odebíraném z celkové hromádky (*stack*). Je zřejmé, že jeden z těchto přechodů mohu odpálit pouze v případě, že ve stacku jsou nějaké tokeny a že token obsahuje i místo human (hraje hráč člověk). Po odebrání 1, 2 nebo 3 zápalek (odpálení t_{h1}, t_{h2} nebo t_{h3}) je odbrán příslušný počet tokenů ze stacku a 1 token z místa human a zároveň je token dodán do místa computer. Nyní hraje počítač, protože token je v místě computer, mohu odpálit jeden z přechodů t_{c1}, t_{c2} nebo t_{c3}. Je to obdoba přechodů t_{h1}, t_{h2} a t_{h3} a symbolizují odebrání učitého počtu zápalek počítačem. Po odpálení jednoho z těchto přechodů (t_{c1}, t_{c2}, t_{c3}) se "hrací" token přesune opět do místa human a hraje člověk. Hra (postupné odpalování) pokračuje až do úplného odebrání všech zápalek (tokenů).

5.2.2 Řešení konfliktu odpalitelnosti

PN na obr. 5.4 pouze modelovala celý proces. Přesto však lze údaje z této sítě považovat za data, na základě kterých bude dále popsaný inferenční mechanismus bude určovat herní strategii.

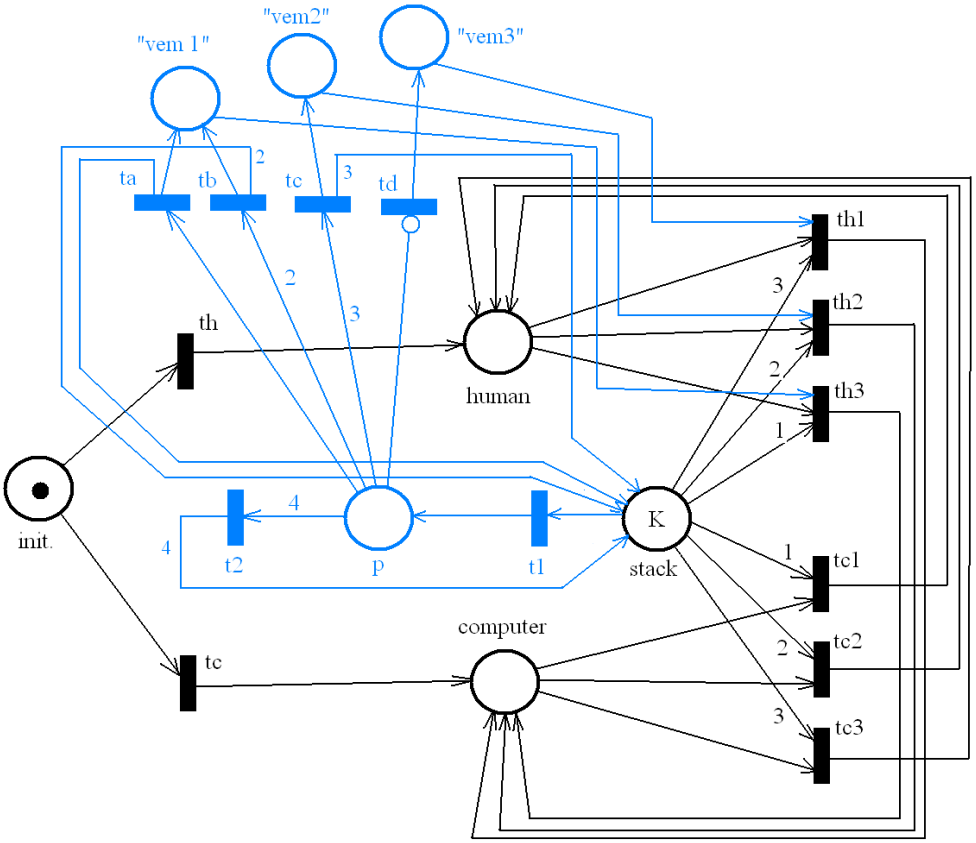
Už při samotném popisu se objevila celá řada otázek. Jak poznám, který hráč má začínat? Jak poznám kolik zápalek mám odebrat? Kde je implementován algoritmus vedoucí hráče human k vítězné strategii?

Teoretická řešení konfliktu proveditelnosti byla popsána ve 4. kapitole. Pro případ výše popsané strategické hry se nejlépe hodí synchronizovaná PN nebo časované PN. Řešení pomocí synchronizované PN popíši nyní.

5.2.3 Řešení pomocí synchronizovaných PN

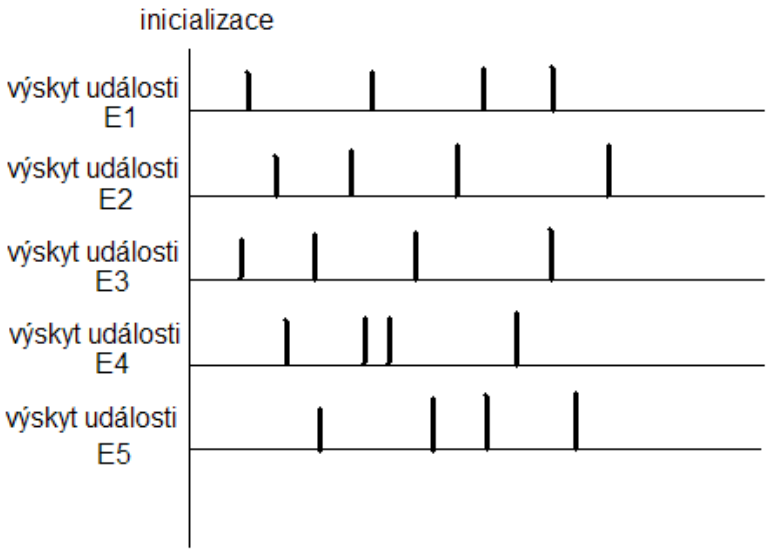
Jak již bylo řečeno v kapitole 4.5 podstatou synchronizovaných PN je, že každý přechod obsahuje tzv. sadu vnějších událostí (set of external events) což je vlastně nulový signál obsahující náhodné jednotkové pulsy. V případě, že se na

přechodu objeví takovýto puls, je testováno, zda je možné příslušný přechod odpálit a při splněných podmínkách odpalitelnosti je daný přechod odpálen. Podle obr. 5.5 se konflikt proveditelnosti vyskytuje u přechodů *th* a *tco* na začátku hry a potom u přechodů t_{c1}, t_{c2} a t_{c3}. Jedním z řešení je tedy synchronizovat příslušné přechody náhodnými signály obsahující jednotkové pulsy, který bude vlastně určovat, který z přechodů bude odpálen.



Obr. 5.5 Petriho síť obsahující synchronizované a prioritní přechody.

Priority jednotlivých přechodů $t_2 > t_c > t_b > t_a$



Obr. 5.6 Příslušné sady náhodných vnějších událostí (sets of external event)

Použitím synchronizovaných PN jsem vyřešil konflikt odpalitelnosti u toho, který z hráčů začne (odpal t_h nebo t_{co}) a kolik zápalek odebere počítač (t_{c1}, t_{c2} nebo t_{c3}). Na obr. 5.6 jsou sady náhodných událostí. Událost E_1 synchronizuje přechod t_h a událost E_2 přechod t_{co}. Jakmile je možné tyto přechody odpálit, systém počká zda dojde dříve k výskytu pulsu v E_1 nebo E_2 a odpálí příslušný přechod. V případě, že bychom požadovali, aby např. začínal hrát hráč human s dvakrát vyšší pravděpodobností než počítač, bylo by potřeba vygenerovat sadu událostí E_1 dvakrát hustší než E_2. V počítači bude mít E_1 tvar řady nul a jedniček, kde hodnota 1 symbolizuje výskyt události.

Zbývá ještě vypracovat expertní “radící” mechanismus, který člověka dovede k vítězství ve výše popsané hře.

Jak jsem již napsal v úvodu této podkapitoly, vítězná strategie spočívá v odebírání konkrétního počtu zápalek podle vydělení aktuálního množství 4.

Zbytek 0 = ber 3

Zbytek 1 nebo 2 = ber 1

Zbytek 3 = ber 2

Expertní "radící" mechanismus v bázi znalostí tvoří modrá část PN na obr 5.5. Před každým tahem hráče human (odpalem přechodů t_{h1}, t_{h2} nebo t_{h3}) je obsah stacku převeden do místa *p* (pomocí odpalování přechodu t_1). Počet všech tokenů je převeden do pomocného místa *p*. Z tohoto místa *p* jsou po 4 tokenech (postupným odpalováním t_2) vraceny tokenydo stacku, až zde zůstane pouze počet tokenů menší než 4. Je – li tento zbytek 1 nebo 2 mohu odpálit přechody t_a nebo t_b, což znamená, že system donutí člověka odebrat jednu zápalku. Token se přesune do místa "vem 1" a dovolí hráči human odpálit pouze přechod t_{h3}, který odebírá 1 token ze zásobníku stack. Zůstanou – li po postupném odpalování t_2 (odebírání po 4 tokenech) v místě p tři tokeny, je odpálen přechod t_c, který přesune token do místa "vem 2". Hráč human pak musí odpálit přechod t_{h2} – odebrat 2 zápalky ze zásobníku. V případě, že místo *p* zůstane prázdné, je možné odpálit pouze přechod t_d, protože obsahuje inhibitor a odpalem tohoto přechodu je hráč přiměn k odebrání 3 zápalek (odpal t_{h3}).

Pro zajištění správnosti fungování celého systému je ještě třeba vhodně nastavit priority jednotlivých přechodů, aby byly odpalovány tak jak potřebujeme a v námi zvoleném pořadí. Nejvyšší prioritu v "radící" modré části musí mít přechod t_2, Který bude odpalován tak dlouho, dokud v místě p nezůstane zbytek po dělení 4. Dále musí být priority přechodů $t_c > t_b > t_a$. Je to z toho důvodu, že pokud by v místě p zůstaly 3 tokeny bylo by možné odpálit všechny 3 tyto přechody, ale my požadujeme v tomto případě odpal přechodu tc. Prioritu td nemá smysl řešit, protože obsahuje orientovanou hranu s inhibitorem a tudíž jeho odpal je možný pouze za předpokladu nepřítomnosti tokenů v místě *p*, což automaticky vyloučí možnost odpalu přechodů t_a, t_b, t_c.

Obecně lze říci, že problém řešení strategické hry popsané v [3] pomocí exepertního systému v jazyku Clips je možné algoritmizovat pomocí PN. Jedním z řešení je použití PN vyšší úrovně a sice kombinaci synchronizované a prioritní PN (synchronized and prority Petri nets).

O výše uvedené Petriho sítí, popisující expertní systém řešící strategickou hru, lze říci, že není bezpečná. Existují 2 místa (*stack* a *p*), která v určitých chvílích obsahují více než 1 token. Dále lze říci, že Petriho je omezená – neexistuje stav, kdy

by po odpalování určité řady přechodů rovnoměrně rostl počet tokenů v určitém místě.

5.2.4 Báze znalostí

Báze znalostí výše popsaného expertního systému je tvořena přechody (jeho vlastnostmi), místy a hranami, které je spojují. Inferenční mechanismus potom pracuje tak, že v každém kroku testuje, které přechody může odpálit a na základě vlastností jednotlivých přechodů (priorita, synchronizace), vybere jeden, který odpálí.

5.2.5 Realizace v objektově orientovaném jazyce

Petriho síť lze popsat $N = \{P, T, F, W, K, M_0\}$, kde v tomto případě

$P = \{human, vem1, vem2, vem3, stack, p\}$

$T = \{t_{h1}, t_{h2}, t_{h3}, t_1, t_2, t_a, t_b, t_c, t_d\}$

$F = \{<human, t_{h1}>, <human, t_{h2}>, <human, t_{h3}>, <stack, t_{h1}>, <stack, t_{h2}>,$
$<stack, t_{h3}>, <stack, t_1>, <t_1, p_1>, <p_1, t_2>, <t_2, stack>, <p_1, t_a>, <p_1, t_b>$
$<p_1, t_c>, <p, t_d>, <t_a, vem1>, <t_b, vem1>, <t_c, vem2>, <t_d, vem3>, <vem1, t_{h1}>$
$<vem2, t_{h2}>, <vem3, t_{h3}>, <t_a, stack>, <t_b, stack>, <t_c, stack>\}$

Prototypy jednotlivých objektů by vypadaly následovně

```
class misto
{
     int pocet_tokenu;
     int kapacita = maxValue;
     bool  je_koncovy;
}
```

Každé místo lze charakterizovat počtem značek a kapacitou. Pokud není kapacita specifikována, je chápána jako neomezená. Proto je přednastavena na *maxValue*.

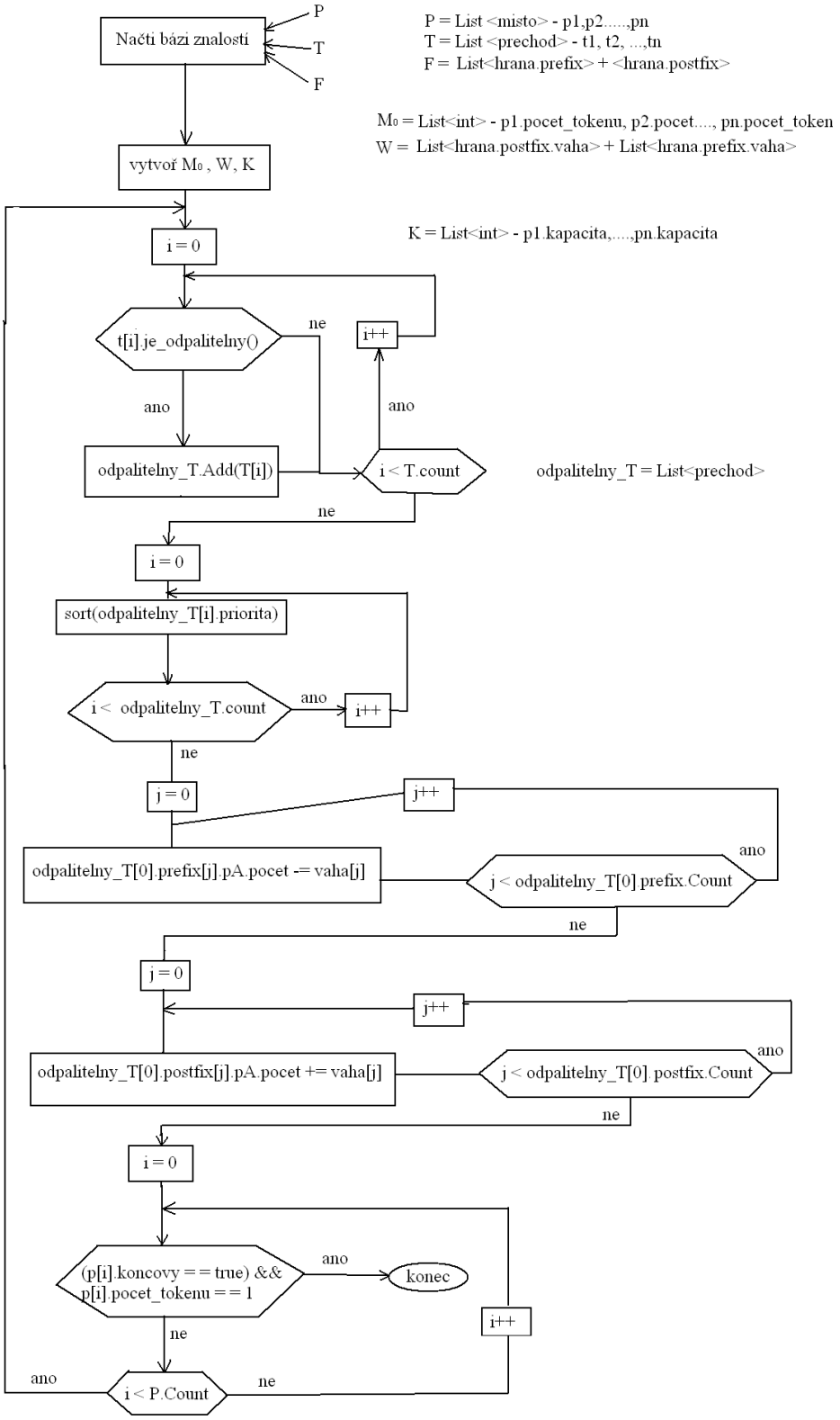
```
class prechod
{
     int priorita = 0 ;
     bool synchronizovany;
     bool posloupnost;
     bool je_odpalitelny();
     List <hrana> - prefix;
     List <hrana> - postfix
}
```

Přechodům lze nastavovat prioritu. Defaultně je nastavena na 0. To znamená, že pokud přiřadím prioritě přechodu nějaké kladné číslo, daný přechod získá vyšší prioritu než neoznačené. Dále je každý přechod označen, zda je odpalitelný nebo ne. Test odpalitelnosti probíhá na základě vstupních a výstupních míst a rovněž závisí na váze hran, kterou je s každým místem spojený. *Hrana – prefix* označuje sadu vstupních míst a *Hrana – postfix* sadu výstupních míst.

```
class hrana
{
     misto pA;
     int vaha = 1;
}
```

Každou hranu vedoucí z/do (prefix/postfix) přechodu, lze popsat místem, se kterým je spojen a váhou. Jestliže není u Petriho sítí vyznačena váha hrany celým číslem, předpokládá se, že je rovna 1.

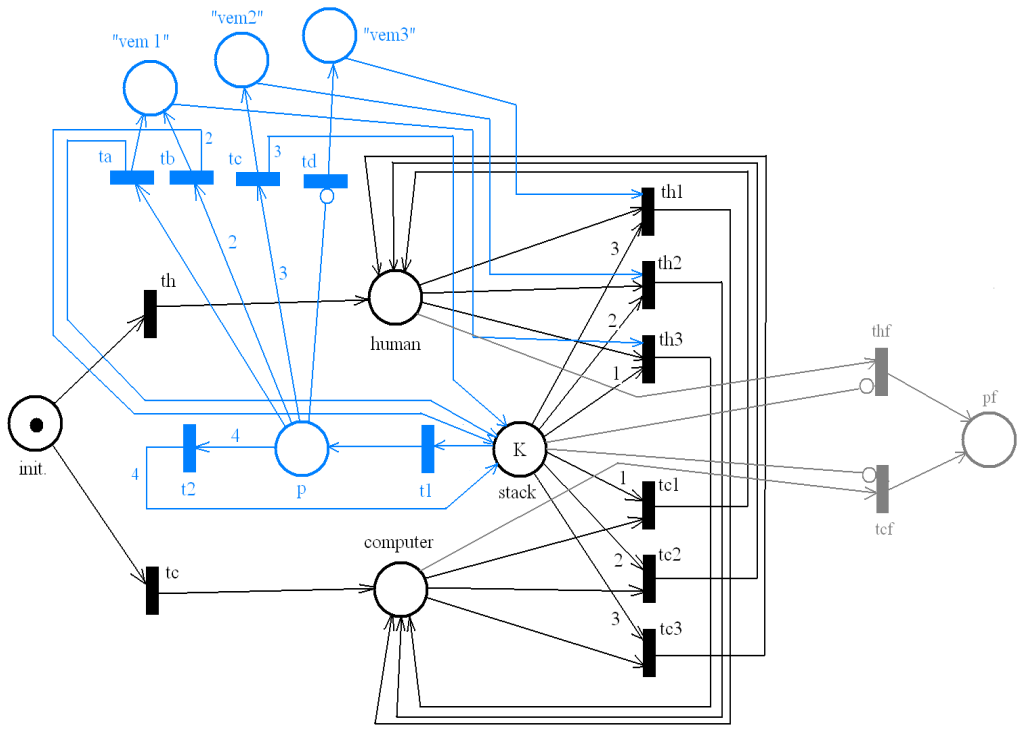
Jak již bylo řečeno, báze znalosti je tvořena místy, přechody a orientovanými hranami, tedy následující Obr.5.7 popisuje proces testování odpalitelnosti a odpalování jednotlivých přechodů


Načti bázi znalostí
P
T
F
P = List <misto> - p1,p2.....,pn
T = List <prechod> - t1, t2, ...,tn
F = List<hrana.prefix> + <hrana.postfix>
M0 = List<int> - p1.pocet_tokenu, p2.pocet...., pn.pocet_token
W = List<hrana.postfix.vaha> + List<hrana.prefix.vaha>
vytvoř M0 , W, K
K = List<int> - p1.kapacita,....,pn.kapacita
i = 0
t[i].je_odpalitelny()
ne
i++
ano
ano
odpalitelny_T.Add(T[i])
i < T.count
odpalitelny_T = List<prechod>
ne
i = 0
sort(odpalitelny_T[i].priorita)
i < odpalitelny_T.count
ano
i++
ne
j = 0
j++
odpalitelny_T[0].prefix[j].pA.pocet -= vaha[j]
j < odpalitelny_T[0].prefix.Count
ano
ne
j = 0
j++
odpalitelny_T[0].postfix[j].pA.pocet += vaha[j]
j < odpalitelny_T[0]. postfix.Count
ano
ne
i = 0
(p[i].koncovy = = true) &&
p[i].pocet_tokenu = = 1
ano
konec
ne
i++
ano
i < P.Count
ne

Vektor `odpalitelny_T` – obsahuje pouze odpalitelná místa. Ta jsou poté seřazena v bloku sort podle priority. Na místě `odpalitelny_T[0]` je tedy odpalitelný přechod s nejvyšší prioritou. Je odpálen a všem prefixovým místům je odebrán příslušný počet tokenů (podle váhy orientované hrany) a postfixovým místům je naopak příslušný počet tokenů přidán. Na závěr testuji, zda koncové místo neobsahuje token. Jestliže ano mechanismus ukončí činnost.

5.2.6 Jazyk typu L Petriho sítě popisující diagnostický expertní systém

Ve 2. kapitole jsem se zabýval jazyky Petriho sítí popisující příslušnou PN. Zaměřil jsem se především na jazyk typu L. Protože je dostatečně obecný a zároveň známý. Petriho sítě obecně vycházejí z konečných automatů. I jazyky generované Petriho sítí lze vygenerovat ekvivalentně zvoleným konečným automatem. Chceme – li Petriho síť popsat jazykem typu L, musí být v takovém tvaru kdy existuje předem definované počáteční a konečné místo. Na začátku se v počátečním místě nachází právě 1 token. Na konci se jediný token nachází v konečném místě. Jako počáteční místo můžeme zvolit místo init. Na konci hry se může token teoreticky nacházet jak v místě *human* tak v místě *computer.* Je tedy potřeba ve výsledku převézt token do jednoho předem definovaného konečného místa.



Obr. 5.8 Petriho síť generující jazyk typu L

Celou síť jsem doplnil o konečné místo pf a přechody thf a tcf. Stack je s těmito přechody spojen inhibitorem a tudíž je odpal těchto přechodů možný až když ve stacku není žádný token (hra je skončena). V případě, že se poslední token nachází v místě human (po posledním tahu počítače) odpálím přechod *thf* . Zbyl – li poslední token v místě computer, mohu odpálit pouze přechod *tcf*. Je jasné, že bude vždy odpalitelný pouze jeden z těchto 2 přechodů a proto zde není nutné řešit konflikt proveditelnosti jako v předešlých případech. Jazyk generovaný touto sítí pak bude vypadat následovně.

$$L = (t_{co}^{p} + t_h^{1-p}) * ((t_{c1} + t_{c2} + t_{c3}) + (t_{h1} + t_{h2} + t_{h3}) * t_1^{K} * t2^{K/4} * (t_a + t_b + t_c + t_d))^{n} * (thf^{q} + tcf^{1-q}) \mid K => 2*n + 1 > 0,\ K -> 0\ \ p = \{0;1\}\ q = \{0;1\})$$

Znaménka „+“ ve výše uvedeném jazyku L značí, že může být odpálen jeden přechod z několika. Jak je možné si všimnout, jedná se o přechody u kterých byl řešen konflikt odpalitelnosti, ať už pomocí synchronizovaných nebo prioritních PN. Exponent n značí, že daná sada přechodů proběhne n-krát. Je dokazatelné, že

$K => 2*n$

Kde K je počáteční počet zápalek a n počet tahů hráče, který hraje jako druhý. V případě, že by začínající rovněž skončil a hráči by po celou dobu hry odebírali konstantně 1 sirku byl by počet

$K = 2 * n + 1$

Což odpovídá lichému počtu sirek na počátku hry. Výraz $t2^{K/4}$ značí že daný přechod bude proveden K/4 (jedná se zde samozřejmě o celočíselné dělení) v každém tahu. V n tazích bude poté daný přechod proveden K / 4 *n.

Přechod *t*co bude odpálen p-krát a t_h (1-p)- krát pro p = {0;1}. Je tedy jasné, že jeden z těchto přechodů bude odpálen jedenkrát a druhý vůbec. Tento fakt symbolizuje, že pouze jeden hráč může začínat. Stejně tak přechody t_{hf} a t_{cf}. Kde poslední odpálený přechod exaktně označuje vítěze – odpal t_{hf} – vyhrál člověk, odpal tcf – vyhrál počítač.

5.2.7 Shrnutí použití dané PN

V této kapitole jsem se zabýval především řešení konfliktu odpalitelnosti u konkrétní PN. Použitím synchronizované sítě, expertní část je pak realizovaná pomocí sady přechodů z vhodně nastavenými prioritami odpalu, aby bylo zajištěno požadované odpálení požadovaného přechodu ve správný okamžik a celý systém se tzv. nezacyklil popř. nedostal do mrtvého stavu (deadlock), což značí, že není možné odpálit již žádný přechod a daná PN se stává mrtvou, aniž by popsala celý proces.

5.3 BÁZE ZNALOSTÍ PRO PLÁNOVACÍ EXPERTNÍ SYSTÉM

Úlohy umělé inteligence jsou obvykle reprezentovány řešením úloh, kde je na počátku k dispozici generátor možných řešení a na základě předem daného kritéria optimality je vybráno to nejvhodnější. Řešením rozumíme nalezení takové posloupnosti, které vede od požadovaného stavu k cíli. Zde se používají metody prohledávání stavového prostoru, kdy každý bod stavového prostoru odpovídá jedné situaci, která může v reálném světě vzniknout[9]. U Petriho sítí je stavový prostor reprezentován stromem pokrytelnosti. Tedy schématickou množinou stavů, která může u dané sítě nastat. Jednotlivé body stavového prostoru jsou vlastně vektory míst ohodnoceny počtem tokenů $M_k(p_i)$, ze kterých se daný strom skládá. Okamžitý k-tý stav systému je reprezentován vektorem M_k. Prohledávání stavového prostoru zpravidla začíná v dané počáteční situaci a postupuje směrem k cíli aplikací posloupností přípustných operátorů. Přípustnost konkrétního posunu z jednoho definovaného stavu do jiného je vyjádřena podmínkou odpalitelnosti přechodu nacházejícího se mezi 2 místy odpovídající konkrétním stavům.

Plánování je tedy vlastně postupné procházení stavového prostoru. Je jasné, že s rostoucí složitostí daného systému poroste robustnost popisujícího stavového prostoru a prohledávání bude čím dál složitější, ne-li nemožné. Proto je důležité mít možnost u takovýchto složitějších systémů pracovat nad menšími (dílčími) úlohami samostatně a dílčí výsledky poté zkombinovat do celkového jediného řešení [8]. Pokud bychom nebyli schopni takovýto rozklad nalézt, je velmi pravděpodobné, že úloha bude neřešitelná vlivem obrovského množství dat a podmínek, jež musíme v každém kroku uvažovat.

Mezi zjednodušující kroky při prohledávání stavového prostoru patří odstranění propočítávání celého stavu úlohy na stav nový při každém kroku. Místo toho nám bude stačit evidovat pouze změny stavu, které aplikace příslušného operátoru (tedy odpalu příslušného přechodu) vyvolá.

5.3.1 Výběr vhodného pravidla

Mezi nejčastěji používané postupy pro výběr pravidla patří metoda, kdy se nejprve určí množina rozdílů mezi požadovaným cílovým stavem a stavem aktuálním a pak se identifikují ta pravidla, která jsou pro redukci těchto rozdílů významná. Máme-li systém popsaný Petriho sítěmi a příslušný strom pokrytelnosti, kde je vyznačen počáteční a cílový stav, není těžké za pomoci incidenční matice stanovit následující stav po odpalu jednotlivých přechodů a určit, zda se tímto odpalem přiblížíme k požadovanému cíli či nikoliv.

5.3.2 Detekce cílových cest

Řešení plánovací úlohy je považováno za úspěšné, je – li nalezena posloupnost operátorů, které transformuje počáteční stav úlohy na stav cílový [9]. Taková posloupnost je u Petriho sítí reprezentována posloupností odpálených přechodů. V kapitole 2 byly popsány jazyky Petriho sítí jako cesta od počátečního stavu k cílovému. Lze tedy říci, že plánování je vlastně stavový popis jazyka Petriho sítě. S tím rozdílem, že při plánování nejde jen o dosažení cílového (požadovaného) stavu, ale rovněž o dosažení tohoto stavu co nejkratší a nejjednodušší cestou.

5.3.3 Řešení úlohy umělé inteligence plánovací metodou

Jednou z klasických úloh umělé inteligence je, že máme k dispozici n nádob o předem definovaných objemech. Je znám počáteční stav systému, tedy množství vody v jednotlivých nádobách, cílový stav, tedy požadované množství vody v jednotlivých nádobách a množina přípustných úkolů, kterými se můžeme z počátečního stavu transformovat do cílového. Takovými úkony jsou možnosti úplného naplnění nebo vyprázdnění nádoby, popř. přelití určitého množství vody mezi 2 nádobami. Pomocí Petriho sítí tento problém nejprve popíši obecně pro

nádoby o předem neznámém objemu a poté vše demonstruji na jednoduchém konkrétním příkladu.

Je-li k dispozici více cest, jak transformovat počáteční stav na cílový, je důležité podle určitého kritéria vybrat optimální cestu. Často bývá takovým kritériem optimálnosti nejmenší počet kroků.

U Petriho sítí bývá stav systému popsaný vektorem M, který popisuje aktuální počet tokenů na všech jeho místech. Lze popsat

počáteční stav jako $M_0 = (p_{10}, p_{20}, p_{30}, \ldots, p_{n0})$

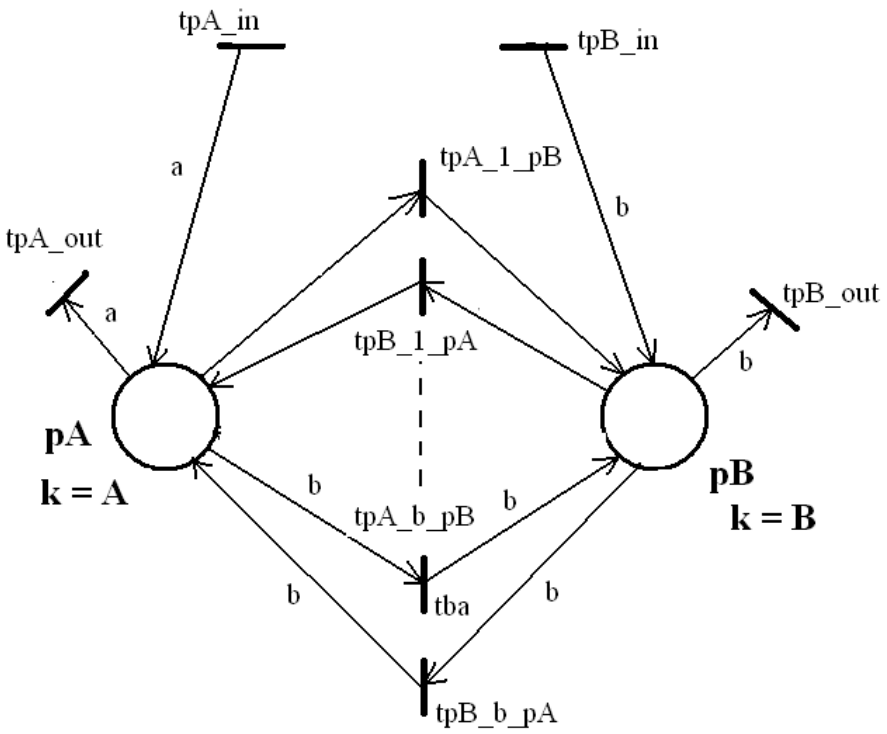
a konečný stav jako $M_f = (p_{1f}, p_{2f}, \ldots, p_{nf})$

kde p_{n0} – počet tokenů v místě pn v počátečním stavu

p_{nf} – počet tokenů v místě pn v konečném stavu

Množina všech přípustných operací, které lze nad danou Petriho sítí provést odpovídá množině přechodů, které lze v daném stavu odpálit.

Mějme tedy obecně n nádob P_1 - P_n. Je zřejmé, aby úloha měla smysl, musí být n rovno minimálně 2. Petriho síť popisující možné operace mezi dvěma libovolnými nádobami je znázorněna na následujícím obrázku.



Obr. 5.9 Petriho síť popisující systém 2 nádob

Místa p_A a p_B značí konkrétní nádoby o objemu *A* a *B* litrů.. Místa jsou kapacitně omezeny ,což symbolizuje omezení objemu jednotlivých nádob.

Přechod t_{pA_in} = naplnění nádoby p_A objemem A litrů

Přechod t_{pA_out} = úplné vylití nádoby *A*

Přechod $t_{pA_1_pB}$ = přelití 1 litru z p_A do p_B.

Považujeme – li nádobu p_A větší nebo rovnu p_B, je jasné že mezi nimi lze přelít nanejvýš *B* litrů. Obecně lze říci, že mezi dvěma libovolnými nádobami lze přelít nanejvýš takové množství vody odpovídající objemu menší nádoby.

Každá nádoba je charakterizována určitým počátečním stavem, cílovým stavem a kapacitou. V objektově orientovaném jazyce by byla reprezentována třídou

```
Class misto
{
int pocatek;        //  počáteční stav
int cil;            //   cílový stav
int kap;            //    kapacita
}
```

5.3.4 Báze faktů

Báze faktů takového plánovacího systému obsahuje vektor M_0, který obsahuje počáteční hodnoty všech nádob. Pro n nádob má $M_0 - n$ prvků.

Dále vektor Mf , ve kterém jsou uloženy cílové stavy jednotlivých nádob a vektor Mkap, kde můžeme nalézt konečné kapacity každé nádoby.

<u>Báze faktů:</u>

$Mo = (P_{10}, P_{20}, \ldots, Pn_0)$ kde Pn_0 = počáteční stav n – té nádoby

Cílový stav $M_f = (P1_f, P2_f, P3_f, \ldots, Pn_f)$ kde $P_{nf} = Pn_{.cil}$;

Kapacity všech nádob $M_{kap} = (P1_{kap}, P2_{kap}, \ldots, Pn_{kap})$

Báze faktů tedy určuje výchozí a cílový stav a omezující podmínky (v podobě kapacit nádob).

5.3.5 Báze znalostí

Báze znalostí je u Petriho sítí vyjádřena přechody, které lze odpálit. Jedná se tedy o množinu přípustných operací. Nádoby lze naplnit (pokud je prázdná) nebo vylít pokud je plná. Obecně lze proces plnění/vyprazdňování zapsat jako odpal přechodů

- t_{px_in} - naplnění nádoby p_x
- t_{px_out} – vyprázdnění nádoby p_x

Dále je možné přelévat vodu mezi nádobami (pokud je v první dostatek vody ve druhé dostatek místa). Opět je příslušná operace znázorněna odpalem přechodu.

- $t_{px_n_py}$ – tento přechod značí přelití n litrů z nádoby px do nádoby py.

Petriho síť popisující celou problematiku by vypadala tak, že všechny místa px modelující nádoby budou mít přechod pomocí nichž je možno je plnit t_{px_in}, vylévat – t_{px_out} a zároveň budou s ostaními místy spojeny množinou přechodů umožňujících přelévání.

Báze znalostí tedy obsahuje přechody, což jsou vlastně vhodně zakodované příkazy co lze s daným systémem provádět. Zapsáno pomocí metod, které lze s danou třídou nádoba provést. Je jasné, že konkrétní čísla (jaké množství vody můžu nalít nebo vylít z jednotlivé nádoby) je určeno kapacitami jednotlivých nádob, proto bude báze znalostí vytvořena na základě báze faktů, především vektoru M_{kap}

5.3.6 Incidenční matice

Incidenční matice popisuje vliv jednotlivých přechodů (sloupce) na jednotlivá místa (řádky). Konkrétní číslo na pozici *[x, y]* značí změnu *x* – tého místa (v našem případě nádoby) po odpalu *y* – tého přechodu v tokenech (pro náš případ v litrech).

Předpokládáme – li že máme p_1 – p_n nádob seřazených podle velikosti objemu, při čemž nádoba p_n má největší objem, lze tedy mezi nádobami obecně přelít nanejvýš objem vody odpovídající druhé největší nádobě p_{n-1}. Proto poslední přechody v bázi znalostí budou $t_{pn_n-1_pn-1}$ a $t_{pn-1_n-1_pn}$ incidenční matice by poté vypadala následovně

$$
\begin{array}{c}
 \\ P1 \\ P2 \\ . \\ . \\ . \\ Pn
\end{array}
\begin{array}{c}
\begin{array}{cccccccc}
t_p1_in & t_p1_out & \ldots t_pn_in & t_pn_out & t_p1_1_p2 & t_p2_1_p1 & \ldots t_pn_n\text{-}1_pn\text{-}1 & t_pn\text{-}1_n_pn
\end{array} \\
\begin{bmatrix}
(p1) & -(p1) & 0 & 0 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & +1 & -1 & 0 & 0 \\
 & & & & & & & \\
 & & & & & & & \\
0 & 0 & (pn) & -(pn) & 0 & 0 & (n\text{-}1) & -(n\text{-}1)
\end{bmatrix}
\end{array}
$$

Je zřejmé, že s rostoucím počtem nádob a jejich objemem bude incidenční matice velmi robustní, nicméně je dokazatelné, že pro dvojice nádob s menším objemem bude většina přechodů nikdy neodpalitelných (tzv. neživých) a tudíž se incidenční matice značně zjednoduší.

Rozměry incidenční matice lze vypočítat následovně. Počet řádků matice je evidentně roven počtu nádob. Aby měla úloha smysl musí mít matice nejméně 2 řádky. Počet sloupců je roven dvojnásobku počtu nádob, protože s každou nádobou lze provést 2 operace (naplnění a vyprázdnění) a dvojnásobku počtu všech možných dvojic vynásobených kapacitou menší z nich.

Incidenční matice [x ,y]

x – počet všech nádob

y – 2 * počet všech možných dvojic nádob * kapacita menší z nich

Mějme například 3 nádoby o vjemech 2, 4 a 7 litrů. Počet řádků je v tomto případě roven 3. Ze tří nádob můžeme vytvořit 3 různé dvojice bez ohledu na pořadí – [2,4] [2,7] a [4,7] Počet sloupců je tedy roven 2 *3 + 2 * 2 + 2 * 2 + 2 * 4 = 22.

2 * 3 – dvojnásobek počtu nádob

2 * 2 – 2 menší kapacita z dvojice 2,4

2 * 2 – 2 menší kapacita z dvojice 2, 7

2 * 4 – 4 menší kapacita z dvojice 4, 7

Na výše uvedené incidenční matice je možno vidět, že přechody modelující plnění a vylévání první nádoby jsou na pozicích [0, 0] [0, 1]. Pro druhou nádobu by to bylo na pozicích [1,2] [1,3] atd. Obecně lze říci, že přechody realizující plnění a vypouštění *n*- té nádoby se v incidenční matici nacházejí na místech $[n, 2 \cdot n]$ a $[n, 2 \cdot n + 1]$. Načteme – li tedy z báze znalostí např. přechod t_{p8_in}, víme že se jedná o plnění 8. nádoby a jeho pozice v incidenční matici bude [8, 16] na kterém bude uložena hodnota kapacity 8. nádoby. Takže v bázi znalostí jsou uloženy přechody, které po „dekódování“ za pomocí incidenční matice mění konkrétní hodnotu aktuálního stavu.

5.3.7 Realizace principu prohledávání stavového prostoru

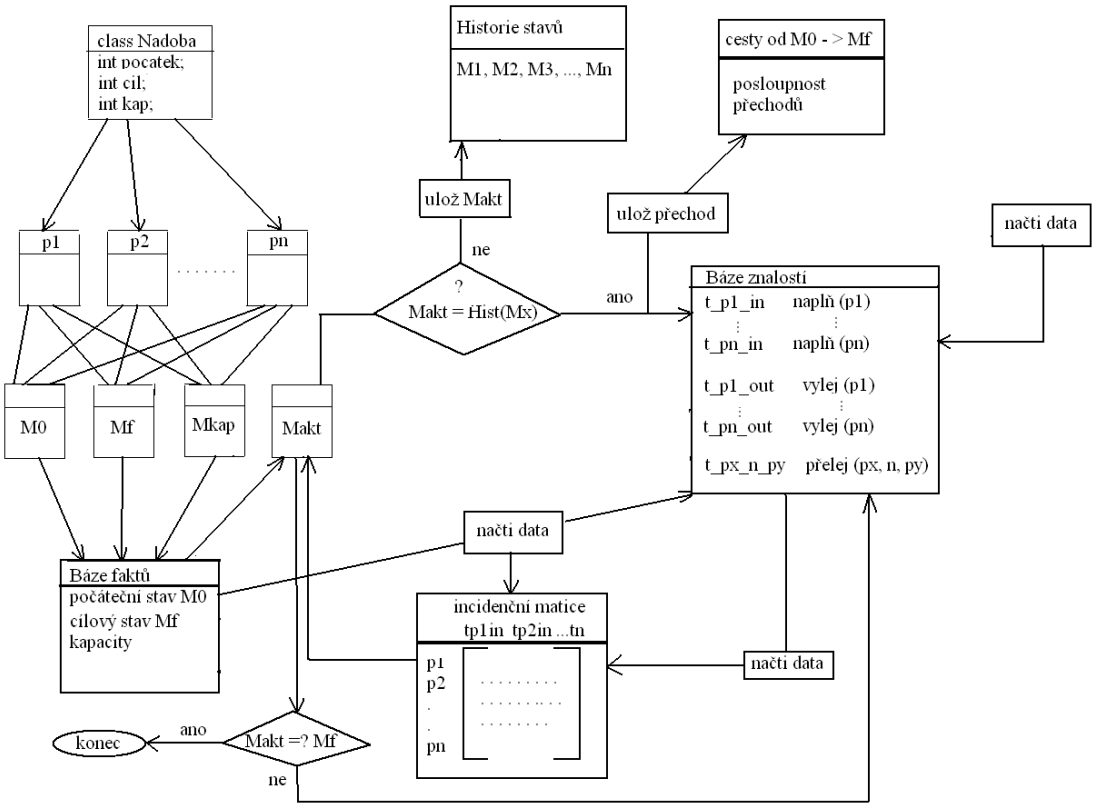
Nyní už máme vytvořenou bázi znalostí, ve které jsou vhodným předepsaným způsobem uloženy všechny možnosti jak můžeme počáteční stav transformovat na cílový. V bázi faktů je uložen počáteční a cílový stav. Je tedy možné z počátečního stavu s využitím znalostí a jejich převedení pomocí incidenční matice na konkrétní číslo pokračovat do dalších možných stavů a tento neustále porovnávat s cílovým. Jakmile se aktuální stav rovná cílovému plánování je ukončeno a cesta od počátečního přechodu k cílovému je reprezentována množinou přechodů, které byli odpáleny abychom se do cílového stavu dostali.

V několika publikacích [3][9][12] jsou popsány metody řešení problematiky plánovací úlohy pomocí několika způsobů. Zde bude popsán způsob prohledávání stavového prostoru za předpokladu, že se stav světa nesmí opakovat. Omezující podmínky jsou nezbytně nutné, protože jinak by se celé prohledávání mohlo omezit na „přelévání“ mezi dvěma určitými stavy a celý systém by se k výsledku nikdy nedobral.

Princip prohledávání stavového prostoru ze předpokladu, že se stav světa nesmí opakovat spočívá v nutnosti doplnit celý systém o stavový prostor, do kterého se ukládají všechny stavy, kterých bylo v průběhu plánování dosaženo a neustálým

porovnáváním aktuálního stavu s historií stavů je zamezeno dostat se znovu do duplicitního stavu. Vzhledem k tomu, že úloha s nádobami je reverzibilní, tedy ke každé operaci existuje operace inverzní (naplnění x vyprázdění, přelití z *A* do *B* x z *B* do *A*), je možné při opakovaném dosažení stavu „vycouvat“ pomocí inverzní operace zpět a z báze znalostí použít jiný přechod (jinou možnost posunu do jiného stavu).

Následující diagram popisuje celý proces prohledávání stavového prostoru.



Obr. 5.10 Blokové schéma popisující prohledávání stavového prostoru

Třída Nadoba (místo - place) definuje objekty typu Nadoba. Jak je vidět z obrázku je možné vytvořit až *n* – instancí této třídy. Po vytvoření objektů (nádob) je vygenerován vektor M_0, M_f, a M_{kap}. Jedná se o pole o velikosti *n* (počet objektů),

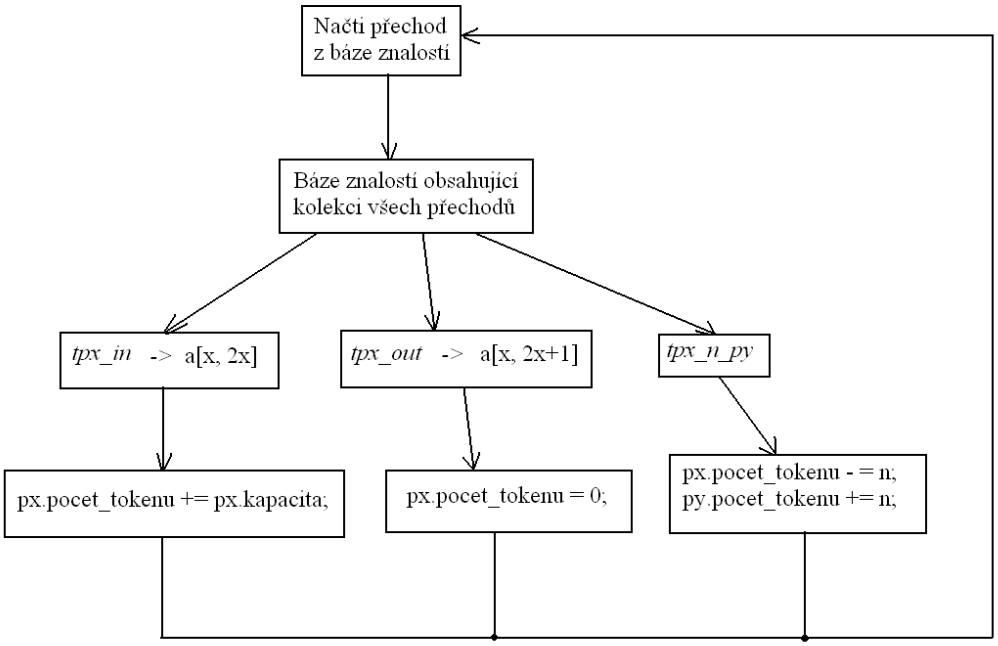
ve kterých je na příslušných místech uložen příslušný stav odpovídající nádoby. Vektory jsou uloženy v bázi faktů. Na základě báze faktů (především kapacit) je vytvořena znalostní báze – reprezentovaná množinou přechodů, které lze odpálit a incidenční matice, která převádí konkrétní znalost na číslo a mění tak aktuální stav celého systému. Jak je vidět z obrázku je báze znalostí načtena podle báze faktů a příslušně „zakodována“ podle uvedených pravidel, lze však načíst jinou bázi znalostí. např pokud chceme dojít z počátečního stavu do cílového aniž bychom museli plnit nádobu P_x – provedeme to tak, že načteme novou bázi znalostí, která neobsahuje přechod t_{px_in} – čímž vyloučíme pokus rozhodovacího mechanismu plnit nádobu P_x. Jakmile je na základě přípustných operací změněn aktuální stav celého systému, dojde k jeho srovnání s historií stavů, pokud je již v historii obsažen, pomocí inverzní operace je vrácen zpět a poté je zkoumána následující možnost operace. Zároveň je aktuální stav porovnáván s cílovým. Dojde – li ke shodě je prohledávání ukončeno. V bloku posloupnosti přechodů jsou uloženy sady přechodů vedoucí od počátečního stavu k cílovému.

5.3.8 Převod přechodu v bázi znalostí na prvek incidenční matice

Nejdůležitějším poznatkem této části je „převod“ libovolného přechodu uloženého v bázi znalostí na prvek incidenční matice, jehož pozice nám zároveň udává vliv na konkrétní místo (místa). Mějme tedy místo definované jako objekt

```
class misto
{
     int pocet_tokenu;
     int cil;
     int kap;
}
```

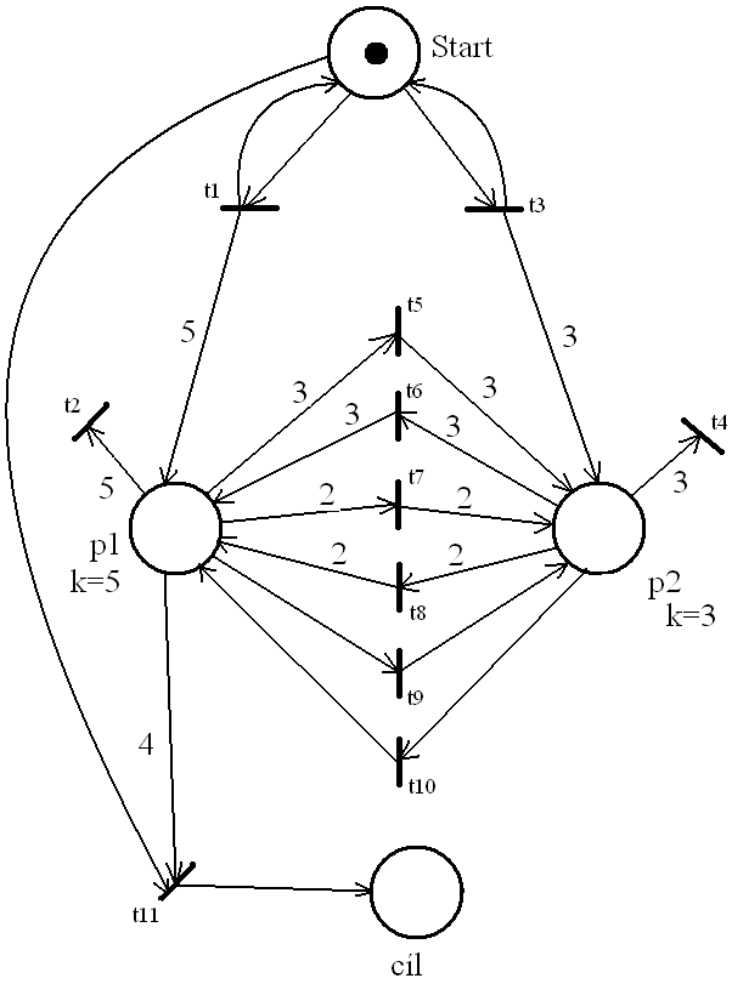
A incidenční matici A s prvky $a[i, j]$ popsanou již dříve. Pak pro p_x a p_y, které můžeme prohlásit za libovolné objekty typu *misto* platí



Obr. 5.11 Interpretace přechodů v bázi znalostí

5.3.9 Příklad pro 2 nádoby o objemech 3 a 5 l

Celou obecnou problematiku zde popíši na konkrétním příkladu. K dispozici jsou 2 nádoby, které mají obsah 5 a 3 litry. Úkolem je odměřit přesně 4 litry. Je jasné, že 4 litry mohou být odměřeny pouze v nádobě o objemu 5l.

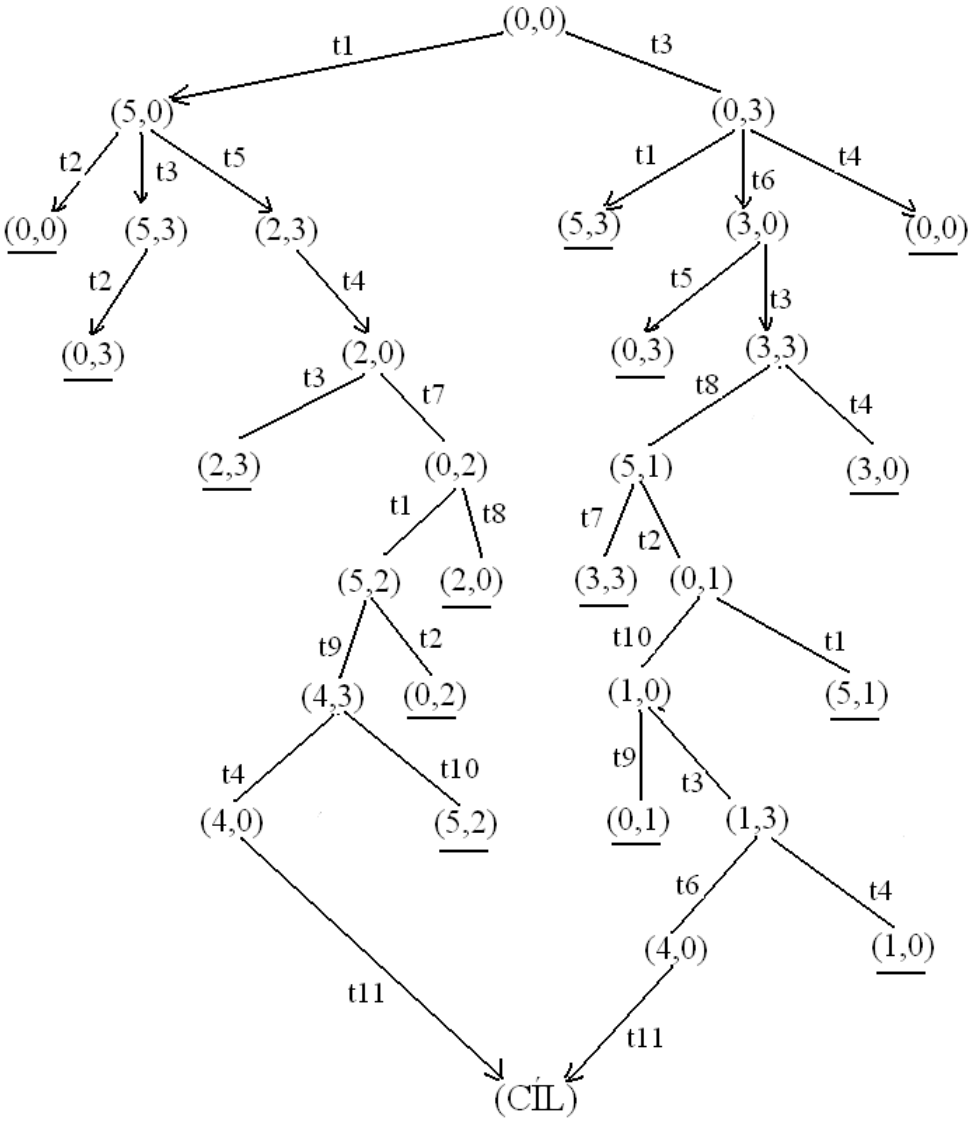


Obr. 5.12 Petriho síť popisující expertní systém 2 nádob

Místo „Start“ slouží jako inicializační stav, Cíl je konečný stav a p_1 a p_2 odpovídají jednotlivým nádržím. Jak je patrno z obrázku p_1 a p_2 jsou kapacitně omezeny, což odpovídá kapacitnímu omezení jednotlivým nádržím. Odpálením přechodu t1 dojde k naplnění nádrže p_1, odpálením přechodu t_3 dojde k naplnění nádrže p_2. Jak je vidět při odpálení přechodu t_1 popř. t_3 je vždy jeden token vrácen do Startu, aby bylo možno nádrže naplňovat opakovaně (neboli odpalovat přechody t_1 popř. t_3) jak je požadováno v zadání. Naopak přechody t_2 event. t_4 slouží k vyprázdnění příslušné nádrže.

Přechody t_5, t_6, t_7, t_8, t_9, t_{10} slouží k „přelévání“ obsahu mezi nádržemi. Jakmile se v nádrži p1 objeví 4 litry – může být odpálen přechod t_{11} a tím pádem je dosaženo cílového stavu. PN umožňují nastavit jednotlivým přechodům různé priority, proto můžeme nastavit přechodu t_{11} vyšší prioritu, aby jakmile to bude možné byl odpálen a my se z požadovaného stavu nedostali zpět. Zároveň je při odpalu t_{11} trvale odebrán token ze Startu, aby nádrže nemohli být opakovaně plněny při dosažení požadované hodnoty. Na následující straně je stavové schéma.

Jak je možné si všimnout, schéma popisuje pouze stavy na dvou místech, a to v nádržích p_1 (první číslo) a p_2 (druhé číslo). Jak již bylo řečeno v úvodu této podkapitoly, jedním ze způsobů zjednodušení stavového popisu složitějších procesů je nezobrazovat všechny stavy, ale pouze ty, které chceme měnit a jejichž změny potřebujeme sledovat. V našem případě jsou to pouze místa p_1 a p_2, které odpovídají jednotlivým nádržím. Zobrazování stavu místa Start (který má po celou dobu vykonávání procesu hodnotu 1 až do odpálení přechodu t_{11}, kdy je nastaven na 0) a místa Cíl (který má po celou dobu vykonávání procesu hodnotu 0 až do odpálení přechodu t_{11} kdy je nastaven na 1) považuji za zbytečné a celé schéma by to jen znepřehlednilo. Celý proces dosažení cílového stavu popisuje následující strom pokrytelnosti.



Obr. 5.13 Strom pokrytelnosti

(x,y) – *x* – počet litrů v 1.nádrži *y* – počet litrů ve 2. nádrži

Ze stromu pokrytelnosti je vidět, že celý proces plnění začíná prázdnými nádobami (0,0). Nyní můžeme naplnit první nádobu (5,0) nebo druhou (0,3). Podtržení u některých stavů značí, že téhož stavu bylo dosaženo již dříve a proto jeho opětovné

začlenění do celého stromu by dosažení potřebného řešení pouze zkomplikovalo a oddálilo.

Incidenční matice výše uvedené Petriho sítě by vypadala následovně.

$$
\begin{array}{c}
 \\ \text{start} \\ \text{p1} \\ \text{p2} \\ \text{cíl}
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} \text{t1} & \text{t2} & \text{t3} & \text{t4} & \text{t5} & \text{t6} & \text{t7} & \text{t8} & \text{t9} & \text{t10} & \text{t11} \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
5 & -5 & 0 & 0 & -3 & 3 & -2 & 2 & -1 & 1 & -4 \\
0 & 0 & 3 & -3 & 3 & -3 & 2 & -2 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
$$

Počet sloupců incidenční matice byl vypočítán jako 2 * každá dvojice * kapacita menší nádoby + 2 * počet nádob.

Tedy musí platit 2 * 2 + 2 * 3 + 1 pomocné místo přechod do cíle = 11.

Pokud je systém ve stavu *(2,0)* a odpálím přechod t_3 – z incidenční matice je patrné, že odpalem přechodu t_3 dodám do místa p_2 tři tokeny a dostanu tudíž duplicitní stav (2,3). K cílovému stavu tedy vedou posloupnosti t_1-t_5-t_4-t_7-t_1-t_9-t_4-t_{11} nebo t_3-t_6-t_3-t_8-t_2-t_{10}-t_3-t_6-t_{11}. Z uvedeného vyplývá, že první způsob vede k požadovanému cíli o krok dříve. Je tedy vhodnější.

5.4 SHRNUTÍ POUŽITÍ PETRIHO SÍTÍ PRO TVORBU BÁZE ZNALOSTÍ EXPERTNÍHO SYSTÉMU

V této kapitole byly popsány základní části expertního systému využívající Petriho sítí.

Diagnostický expertního systému vyžaduje použití PN vyšší úrovně. Jádrem takového systému je báze dat nebo výsledky měření na základě, kterých inferenční mechanismus upravuje svoje podmínky. Báze dat je popsána jednoduchou PN a

podle výsledků se mění způsob vyhodnocování inferenčním mechanismem. U inferečního mechanismu není žádoucí konflikt odpalitelnosti, který může být řešen PN se změněnou prioritou, popř. synchronizovaným nebo časovanými PN. V principu jde o to, předem definovat pokud se systém dostane do určitého stavu jak má postupovat dále. PN umožňují na různé vstupní parametry reagovat různě. Což je hlavním rysem diagnostických expertních systémů. V příkladě se zápalkami bylo vidět, jak se měnil počet zápalek v zásobníku měnila se i cesta PN, kudy celý modelovaný proces procházel.

Báze znalostí pro expertní systém plánovacího charakteru – jedna z definic [3] říká, že plánovací mechanismy pracují na principu prohledávání stavového prostoru a neustálého porovnávání aktuálního stavu s cílovým stavem. Stavový prostor je v prostředí PN zobrazen jako strom pokrytelnosti, následující stavy mohu určit ze znalosti aktuálního stavu a incidenční matice. Potom již není složité určit cestu od počátečního stavu k cílovému. Tato cesta je poté popsána množinou přechodů, které je nutné odpálit abychom se dostali do cílového stavu. Jestliže platí, že každý přechod reprezentuje určitou podmínku, pak výsledkem plánovacího systému je množina podmínek (postup) jejichž splnění vede k požadovanému cíli. V příkladu s nádobami to bylo (naplň nádobu 1, přelej 2 litry do 2, …atd).

Zatímco u plánovacího ES je důležité projít stavový prostor u diagnostického je nutné zkoumat veškeré eventuality, které mohou nastat a explicitně systému definovat jak se má chovat v neurčitých stavech.

6. ZÁVĚR

Petriho sítě mají výraznou modelovací a rozhodovací schopnost. Zvýrazněním jedné dojde k potlačení druhé. Jednoduché Petriho sítě mají především modelovací schopnost, použitím Petriho sítí vyšší úrovně získáme lepší rozhodovací schopnost.

Při realizaci báze znalostí pro expertní systém plánovacího charakteru je využíváno především modelovacích schopností Petriho sítě. Je znám počáteční stav systému, konečný stav a množina operací, které mohu provádět. Modelovací schopnost je zde důležitější, protože potřebuji prohledat celý stavový prostor a zjistit jak se nejlépe (např. v co nejkratší době) dostanu z požadovaného stavu do cíle. Stavový prostor je reprezentován stromem dosažitelných značení, báze znalostí je vyjádřena sadou přechodů, které podle incidenční matice ovlivňují počet tokenů v konkrétních místech. Mění tedy určitou část vektoru popisující aktuální stav a lze v každém kroku porovnat aktuální a cílový stav, protože jsou oba reprezentovány pomocí vektorů stejných rozměrů. Často zde dochází ke konfliktu odpalitelnosti (Petriho síť s volným výběrem). Prohledávání stavového prostoru se musí řídit nějakými omezujícími podmínkami. Často bývá takovou podmínkou, že se stav světa nesmí opakovat, proto je potřeba zaznamenávat historii stavu, která nedovolí inferenčnímu mechanismu pomocí jinak platných operací uvést systém do již jednou dosaženého stavu.

Při realizaci báze znalostí pro diagnostický expertní systém je zásadní řešit problém odpalitelnosti. Petriho síť s volným výběrem zde není vhodná. Je nutné pro každý stav systému definovat následující operace. Proto při tvorbě báze znalostí pro expertní systémy tohoto typu lze využít Petriho sítě vyšší úrovně. V předcházející kapitole to byly především synchronizované Petriho sítě a Petriho sítě s nastavenou prioritou přechodů, nicméně lze využít i stochastické, barvené nebo časované Petriho sítě. Tímto řešením je dosaženo zvýraznění rozhodovacích schopností Petriho sítě potlačení modelovacích schopností. V úlohách diagnostického charakteru je potřeba především schopnost rozhodnout jak pokračovat z určitého stavu dále, na základě faktů (stav Petriho sítě) a ne modelování celého spektra stavů. Inferenční

mechanismus potom pracuje s danou bází znalostí tak, že zjistí odpalitelnost všech přechodů a na základě vnitřních podmínek přechodů (priorita, synchronizace, barva) rozhodne který přechod bude odpálen.

Další variantou je doplnit přechody, u kterých dochází ke konfliktu odpalitelnosti pomocnými místy, která jsou svázána se zodpovězením dotazu. Na základě odpovědi na tento dotaz bude rozhodnuto o umístění tokenu do pomocného místa a tím pádem bude nebo nebude splněna podmínka odpalitelnosti daného přechodu.

7. SEZNAM POUŽITÉ LITERATURY

[1] Dvořák J.: Expertní systémy, Brno: Vysoké učení technické, Fakulta strojního inženýrství, Ústav automatizace a informatiky, Skriptum VUT, 2004

[2] Provazník I., Kozumplík J.: Expertní systémy, Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, Ústav biomedicínckého inženýrství, Skriptum VUT, 1999

[3] Provazník I. , Bartoňová J.: Expertní systémy – praktická cvičení, 1. vydání Vysoké učení technické v Brně, Fakulta elektrotechniky a informatiky, Ústav biomedicínckého inženýrství, Skriptum VUT 2000, 49s ISBN 80-214-1768-4

[4] David R., Alla H. : Discrete, Continuous and Hybrid Petri Nets, Berlin, Springer – Verlag Berlin Heidelberg, 2005 524s ISBN 3-540-22480-7

[5] Češka M. : Petriho sítě, akademické nakladatelství CERM Brno, Říjen 1994, 94s., ISBN 80-85867-35-4

[6] Češka M. a kol. : Petriho sítě, Brno, Vysoké učení technické v Brně, Fakulta informačních technologií, Skriptum VUT, 2009, 244s

[7] Kochaníčková M. : Petriho sítě, Univerzita Palackého, Přírodovědecká fakulta, Katedra informatiky, Olomouc 2008, 69s

[8] Mařík V. Štěpánková O., Katanský J. a kol : Umělá inteligence 1 1.vydání vyd. Academia, nakladatelství Akademie věd České republiky, Praha 1993, 264 s. ISBN 80-200-0496-3

[9] Mařík V. Štěpánková O., Lažanský J. a kol : Umělá inteligence 2 1.vydání vyd. Academia, nakladatelství Akademie věd České republiky, Praha 1997, 373 s. ISBN 80-200-0504-8

[10] Faltus V.: Aplikace Petriho sítí při modelování dynamického řízení křižovatek, Automatizace: *magazín systémů řízení a informačních technologií* únor 2005, roč. 48, č.2 ,s. 88 - 91

[11] Michael D. Fischer : Expert systems and anthropological anylysis, University of Canterbury, 2007, Dostupné z URL http://lucy.ukc.uk/bicaweb/b4_/expert.html [citováno 1.12.2008]

[12] Expert systems in production planning and scheduling: A state-of-the-art survey Dostupné z URL http://www.springerlink.com/content/237kgq30ggtramg3 [citováno 12.03.2010]

[13] Zbořil F., Hanáček P. : Umělá inteligence, Fakulta elektrotechniky, nakladatelství Vysokého učení technického v Brně.26.7.1990, 125s., ISBN 80-214-0209-1

[14] Jirsík V. : Expertní systém pro výběr automobilů, Automatizace: *magazín systémů řízení a informačních technologií,*květen 2006, roč. 49, č.5, str. 319 - 321

[15] Planning Expert systems, Halton Borough Council Dostupné z URL <http://www2.halton.gov.uk/content/environment/planning/developmentcontrol/pes/?a=5441> [citováno 30.04.2010]

Seznam příloh

CD – obsahující elektronickou verzi DP + implementace diagnostického a plánovacího expertního systému v prostředí Microsoft Visual C#, .Net Framework