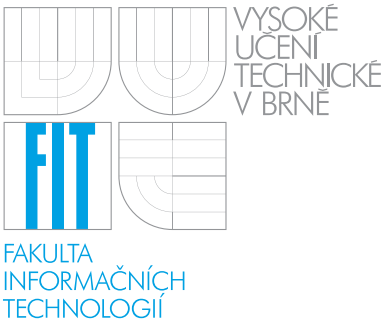


Petriho sítě
PES
Studijní opora

Milan Češka, Vladimír Marek, Petr Novosad, Tomáš Vojnar
Ústav inteligentních systémů
Fakulta informačních technologií
VUT v Brně

Prosinec ’09
Verze 1.6

Obsah

Kapitola 1

Úvod

1.1 Obecná charakteristika Petriho sítí

Petriho sítěmi je označována široká třída diskrétních matematických modelů (strojů), které umožňují popisovat specifickými prostředky řídící toky a informační závislosti uvnitř modelovaných systémů. Jejich historie je datována od r. 1962, kdy německý matematik C. A. Petri zavedl ve své doktorské disertační práci „Kommunikation mit Automaten“ nové koncepty popisu vzájemné závislosti mezi podmínkami a událostmi modelovaného systému. Jak napovídá název práce, tyto koncepty vyšly z dekompozice systému na podsystémy popisované konečnými automaty, jež pracují autonomně, avšak jejich činnost může být v potřebné míře vzájemně koordinována.

V současné době jsou Petriho sítě nejčastěji spojovány s aplikacemi při návrhu, analýze a modelování paralelních a distribuovaných systémů a to jak v oblasti technického, tak i programového vybavení počítačů. Typické použití Petriho sítí nalezneme v takových oblastech jako je návrh a popis paralelních architektur, popis komunikačních protokolů, paralelních databázových systémů, překladačů a počítačových sítí. Použití Petriho sítí se však neomezuje jen na počítačové systémy; jejich aplikace, např. v telekomunikacích, při popisu automatizovaných průmyslových systémů nebo systémů v administrativě, jsou stejně úspěšné a perspektivní.

Studium Petriho sítí zahrnuje celou řadu hledisek. Čtyřicetiletá historie těchto matematických modelů vyústila v bohatou a neustále se rozvíjející teorii, jež dává výsledky využitelné při analýze modelovaných systémů a v postupech blížících se důkazům určitých požadovaných vlastností systémů. Metody analýzy stavového prostoru Petriho sítí, strukturálních vlastností sítí a výpočetních posloupností generovaných těmito sítěmi tvoří v současné době hlavní oblast výzkumu, jehož cílem je velmi obtížný a důležitý problém verifikace paralelních

a distribuovaných aplikací. Základní výzkum i aplikace Petriho sítí nalezneme také v několika projektech vrcholového výzkumného programu ESPRIT (European Stategic Programme for Research and Development in Information Technology).

Velmi významné je rovněž studium Petriho sítí z hlediska jejich popisných a modelovacích možností. Aplikace metody postupného snižování abstrakce (metody návrhu shora dolů), ale i metody návrhu zdola nahoru, popis asynchronních událostí nebo simultánních (paralelních) činností představuje typické možnosti této třídy matematických modelů. Petriho sítí se rovněž používá pro popis sémantiky moderních programovacích jazyků (Smalltalk, Occam).

1.2 Obsahové a metodické informace o předmětu Teoretická informatika

1.2.1 Cíle předmětu

Pochopení základních konceptů a metod modelování systémů prostřednictvím Petriho sítí. Zvládnutí teorie Petriho sítí a její aplikace pro modelování, navrhování a verifikaci počítačových systémů. Praktické zvládnutí využívání počítačových nástrojů pro typické aplikace Petriho sítí.

1.2.2 Anotace předmětu

Základní koncepty a prostředky popisu diskrétních systémů Petriho sítěmi, klasifikace Petriho sítí, teorie C/E Petriho sítí, metody analýzy C/E Petriho sítí, teorie P/T Petriho sítí, metody analýzy P/T Petriho sítí, jazyky Petriho sítí a problém automatizované syntézy modelů, podtřídy a rozšíření Petriho sítí, barvené Petriho sítě, hierarchické a objektově orientované Petriho sítě, nástroje pro práci s Petriho sítěmi, aplikace.

1.2.3 Požadované prerekvizitní znalosti a dovednosti

Základní znalosti z binárních relací, teorie grafů a formálních jazyků včetně konečných a zásobníkových automatů, Turingových strojů, pojmů algoritmické složitosti a a principů počítačového modelování.

1.2.4 Osnova přednášek a přiřazení ke kapitolám opory

1. [Kapitola 3] Úvod, filozofie, historie a aplikace Petriho sítí, pojem sítě a odvozených základních pojmů.

2. [Kapitola 4] C/E Petriho sítě, případy a kroky, stavový prostor C/E systémů, cyklické a živé C/E systémy, ekvivalence C/E systémů.

3. [Kapitola 4] Bezkontaktní C/E systémy, komplementace, případové grafy a jejich aplikace pro analýzu C/E systémů.

4. [Kapitola 5] Procesy C/E systémů, relace podobnosti a její oblasti, výskytové sítě, vlastnosti procesů a jejich kompozice.

5. [Kapitola 6] Vlastnosti C/E systémů, pojem synchronizační vzdálenosti, speciální synchronizační vzdálenosti, reprezentace vlastností výrokovou logikou, fakta.

6. [Kapitola 7] P/T Petriho sítě, definice, evoluční pravidla, stavový prostor, základní problémy analýzy (bezpečnost, omezenost, konzervativnost, živost).

7. [Kapitola 8] Reprezentace nekonečného stavového prostoru, strom dosažitelných značení, výpočet a využití stromu dosažitelných značení pro analýzu P/T sítí.

8. [Kapitola 9] Pojem invariantu, P a T invarianty, definice, výpočet a využití invariantu pro analýzu P/T sítí.

9. [Kapitola 10] Pojem jazyka Petriho sítě, typy jazyků, uzávěrové vlastnosti jazyků Petriho sítí, vztah těchto jazyků k Chomského hierarchii (modelovací schopnost).

10. [Kapitola 11] Podtřídy a rozšíření P/T sítí, stavové stroje, značené grafy, P/T sítě s volným výběrem, sítě s inhibitory, časované a stochastické Petriho sítě.

11. [Kapitola 12] Barvené Petriho sítě, základní vyjadřovací prostředky, inskripční jazyk, počítačový nástroj pro práci s těmito sítěmi (CPN Design).

12. [Kapitola 13] Hierarchické a objektově orientované Petriho sítě, prostředky hierarchického návrhu, substituce a invokace, začlenění prostředků objektově orientovaného návrhu, PNtalk jako jazyk pro práci s OO Petriho sítěmi.

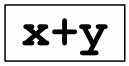
1.2.5 Způsob hodnocení

Písemná půlsemestrální a semestrální zkouška za 30 a 50 bodů. Vypracování pěti hodnocených úkolů po 4 bodech.

1.2.6 Tabulka používaných piktogramů

Cíl


Čas potřebný pro studium



Příklad

Definice


Otázka, příklad k řešení


Souhrn


Důležité

Kapitola 2

Definice základních matematických pojmů

Čas potřebný ke studiu: 3 hodiny

Cíle kapitoly

Ústředním tématem této publikace jsou Petriho sítě, které jsou definovány jako algebraická struktura (n-tice). Obdobně i jejich vlastnosti jsou definovány formálně za pomocí množin, relací, zobrazení, atd. Definice a vysvětlení základních pojmů z oblasti diskrétní matematiky, které jsou nezbytné pro porozumění ostatním kapitolám, jsou shrnuty v této kapitole. Její nedílnou součástí je také notace, pomocí níž se tyto algebraické struktury zapisují.

Průvodce studiem

Kapitola je určena jednak čtenářům, kteří si chtějí osvěžit své znalosti základních algebraických struktur a jednak čtenářům, kteří používají pro jejich zápis odlišnou notaci. Čtenáři, kteří mají solidní základy algebry a rozumějí použité notaci, mohou tuto kapitolu přeskočit.

Obsah

Ve výkladu začneme *množinami*, které jsou později rozšířeny do pojmu *multimnožina*. Od množin se přes *relace* a *zobrazení* postupuje až k definici *vektorů* a *matic*.

2.1 Množina

množina

Ačkoliv je pojem *množina* (angl. *set*) zcela základním aspektem, na jehož základě jsou postaveny takřka všechny definice této publikaci, neuvádíme zde její precizní definici. Postačí nám takzvaná naivní definice množiny, která množinu definuje jako strukturu, která splňuje následující vlastnosti:

- Množina je plně definována prvky, které obsahuje. Dvě množiny, které obsahují tytéž prvky, jsou identické. Při explicitním vyjádření množiny pomocí jejích prvků nezáleží na jejich pořadí.
- Každý prvek množiny je v ní obsažen právě jednou, množina tedy neobsahuje prvky duplicitně.
- Množina nemusí obsahovat žádný prvek, v takovém případě hovoříme o *prázdné množině*.

Pokud je množina definována výčtem prvků, pak tyto prvky uvádíme uzavřené do složených závorek. Např. $\{a, b, c\}$, $\{x\}$, $\{0, 1, 2, 3\}$ jsou množiny. Speciální množinou, neobsahující žádný prvek, je *prázdná množina* , kterou označujeme symbolem $\emptyset$.

prázdná množina

Množiny obvykle označujeme velkými písmeny, jejich prvky obvykle malými písmeny, např. $A = \{a, b, c\}$, $P = \{p_1, p_2, p_3\}$. Pokud potřebujeme vyjádřit, že daný prvek je prvkem množiny, používáme k tomu symbol $\in$, např. $p_2 \in \{p_1, p_2, p_3\}$. Pokud potřebujeme vyjádřit opačné tvrzení, tedy že daný prvek není prvkem množiny, používáme k tomu symbol $\notin$. Např. $a \notin \{p_1, p_2, p_3\}$.

Symboly $\mathbb{N}$ a $\mathbb{Z}$ označujeme množiny se speciálním významem:

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ je *množina přirozených čísel*
- $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$ je *množina celých čísel*.

Výraz $|A|$ nad množinou A popisuje *kardinalitu* množiny (počet prvků).

kardinalita

DEF

■ **Definice 2.1** *Množinové sjednocení, průnik a rozdíl.*

Nechť A a B jsou dvě množiny. Definujme operace nad množinami: *sjednocení* (angl. *set union*), *průnik* (angl. *set intersection*) a *rozdíl* (angl. *set difference*).

1. *Sjednocení množin:*

$$A \cup B \stackrel{\text{def}}{=} \{x | x \in A \vee x \in B\}$$

Jinými slovy, x je prvkem sjednocení množin A a B, když x je v množině A nebo v množině B. Např. $\{p, q, r\} \cup \{p, q, s\} = \{p, q, r, s\}$.

2. *Průnik množin:*

$$A \cap B \stackrel{\text{def}}{=} \{x | x \in A \wedge x \in B\}$$

Jinými slovy, x je prvkem průniku množin A a B, když x je v množině A a zároveň v množině B. Např. $\{p, q, r\} \cap \{p, q, s\} = \{p, q\}$.

3. *Množinový rozdíl:*

$$A \setminus B \stackrel{\text{def}}{=} \{x | x \in A \wedge x \notin B\}$$

Jinými slovy, x je prvkem rozdílu množin A a B, když x je v množině A a zároveň není v množině B. Např. $\{p, q, r\} \setminus \{p, q, s\} = \{r\}$.

□

DEF

■ **Definice 2.2** *Podmnožina.*

Nechť A a B jsou dvě množiny. Množinu A označujeme jako *podmnožinu* (angl. *subset*) množiny B, když každý prvek množiny A je rovněž prvkem množiny B. Formálně:

$$A \subseteq B \stackrel{\text{def}}{\Leftrightarrow} (\forall x \in A : x \in B)$$

Pokud A není podmnožinou B, tedy $\neg(A \subseteq B)$, pak zapisujeme $A \nsubseteq B$. □

DEF

■ **Definice 2.3** *Vlastní podmnožina.*

Nechť A a B jsou dvě množiny. Množinu A označujeme jako *vlastní podmnožinu* (angl. *proper subset*) množiny B, pokud je A podmnožinou B a zároveň jsou tyto množiny rozdílné. Formálně:

$$A \subset B \stackrel{\text{def}}{\Leftrightarrow} (A \subseteq B) \wedge (A \neq B)$$

Pokud A není vlastní podmnožinou B, tedy $\neg(A \subset B)$, pak zapisujeme $A \not\subset B$. □

x+y

■ **Příklad 2.1** *Podmnožina, vlastní podmnožina.*

Platí:

$$\begin{array}{rcl} \{p,s\} & \subseteq & \{p,q,s\} \\ \emptyset & \subseteq & \{p,q,s\} \\ \{p,q,s\} & \subseteq & \{p,q,s\} \\ \emptyset & \subseteq & \emptyset \\ \{p,r\} & \not\subseteq & \{p,q,s\} \end{array} \qquad \begin{array}{rcl} \{p,s\} & \subset & \{p,q,s\} \\ \emptyset & \subset & \{p,q,s\} \\ \{p,q,s\} & \not\subset & \{p,q,s\} \\ \emptyset & \not\subset & \emptyset \\ \{p,r\} & \not\subset & \{p,q,s\} \end{array}$$

□

DEF

■ **Definice 2.4** *Potenční množina.*

Nechť A je množina. Pak symbolem $\mathbb{P}(A)$ značíme *potenční množinu* (angl. *powerset*), tedy množinu všech podmnožin množiny A.

$$\mathbb{P}(A) \stackrel{\text{def}}{=} \{x | x \subseteq A\}$$

□

Pro zápis potenční množiny nad množinou A se často používá odlišná notace, a to 2^A. Tento zápis vychází z faktu, že počet prvků potenční množiny nad konečnou množinou A je roven $2^{|A|}$. Tedy

$$|\mathbb{P}(A)| = |2^A| = 2^{|A|}$$

x+y

■ **Příklad 2.2** *Potenční množina.*

Platí:

$$\begin{array}{rcl} \mathbb{P}(\emptyset) & = & \{\emptyset\} \\ \mathbb{P}(\{a\}) & = & \{\emptyset, \{a\}\} \\ \mathbb{P}(\{a,b,c\}) & = & \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c,\}\} \end{array}$$

□

DEF

■ **Definice 2.5** *Kartézský součin.*

Nechť A a B jsou dvě množiny. *Kartézský součin* (angl. *Cartesian product*) těchto dvou množin označujeme $A \times B$ a je definován jako množina všech uspořádaných dvojic takových, že první složka dvojice je prvkem množiny A a druhá složka dvojice je prvkem množiny B. Formálně:

$$A \times B \stackrel{\text{def}}{=} \{\langle x, y \rangle | x \in A \wedge y \in B\}$$

□

x+y

■ **Příklad 2.3** *Kartézský součin.*

Mějme dvě množiny $A = \{a, b\}$ a $B = \{p, q, r\}$. Platí:

$$\begin{array}{lcll} A & \times & B & = \{\langle a,p\rangle, \langle a,q\rangle, \langle a,r\rangle, \langle b,p\rangle, \langle b,q\rangle, \langle b,r\rangle\} \\ B & \times & A & = \{\langle p,a\rangle, \langle p,b\rangle, \langle q,a\rangle, \langle q,b\rangle, \langle r,a\rangle, \langle r,b\rangle\} \\ A & \times & \emptyset & = \emptyset \\ \emptyset & \times & A & = \emptyset \\ \emptyset & \times & \emptyset & = \emptyset \end{array}$$

□

2.2 Multimnožina

multimnožina

Multimnožina (angl. *bag*) je zobecněním pojmu *množina*. Naopak, množinu můžeme chápat jako speciální případ multimnožiny. Multimnožina se od množiny liší tím, že připouští vícenásobný výskyt téhož prvku. Pro zápis multimnožiny se používá notace shodná s notací pro zápis množiny – multimnožinu lze tedy zachytit výčtem prvků uzavřeným ve složených závorkách.

Význam symbolů $\in$, $\subset$ a $\subseteq$ je v případě multimnožin obdobný jako v případě množin:

- $a \in A$ značí, že v multimnožině A je obsažen prvek a *alespoň jedenkrát*
- $A \subseteq B$ značí, že všechny prvky multimnožiny A jsou také obsaženy v multimnožině B a to ve stejném, nebo větším počtu výskytů
- $A \subset B$ značí totéž, co v případě množin, tedy $A \subseteq B \wedge A \neq B$.

Typickým příkladem multimnožiny je rozklad přirozeného čísla na prvočinitele, $\underline{180} = \{2, 2, 3, 3, 5\}$ je multimnožina reprezentující číslo 180.

Pro používání multimnožiny zavedeme tyto konvence:

- Nechť A je multimnožina a a její prvek. Symbolem $\#(a, A)$ budeme značit počet výskytů prvku a v A. Např. $\#(3, \underline{180}) = 2$. Zřejmě platí následující implikace: $\#(a, A) = 0 \Rightarrow a \notin A$
- Je-li A množina, pak symbolem A^∞ značíme systém multimnožin vytvořených z prvků množiny A.

2.3 Binární relace

Binární relace je definována jako podmnožina kartézského součinu dvou množin.

DEF

■ **Definice 2.6** *Binární relace.*

Nechť A a B jsou dvě množiny. Podmnožinu kartézského součinu $R \subseteq A \times B$ nazýváme *binární relace* z množiny A do množiny B. □

x+y

■ **Příklad 2.4** *Binární relace.*

Mějme dvě množiny $A = \{a, b\}$ a $B = \{p, q, r\}$. Následující množiny jsou binárními relacemi z A do B:

$$\begin{aligned} \emptyset &\subseteq A \times B \\ \{\langle a,p\rangle, \langle b,p\rangle\} &\subseteq A \times B \\ \{\langle b,p\rangle, \langle b,q\rangle\} &\subseteq A \times B \\ \{\langle a,r\rangle, \langle b,q\rangle, \langle b,r\rangle\} &\subseteq A \times B \end{aligned}$$

□

Notace *Zápis binární relace.*

Při vyjádření faktu, že uspořádaná dvojice $\langle a, b\rangle$ patří do dané relace R, lze vycházet z toho, že relace je množina, tedy lze zapsat $\langle a, b\rangle \in R$. Lze však použít i stručnější notaci pro vyjádření téhož, aRb. Tento infixově zapsaný predikát se používá zejména u relací, které jsou pojmenovány neabecedním symbolem, např. $\approx, \preceq$, apod., případně tam, kde to zvýší stručnost a přehlednost zápisu. □

Speciálním případem binární relace je relace z množiny do téže množiny, tedy $R \subseteq A \times A$. U takovýchto relací můžeme zkoumat další vlastnosti, viz dále.

DEF

■ **Definice 2.7** *Reflexivita, symetrie a tranzitivita.*

Nechť $R \subseteq A \times A$ je binární relace na množině A. Relace R je:

- *reflexivní*, $\overset{\text{def}}{\Leftrightarrow} \forall a \in A : aRa$
- *ireflexivní*, $\overset{\text{def}}{\Leftrightarrow} \forall a \in A : \neg(aRa)$
- *symetrická*, $\overset{\text{def}}{\Leftrightarrow} \forall a, b \in A : aRb \Rightarrow bRa$
- *antisymetrická*, $\overset{\text{def}}{\Leftrightarrow} \forall a, b \in A : aRb \wedge bRa \Rightarrow a = b$

- *asymetrická*, $\overset{\text{def}}{\Leftrightarrow} \forall a, b \in A : aRb \Rightarrow \neg(bRa)$
- *tranzitivní*, $\overset{\text{def}}{\Leftrightarrow} \forall a, b, c \in A : aRb \wedge bRc \Rightarrow aRc$

□

DEF

■ **Definice 2.8** *Ekvivalenční relace, ekvivalence.*

Nechť $R \subseteq A \times A$ je binární relace na množině A. Relace R je *ekvivalence*, když je:

- reflexivní
- symetrická
- tranzitivní

Relace ekvivalence je obvykle značena symbolem $\approx$, resp. $\equiv$. □

x+y

■ **Příklad 2.5** *Ekvivalence.*

Nechť $A = \{a, b, c, d\}$. Relace $R = \{\langle a, a\rangle, \langle b, b\rangle, \langle b, d\rangle, \langle c, c\rangle, \langle d, b\rangle, \langle d, d\rangle\}$ je ekvivalenční relace na množině A. □

DEF

■ **Definice 2.9** *Částečné uspořádání $\preceq$.*

Nechť $R \subseteq A \times A$ je binární relace na množině A. Relace R je *částečné uspořádání* $\preceq$, když je:

- reflexivní
- antisymetrická
- tranzitivní

Relace částečného uspořádání definovaná jako reflexivní, antisymetrická a tranzitivní je obvykle značena symbolem $\preceq$. □

DEF

■ **Definice 2.10** *Uspořádání $\prec$.*

Nechť $R \subseteq A \times A$ je binární relace na množině A. Relace R je *uspořádání* $\prec$, když je:

- asymetrická
- tranzitivní

Relace uspořádání definovaná jako asymetrická a tranzitivní je obvykle značena symbolem $\prec$. □

■ **Definice 2.11** *Relace podobnosti.* **DEF**

Nechť $R \subseteq A \times A$ je binární relace na množině A. Relace R je *relace podobnosti*, když je:

- reflexivní
- symetrická

□

2.4 Zobrazení

Zobrazení z množiny A do B je definováno jako binární relace z množiny A do B, kde každý prvek množiny A je v relaci s právě jedním prvkem množiny B.

■ **Definice 2.12** *Zobrazení.* **DEF**

Nechť $R \subseteq A \times B$ je binární relace z A do B. Tuto relaci nazveme *zobrazení* (angl. *map, mapping*), když

$$\forall a \in A : \exists b \in B : \{x | \langle a, x \rangle \in R\} = \{b\}$$

Pokud je relace R zobrazení z A do B, pak tuto skutečnost zapisujeme výrazem

$$R : A \to B$$

□

Notace *Zápis zobrazení.*

Jelikož je zobrazení relací, lze jej jako relaci zapsat. Jako příklad uveďme zobrazení $R : \{a, b, c, d\} \to \{1, 2, 3\}$ definované jako

$$R = \{\langle a, 1 \rangle, \langle b, 3 \rangle, \langle c, 2 \rangle, \langle d, 1 \rangle\}$$

Pro výstižnost však často v případě zobrazení používáme jinou notaci zápisu uspořádaných dvojic, a to pomocí symbolu $\mapsto$. Stejná relace v této notaci je definována

$$R = \{a \mapsto 1, b \mapsto 3, c \mapsto 2, d \mapsto 1\}$$

Běžná je rovněž notace, která zobrazení definuje zápisem, jaký používáme u funkcí:

$$\begin{aligned} R(a) &= 1 \\ R(b) &= 3 \\ R(c) &= 2 \\ R(d) &= 1 \end{aligned}$$

□

2.5 Vektory a matice

Pro potřeby teorie Petriho sítí si vystačíme s vektory a maticemi přirozených čísel, tedy nad množinou $\mathbb{N}$, resp. celočíselnými, tedy nad množinou $\mathbb{Z}$.

V jiných matematických disciplínách se vektor a matice definuje jako jedno- či dvourozměrné pole prvků, které jsou v jednotlivých rozměrech indexovány pomocí přirozených čísel. Význam vektoru, resp. matice obvykle spočívá v přiřazení číselných hodnot prvkům množiny, resp. prvkům kartézského součinu dvou množin. Indexace prvků pomocí přirozených čísel však v takovém případě vede na nutnost stanovení pořadí (tedy indexů) v rámci množiny, jejímž prvkům pomocí vektoru či matice stanovujeme číselnou hodnotu.

Jako příklad lze uvést značení Petriho sítě, jejíž množina míst je $P = \{p_1, p_2, p_3\}$. Vyjádření značení M, v němž místo p_1 obsahuje 2 tečky a místa p_2 a p_3 neobsahují žádnou tečku, lze např. pomocí vektoru $(2, 0, 0)$. Pokud však explicitně není řečeno, že jednotlivé složky vektoru odpovídají postupně prvkům p_1, p_2 a p_3, pak je toto vyjádření nepostačující – pokud by někdo předpokládal jiné pořadí míst Petriho sítě odpovídající jednotlivým složkám (např. $\{p_2, p_3, p_1\}$), pak by dospěl k jinému znační sítě.

Z tohoto důvodu používáme indexování prvků vektoru, resp. matice přímo prvky množiny (množin) - nejednoznačnost přiřazení číselných hodnot jednotlivým prvkům je odstraněna.

DEF

▪ **Definice 2.13** *Vektor.*

Nechť A je neprázdná konečná množina. Zobrazení $v : A \to \mathbb{Z}$ se nazývá *vektor*, případně též *A-vektor*. □

DEF

▪ **Definice 2.14** *Operace nad vektory.*

Nechť $v, v_1, v_2 : A \to \mathbb{Z}$ jsou vektory a $x \in \mathbb{Z}$ celé číslo

- *Součet*:
$$v = v_1 + v_2 \overset{\text{def}}{\Leftrightarrow} \forall a \in A : v(a) = v_1(a) + v_2(a)$$
- *Součin:*
$$x = v_1.v_2 \overset{\text{def}}{\Leftrightarrow} x = \sum_{a \in A} v_1(a).v_2(a)$$
- *Celočíselný násobek:*
$$v = z.v1 \overset{\text{def}}{\Leftrightarrow} \forall a \in A : v(a) = z.v_1(a)$$

□

■ **Definice 2.15** *Nulový, nezáporný a kladný vektor.* **DEF**

- *Nulový vektor:* $v : A \to \{0\}$ nazýváme *nulový vektor* a značíme symbolem $\mathbf{0}$.
- *Nezáporný vektor:* $v : A \to \mathbb{Z}$ se nazývá *nezáporný*, jestliže $\forall a \in A : v(a) \geq 0$.
- *Kladný vektor:* $v : A \to \mathbb{Z}$ se nazývá *kladný*, jestliže $\forall a \in A : v(a) > 0$.

□

■ **Definice 2.16** *Charakteristický vektor množiny.* **DEF**

Nechť A a B jsou množiny, $B \subseteq A$. Vektor $c_B : A \to \{0, 1\}$ se nazývá *charakteristický* vektor množiny B, když

$$\forall a \in A : c_B(a) = \begin{cases} 1, & \text{je-li } a \in B \\ 0, & \text{je-li } a \notin B \end{cases}$$

□

■ **Definice 2.17** *Matice.* **DEF**

Nechť A a B jsou neprázdné, konečné a disjunktní množiny.

1. Zobrazení $C : A \times B \to \mathbb{Z}$ se nazývá *matice*.
2. *Transponovanou maticí* C^T k matici $C : A \times B \to \mathbb{Z}$ nazýváme matici $C^\mathsf{T} : B \times A \to \mathbb{Z}$, pro kterou platí:
$$\forall a \in A, \forall b \in B : C^\mathsf{T}(b, a) = C(a, b)$$
3. *Součinem* matice $C : A \times B \to \mathbb{Z}$ a vektoru $v : B \to \mathbb{Z}$ nazýváme vektor $C.v : A \to \mathbb{Z}$, kde
$$\forall a \in A : C.v(a) = \sum_{b \in B} C(a, b).v(b)$$

□

Vektory a matice obvykle zobrazujeme jako tabulky, kde řádky, resp. sloupce jsou označeny prvky indexových množin A, resp. B.

Pojmy k zapamatování

- množina, podmnožina, multimnožina
- kartézský součin, relace, zobrazení
- vektor, matice

Shrnutí

Tato kapitola si kladla za cíl uvést čtenáře do základů algebraických struktur, které je nutné znát pro pochopení následujících kapitol.

Příklady k procvičení

1. Definujte pojem množina a vyjmenujte možné operace nad množinami.
2. Uveďte příklad binární relace.
3. Definujte pojem ekvivalence a částečné uspořádání.
4. Definujte pojem vektor a vyjmenujte možné operace nad vektory.
5. Definujte pojem matice a vyjmenujte možné operace nad maticemi.

Kapitola 3

Základní koncepty Petriho sítí

Čas potřebný ke studiu: 3 hodiny

Cíle kapitoly

Cílem této kapitoly je zavedení základních definice Petriho sítí. Vymezení odlišností a výhod, které přinášejí Petriho sítě v popisu systému ve srovnání s konečnými automaty. Kapitola stanovuje rámce teorie, které jsou využívány i v následujících částech učebního textu.

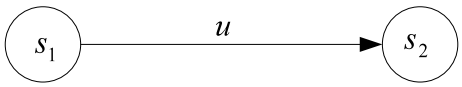
Průvodce studiem

Na tuto kapitolu navazují prakticky všechny ostatní kapitoly, proto je potřeba důkladnému pochopení jednotlivých termínů věnovat náležitou pozornost.

Obsah

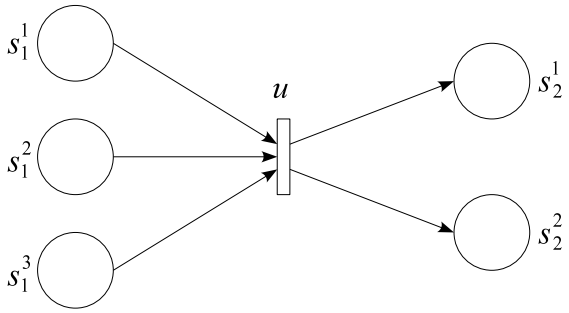
3.1 Základní koncepty

Petriho sítě vznikly rozšířením modelovacích možností konečných automatů. Základním prostředkem popisu změny v modelovaném systému při použití konečného automatu je pojem stav a přechod mezi stavy. Na obrázku 3.1 je např. modelována změna v určitém systému, která nastane při události u (podmíněné stavem s_1) a která přivede systém do stavu s_2.



Obrázek 3.1: Modelování změny stavu systému konečným automatem

Předpokládejme nyní, že je účelné popsat stav s_1 určitými podmínkami či parciálními stavy např. s_1^1, s_1^2, s_1^3, a stav s_2 analogicky např. parciálními stavy s_2^1 a s_2^2. Pak situaci na obrázku 3.1 lze prostředky Petriho sítě vyjádřit tak, jak je uvedeno na obrázku 3.2.



Obrázek 3.2: Modelování změny stavu systému Petriho sítí

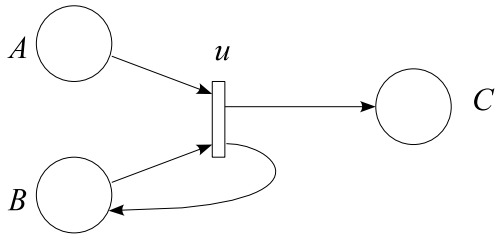
Znázorněná distribuce „globálního“ stavu systému na parciální stavy, které jsou vyjádřitelné velmi často nějakými podmínkami, a spojení těchto podmínek s událostmi systému je koncepčním jádrem popisu diskrétního systému Petriho sítí. Parciální stavy, graficky zobrazované jako malé kružnice, se nazývají *místa Petriho sítě*; události, graficky zobrazované jako obdélníčky nebo úsečky, se nazývají *přechody Petriho sítě*. Na obrázku 3.2 je uvedeno propojení přechodu u s místy, které popisují jedno z pravidel, podle nichž se systém chová:

událost u může nastat (proběhnout) v případě, že jsou splněny podmínky odpovídající místům s_1^1, s_1^2, s_1^3, a způsobí, že systém dosáhne stavu, při kterém platí podmínky příslušející místům s_2^1 a s_2^2.

Petriho síť lze chápat jako automat, který definuje chování systému posloupnostmi událostí a odpovídajícími stavovými změnami systému. V případě konečných automatů byl stav systému jednoznačně určen prvkem množiny stavů. V případě Petriho sítí je okamžitý stav modelovaného systému konstituován určitými parciálními stavy spojenými s příslušnými místy sítě. Vzniká tedy otázka jakým způsobem je registrováno dosažení jistého parciálního stavu odpovídajícímu příslušnému místu sítě. K tomuto účelu slouží *značení* místa.

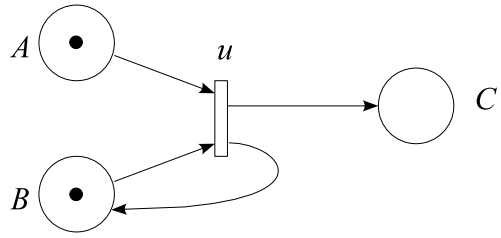
Značení místa je nezáporná celočíselná informace, jež je graficky vykreslena určitým počtem černých teček nazývaných *značkami*, které jsou umístěny v příslušném místě. V případě, že místo modeluje logickou podmínku, je tato informace binární; místo obsahuje značku v případě, že příslušná podmínka platí a značku neobsahuje, jestliže tato podmínka neplatí. Podmínkou k tomu, aby mohla proběhnout událost modelovaná určitým přechodem, je existence značky ve všech vstupních místech přechodu. Provedením přechodu (události) se odeberou značky vstupních míst (přestanou platit vstupní podmínky) a vzniknou značky ve výstupních místech (začnou platit podmínky způsobené touto událostí).

Na obrázku 3.3 je prostředky Petriho sítí popsána závislost události u a podmínek A, B, C. Událost u nastává, jestliže platí A i B; provedení události u má za následek platnost podmínky C a neplatnost A. Obrázek 3.4 znázorňuje značení míst, kdy může událost u nastat, obrázek 3.5 pak značení míst po proběhnutí události u.

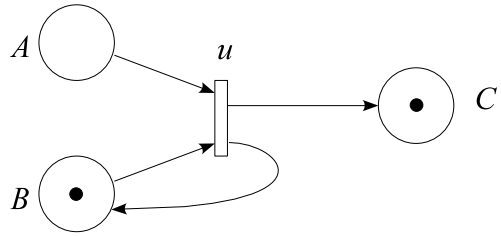


Obrázek 3.3: Závislost události a podmínek

Interpretace míst jako logických (dvouhodnotových) podmínek je charakteristická pro nejjednodušší třídu Petriho sítí, tzv. C/E Petriho sítě (Condition/Event Petri Nets). V zobecnění, které v současné době představuje nejrozšířenější třídu Petriho sítí, je místo interpretováno



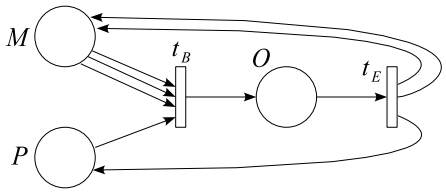
Obrázek 3.4: Značení míst, kdy může událost nastat



Obrázek 3.5: Značení míst, po proběhnutí události

jako parciální stav systému, který je specifikován celočíselným nezáporným značením. V tomto případě lze vstupní podmínku pro událost formulovat obecněji, jako minimální počet značek, které pro provedení této události musí vstupní místo obsahovat. Tohoto zobecnění se často výhodně využívá při modelování požadavků na omezené zdroje systému.

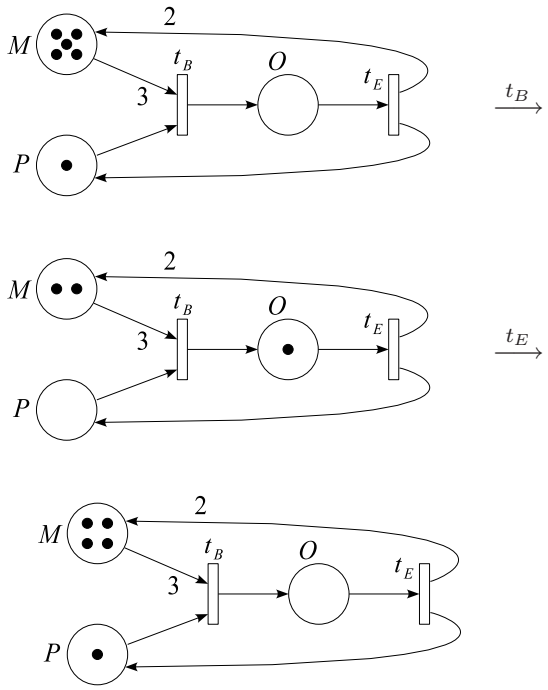
Na obrázku 3.6 je modelována jistá operace počítače, jejíž provedení vyžaduje kromě volného procesoru (místo P) také přidělení tří paměťových bloků hlavní paměti (místo M). Po ukončení operace jsou dva z těchto bloků vráceny k dalšímu použití.



M – počet volných paměťových bloků
P – procesor je volný (ano/ne)
O – operace probíhá (ano/ne)
t_B – začátek operace
t_E – konec operace

Obrázek 3.6: Modelování požadavků na zdroje

Pro zjednodušení grafu sítě se často místo násobné hrany používá celočíselné ohodnocení hrany, určující počet značek, jež se odebírají, či přidávají do příslušných míst. S touto konvencí jsou na obrázku 3.7 znázorněna možná značení před a po provedení přechodů t_B a t_E.



Obrázek 3.7: Provedení přechodů Petriho sítě

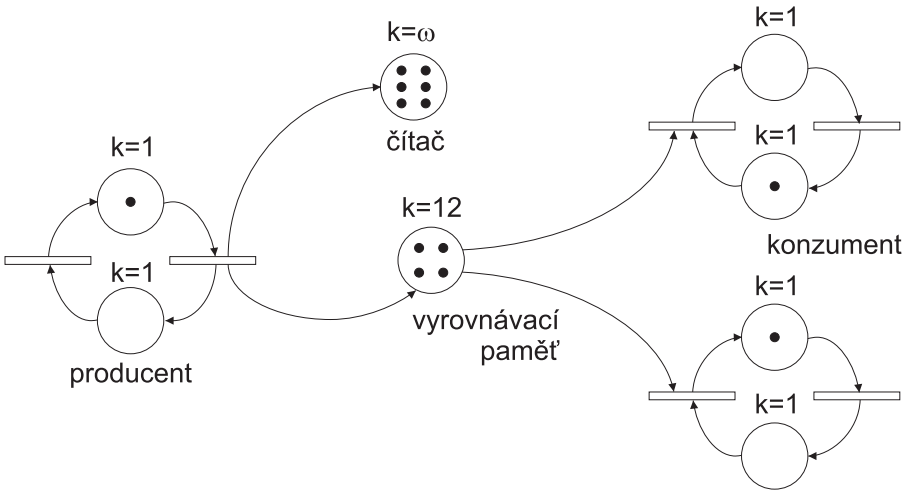
Značení místa může být *neomezené*, tj. může nabývat libovolné celočíselné hodnoty, nebo může být omezeno *kapacitou místa* – celým kladným číslem udávajícím maximální možný počet značek v daném místě. Kapacitní omezení se využívá při modelování situací, kdy chceme vyloučit přeplnění např. vyrovnávací paměti nebo délky fronty. V takovém případě je pravidlo pro provedení přechodu doplněno o podmínku na značení výstupních míst přechodu: přechod může být proveden, jestliže nebude překročena kapacita jeho některého výstupního místa.

x+y

▪ Příklad 3.1

Na obrázku 3.8 je uvedena Petriho síť modelu, který se nazývá „producent – konzument“, jehož implementaci můžeme často nalézt např. v operačním systému počítače. Producent produkuje data, která ukládá do vyrovnávací paměti. Odtud jsou data odebírána konzumentem k dal-

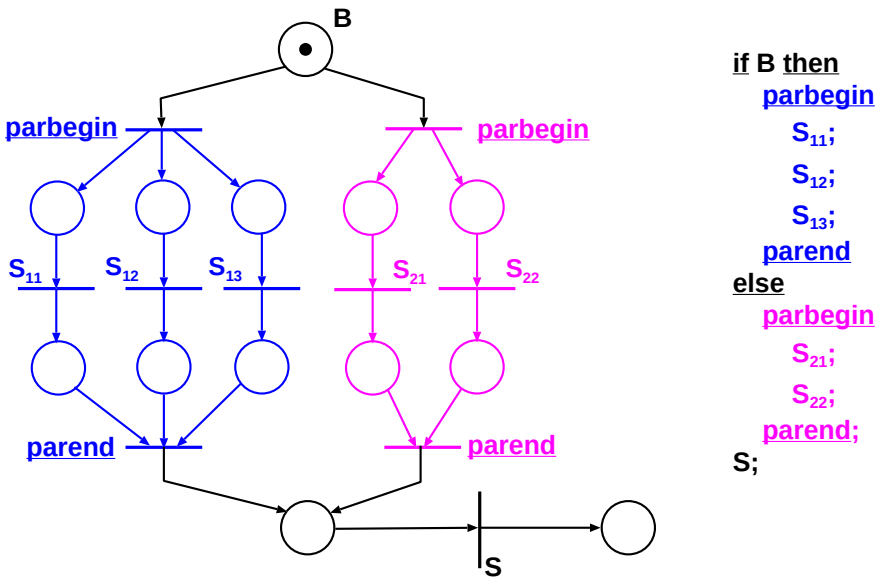
šímu případnému zpracování. Na obrázku 3.8 je model zahrnující jednoho producenta a dva konzumenty. Kapacita místa je značena písmenem κ; ω značí neomezenou kapacitu. Místo *čítač* s neomezenou kapacitou slouží k registraci celkového počtu produkovaných datových položek. □



Obrázek 3.8: Petriho síť modelu producent – konzument s omezenou vyrovnávací pamětí

Na závěr tohoto odstavce si položme otázku, jaké odlišnosti a výhody přinášejí Petriho sítě v popisu systému ve srovnání s konečnými automaty. Neomezenost značení míst vede zřejmě k nekonečnému stavovému prostoru obecné Petriho sítě, a tudíž modelovací schopnost Petriho sítí je obecně větší, než schopnost konečných automatů. Můžeme však pozorovat důležitou odlišnost i v případě Petriho sítí s konečným stavovým prostorem. Konečný automat definuje, stejně jako Petriho síť, posloupnosti událostí (tzv. *výpočetní posloupnosti*), které charakterizují chování systému. V případě konečného automatu jsou tyto posloupnosti výsledkem uspořádání jejich prvků podle závislostí definovaných přechodovou funkcí. Distribuce stavů a následně i událostí popisovaných Petriho sítí vede naopak k explicitnímu zobrazení relace vzájemné nezávislosti událostí. V tomto faktu spočívá hlavní důvod, proč se Petriho sítě využívají jako matematický model paralelních a distribuovaných výpočtů.

Na obrázku 3.9 je pro ilustraci uvedena Petriho síť části paralelního programu. Podobně obrázek 3.8 popisuje paralelismus procesů konzument a procesů konzument a producent.



Obrázek 3.9: Petriho síť úseku paralelního programu

3.2 Definice sítě

DEF

■ **Definice 3.1** *Síť.*

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

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

Množina P se nazývá *množinou míst* (Places) sítě N, množina T *množinou přechodů*(Transitions) sítě N a relace F *tokovou relací* (Flow relation) sítě N. □

Grafem sítě nazýváme bipartitní orientovaný graf, který vznikne grafovou reprezentací relace F. Množina $P \cup T$ je množinou vrcholů grafu sítě; místa jsou obvykle kreslena ve tvaru kružnic, přechody ve tvaru obdélníků nebo úseček.

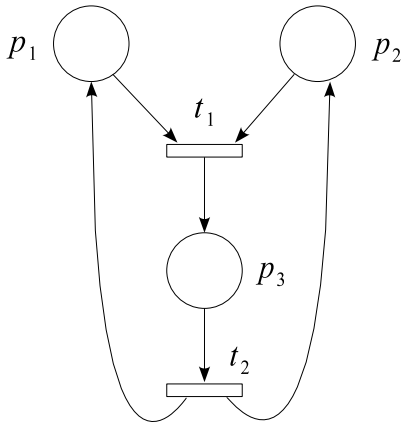
x+y

■ **Příklad 3.2**

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

$$\begin{aligned} P &= \{p_1, p_2, p_3\} \\ T &= \{t_1, t_2\} \\ F &= \{\langle p_1, t_1\rangle, \langle p_2, t_1\rangle, \langle p_3, t_2\rangle, \langle t_1, p_3\rangle, \langle t_2, p_2\rangle, \langle t_2, p_2\rangle\} \end{aligned}$$

je síť. Graf sítě N je vykreslen na obrázku 3.10. □



Obrázek 3.10: Graf sítě N

DEF

■ **Definice 3.2** *Vstupní a výstupní množina prvku a množiny prvků.*
Nechť $N = (P, T, F)$ je síť. Pro všechny prvky $x \in (P \cup T)$

$${}^\bullet x = \{y | yFx\}$$ se nazývá *vstupní množinou (preset)* prvku x.

$$x^\bullet = \{y | xFy\}$$ se nazývá *výstupní množinou (postset)* prvku x.

Pojmy *vstupní množina* a *výstupní množina* lze zobecnit také pro množinu prvků $X \subseteq (P \cup T)$:

$${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$$

$$X^\bullet = \bigcup_{x \in X} x^\bullet$$

Zřejmě pro každé $x, y \in (P \cup T)$ platí: $x \in {}^\bullet y \Leftrightarrow y \in x^\bullet$. □

DEF

■ **Definice 3.3** *Vlastní cyklus, čistá síť.*
Nechť $N = (P, T, F)$ je síť. Uspořádaná dvojice $\langle p, t \rangle \in P \times T$ se nazývá *vlastní cyklus (self-loop)*, jestliže $pFt \wedge tFp$. Neobsahuje-li N vlastní cyklus pak se nazývá *čistou sítí (pure net)*. □

DEF

■ **Definice 3.4** *Izolovaný prvek.*

Nechť $N = (P, T, F)$ je síť. Prvek $x \in (P \cup T)$ se nazývá *izolovaný*, jestliže ${}^\bullet x \cup x^\bullet = \emptyset$. □

x+y

■ **Příklad 3.3**

V síti z příkladu 3.2 platí např.

$${}^\bullet t_1 = \{p_1, p_2\}$$

$$p_2^\bullet = \{t_1\}$$

$$\{t_1, t_2\}^\bullet = {}^\bullet\{t_1, t_2\} = \{p_1, p_2, p_3\}$$

Tato síť je čistá a neobsahuje izolovaný prvek. Naproti tomu síť z obrázku 3.3 není čistá, protože obsahuje vlastní cyklus $\langle B, u \rangle$. □

DEF

■ **Definice 3.5** *Isomorfismus sítí.*

Nechť $N_1 = (P_1, T_1, F_1)$ a $N_2 = (P_2, T_2, F_2)$ jsou dvě sítě. Sítě N_1 a N_2 se nazývají *izomorfní*, jestliže existuje bijekce $\varphi : (P_1 \cup T_1) \leftrightarrow (P_2 \cup T_2)$ taková, že

$$x \in P_1 \Leftrightarrow \varphi(x) \in P_2$$

$$x F_1 y \Leftrightarrow \varphi(x) F_2 \varphi(x)$$

□

Poznamenejme, že grafická reprezentace sítě, ve které nejsou vrcholy explicitně pojmenovány, je jednoznačná až na isomorfismus. Takovou reprezentaci používáme v případech, kdy jména prvků nejsou důležitá.

Pojmy k zapamatování

- síť, vstupní a výstupní množina prvku
- vlastní cyklus, čistá síť
- isomorfismus sítí

Shrnutí

V této kapitole jsme se soustředili na vysvětlení základních konceptů Petriho sítí. Ukázali jsme si, jakým způsobem modelujeme změnu stavu systému v konečném automatu a v Petriho síti. A že právě v popsaném rozdílu spočívá důvod, proč se Petriho sítě využívají jako matematický model paralelních a distribuovaných výpočtů.

Příklady k procvičení

1. Definujte pojem síť a uveďte jednoduchý příklad.
2. Definujte pojem vstupní a výstupní množina.
3. Definujte pojem vlastní cyklus a čistá síť.
4. Definujte pojem izolovaný prvek.
5. Za jakých podmínek nazýváme dvě sítě isomorfní?

Část I

C/E sítě

Kapitola 4

Sítě z podmínek a událostí

Čas potřebný ke studiu: 9 hodin

Cíle kapitoly

Tato kapitola představí čtenářům základní koncepty C/E Petriho sítí a jejich matematické definice. Nejprve zavedeme pojem krok jako výskyt jedné události nebo více nezávislých událostí. Následně zavedeme ekvivalenci pro Condition/Event systémy a ukážeme, jak může být každý systém převeden na ekvivalentní bezkontaktní systém. Nakonec probereme případový graf C/E systému. Tento graf poskytuje přehled všech případů a kroků v systému.

Průvodce studiem

Na tuto kapitolu navazují ostatní kapitoly věnované C/E Petriho sítím, proto je důkladnému pochopení jednotlivých termínů, včetně vzájemných souvislostí, věnovat náležitou pozornost.

Obsah

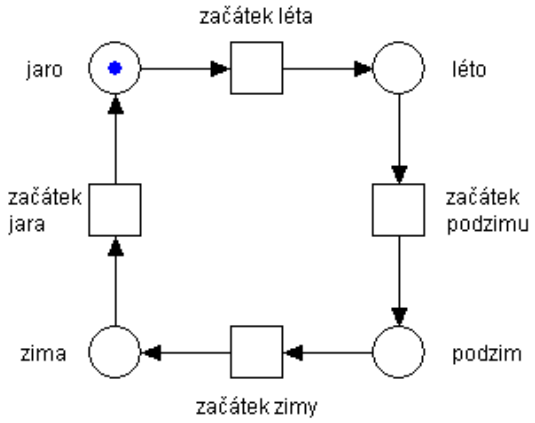
4.1 Podmínky a události

V předcházející kapitole jsme si představili základní koncepty obecných Petriho sítí. Zaměřme se nyní na systémy zahrnující *podmínky* (angl. *conditions*) a *události* (angl. *events*). Na obrázku 4.1 vidíme model s podmínkami „je jaro“, „je léto“, „je podzim“, „je zima“ a událostmi „začátek jara“, „začátek léta“, „začátek podzimu“ a „začátek zimy“. Podmínky jsou grafiky reprezentovány kružnicí a události čtvercem. Splněné podmínky obsahují *značku* (angl. *token*), na obrázku 4.1 je to podmínka „jaro“. Množina podmínek splněných ve stejný okamžik se nazývá *případ* . V příkladu na obrázku 4.1 obsahuje každý případ jen jednu podmínku. Při *provedení* (můžeme říkat i výskytu) (angl. *occurrence*) události nastává nový případ. V našem modelu změn ročních období se při provedení události vždy přesune značka do dalšího ročního období.

podmínka

událost

případ



Obrázek 4.1: Model změn ročních období

Podmínka a událost spolu mohou být ve vztahu:

1. Podmínka začne platit po provedení události. Podmínku potom nazýváme *výstupní podmínkou* (angl. *postcondition*) události. Graficky je tento vztah reprezentován hranou vedoucí z události do podmínky.

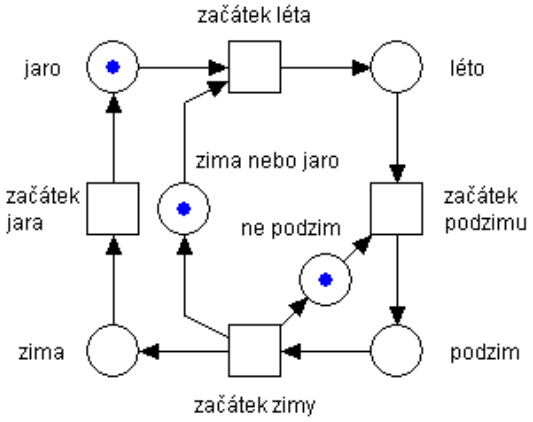
   výstupní podmínka

2. Podmínka platí při provedení události. Podmínku potom nazýváme *vstupní podmínkou* (angl. *precondition*) události. Graficky je tento vztah reprezentován hranou vedoucí z podmínky do události.

   vstupní podmínka

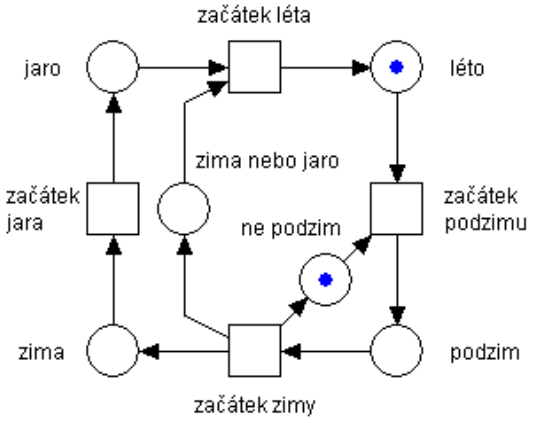
Pokud není podmínka nijak ovlivněna provedením události, žádná hrana je nespojuje.

Do modelu změn ročních období můžeme přidat ještě další podmínky a události, jak vidíme na obrázku 4.2, kde jsou oproti modelu na obrázku 4.1 navíc podmínky „zima nebo jaro“ a „ne podzim“. Nyní mají některé události více vstupních a výstupních podmínek.



Obrázek 4.2: Alternativní model k příkladu na obrázku 4.1

Podívejme se nyní podrobněji na model změn ročních období na obrázku 4.2 a zaměřme se na případ, ve kterém může nastat událost „začátek léta“. Ta může nastat tehdy, když jsou splněny podmínky „jaro“ a „zima nebo jaro“ a ještě nenastalo „léto“. Provedením této události dostáváme stav na obrázku 4.3. Řečeno obecně, událost může nastat, když jsou splněny všechny její vstupní podmínky a není splněna ani jedna z výstupních podmínek. Po provedení události přestanou platit vstupní podmínky a začnou platit výstupní podmínky.



Obrázek 4.3: Model z obrázku 4.2 po provedení události „začátek léta“

4.2 Případy a kroky

Podmínky v C/E síti jsou reprezentovány *S-prvky* (angl. *S-elements*) a události *T-prvky* (angl. *T-elements*). Protože S-prvky a T-prvky interpretujeme jako podmínky (něm. *Bedingung*) a události (něm. *Ereignis*), píšeme (B, E, F) místo (S, T, F).

DEF

■ **Definice 4.1** *C/E síť*

Nechť $N = (B, E, F)$ je síť.

1. Podmnožina $c \subseteq B$ se nazývá *případ* (angl. *case*)
2. Nechť $e \in E$ a $c \subseteq B$. Událost e je *proveditelná*, přesněji c-proveditelná, jestliže ${}^\bullet e \subseteq c \quad \wedge \quad e^\bullet \cap c = \emptyset$
3. Nechť $e \in E$, $c \subseteq B$ a nechť e je c-proveditelná. Případ $c' = (c \setminus {}^\bullet e) \cup e^\bullet$ se nazývá *následným případem* c (následníkem k c) při události e. Píšeme $c\ [e\rangle\ c'$

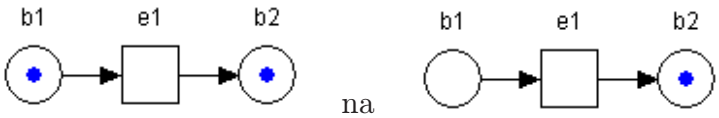
□

Jak jsme si již ukázali v předchozí podkapitole, případ c reprezentujeme graficky tak, že kreslíme *značku* v každé podmínce příslušející případu c.

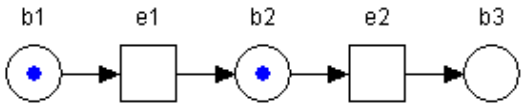
Podle definice 4.1 může být událost e provedena jen, když není splněna žádná z podmínek v její výstupní množině $e^\bullet$. Situace, kdy některá z výstupních podmínek brání provedení události e se nazývá *kontaktní situací* (angl. *contact-situation*). Tedy, že pro nějaké c a e nastane ${}^\bullet e \subseteq c \quad \wedge \quad e^\bullet \cap c \neq \emptyset$. Na první pohled nemusí být jasné, proč není dovoleno událost e provést. Můžete například namítnout, že každá výstupní podmínka, splněná před výskytem události e, zůstane splněna i nadále po provedení události e. Zamysleme se nad důsledky. Použijeme-li příkladu 4.1, znamenalo by to například, že začne jaro, když už jaro je. Nebo že jednou zapsaná paměťová buňka bude přepsaná znovu; že plná sklenice bude znovu naplněna; že jednou rezervované místo bude rezervováno znovu. Uvidíme dále, jak můžeme takovéto případy popsat, nalézt a předejít jim.

kontaktní situace

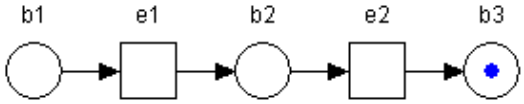
Nedovolit provedení události e v kontaktní situaci má i formální důvody. Předpokládejme, že připustíme následující přechod:



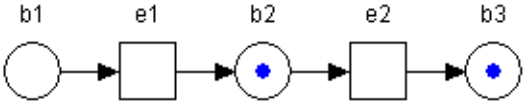
Potom v následující situaci, kdy události e_1 a e_2 provedeme právě jednou



bude záviset na pořadí jejich provedení a nevíme, zda výsledkem bude případ



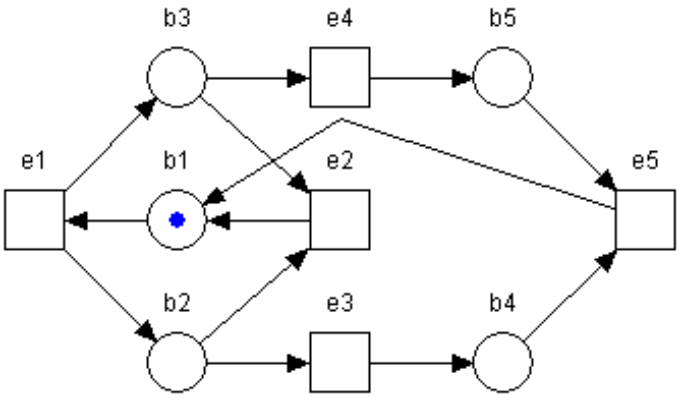
nebo případ



my ovšem chceme být schopni explicitně rozlišit a sledovat, zda se události vyskytují nezávisle nebo v určitém pořadí.

Události jsou na sobě navzájem různě závislé. Například v modelu na obrázku 4.4 musí událost e_1 nastat před událostí e_3 a e_4, události se předchází. Události e_3 a e_4 tvoří s událostí e_2 alternativu k sobě navzájem. Události e_3 a e_4 mohou být sloučeny (kombinovány) do jednoho *kroku* . Provedení několika událostí z množiny G v jednom kroku je možné, když jsou všechny události z G proveditelné a jejich vstupní a výstupní množiny jsou disjunktní. Množinu G nazýváme *nezávislou*.

krok



Obrázek 4.4: Ilustrační model

DEF

■ **Definice 4.2** *Nezávislá množina, krok*

Nechť $N = (B, E, F)$ je síť.

1. Množina událostí $G \subseteq E$ se nazývá *nezávislá* (angl. *detached*), jestliže $\forall e_1, e_2 \in G\colon\ e_1 \neq e_2 \ \Rightarrow\ {}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset = e_1^\bullet \cap e_2^\bullet$
2. Nechť c, c' jsou případy N a nechť $G \subseteq E$ je nezávislá množina událostí. G se nazývá *krokem* (angl. *step*) z c do c' (notace $c\ [G\rangle\ c'$), jestliže každá událost $e \in G$ je c-proveditelná a $c' = (c \setminus {}^\bullet G) \cup G^\bullet$

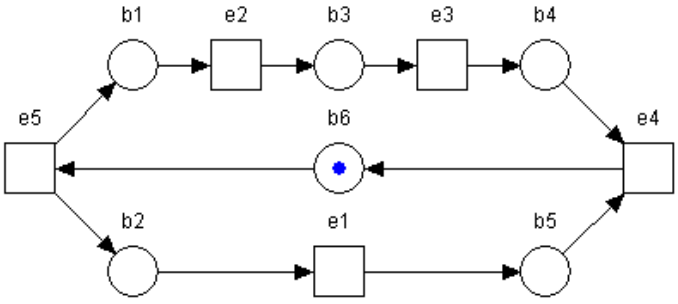
□

V kroku $c\ [G\rangle\ c'$ vede množina událostí G z případu c do případu c'. Pokud bude množina G obsahovat jen jeden prvek, potom platí: $G = \{e\} : c\ [G\rangle\ c' \Leftrightarrow c\ [e\rangle\ c'$. Následující lemma objasňuje některé vztahy mezi c, G a c'.

■ **Lemma 4.1** Lemma

Nechť N je síť, $G \subseteq E_N$ je nezávislá množina a c, c' jsou případy z C_N, potom: $c\ [G\rangle\ c' \ \Leftrightarrow\ c \setminus c' = {}^\bullet G \ \wedge\ c' \setminus c = G^\bullet$ □

Obecně máme několik možností, jak zkombinovat události do kroku. V modelu na obrázku 4.5 můžeme nalézt dva kroky. První krok $\{e_1, e_2\}$ z případu $\{b_1, b_2\}$ do případu $\{b_3, b_5\}$ a druhý krok $\{e_1, e_3\}$ z případu $\{b_2, b_3\}$ do případu $\{b_4, b_5\}$. Pokud je krok konečný, můžeme jej také realizovat provedením jeho událostí v libovolném pořadí.



Obrázek 4.5: Ilustrační model

Lemma

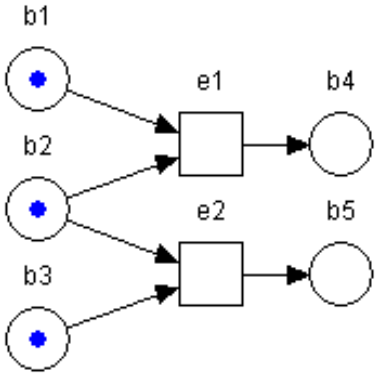
■ **Lemma 4.2**

Nechť N je síť, c, c' případy sítě N a nechť G je konečný krok z c do c'. Nechť $(e_1, e_2, \ldots, e_n)$ je libovolné uspořádání událostí kroku $G = \{e_1, e_2, \ldots, e_n\}$. Pak existují případy $c_0, c_1, \ldots, c_n$ takové, že
$c = c_0,\ c' = c_n$ a $c_{i-1}\ [e_i\rangle\ c_i \quad$ pro $\quad i = 1, \ldots, n$ □

Důkaz. Nechť $e, e' \in G$ a nechť c je případ, ve kterém jsou proveditelné obě události e, e'. Pak ${}^\bullet e \cap {}^\bullet e' = \emptyset \quad \wedge \quad e^\bullet \cap e'^\bullet = \emptyset$. Takže když $c\ [e\rangle\ c'$, pak ${}^\bullet e' \subseteq c'$. Analogicky platí $e'^\bullet \cap c' = \emptyset$, a tedy e' je proveditelná v c'. Zbytek indukce. □

Představme si nyní dvě události, které jsou v určitém případu proveditelné, ale mají společné některé podmínky ze vstupních nebo výstupních množin. Provedení takových dvou událostí se bude vzájemně vylučovat. Říkáme, že události jsou spolu v *konfliktu* (angl. *conflict*).

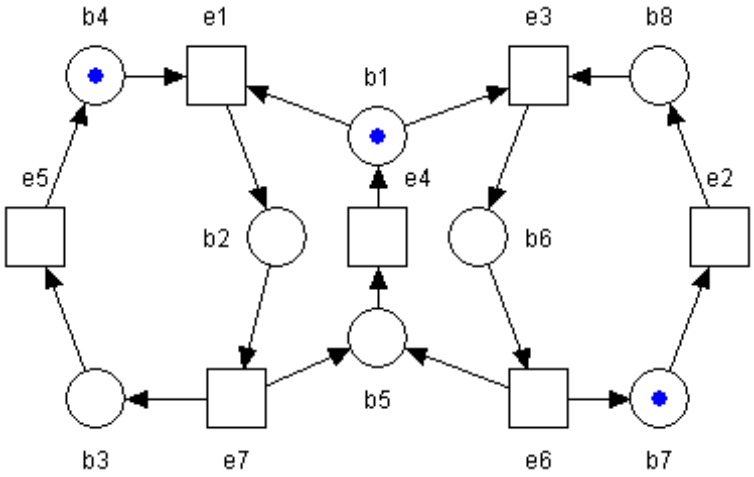
konflikt



Obrázek 4.6: Konflikt mezi událostmi e_1 a e_2 v podmínce b_2.

Konflikt nemusí být na první pohled v modelu patrný, jako například na obrázku 4.7. Jestliže se událost e_1 objeví před událostí e_2, konflikt mezi událostmi e_1 a e_3 nevznikne. Vyskytne-li se ovšem událost e_2 před událostí e_1, pak vzniká konflikt mezi událostmi e_1 a e_3 v podmínce b_1. Protože neexistuje žádná specifikace pořadí provádění událostí e_1 a e_2, je tato situace označována jako *zmatek* (angl. *confusion*).

zmatek



Obrázek 4.7: Zmatek (confusion)

4.3 Condition/Event systémy

Samotná definice C/E sítě nestačí k úplnému popisu reálných systémů. Je nutné ji rozšířit o uvažovanou množinu případů C. Například v modelu na obrázku 4.1 by pro správný popis ročních období nedával smysl případ, kde by bylo splněno současně více podmínek nebo kde by nebyla splněna žádná z nich. Množinu případů nazýváme *případovou třídou* a představuje všechny stavy, ve kterých se modelovaný systém může nacházet.

případová třída

Množina případů C má následující vlastnosti:

1. Existuje-li krok $G \subseteq E$ v případu $c \in C$, potom G vede znovu do případu z C (kroky nevedou mimo C).

2. A naopak, pokud případ $c \in C$ vyplývá z provedení kroku $G \subseteq E$, tak situace, ze které jsme vyšli byl také případ z C (jinými slovy, pokud uvažujeme provádění událostí pozpátku a sledujeme předchozí případy, nacházíme jen případy z C).

3. Množina C musí být dostatečně veliká:

   (a) Pro každou událost $e \in E$ existuje případ $c \in C$, ve kterém je událost e proveditelná.

   (b) Každá podmínka $b \in B$ patří alespoň do jednoho případu z C, avšak ne do každého. To vylučuje smyčky a izolované prvky.

V C/E systému se dále nedovoluje, aby měly dvě podmínky shodné vstupní a výstupní množiny. Takové podmínky by měly v každém z možných případů stejné značení a lze je nahradit jedinou podmínkou. Obdobně se nedovoluje, aby měly dvě události shodné vstupní a výstupní množiny. Takové události by byly proveditelné ve stejných případech a provedení každé z nich by vedlo do stejného případu. Události by tak byly od sebe nerozpoznatelné (systém pozorujeme z hlediska změny případů). Tím se vyloučí vlastní smyčky (angl. self-loop) a izolované podmínky. Je také nutné vyloučit izolované události, jejichž provedení by také nebylo rozpoznatelné na změně případu.

DEF

■ **Definice 4.3** *C/E systém*

Čtveřice $\Sigma = (B, E, F, C)$ se nazývá C/E systém, jestliže:

1. (B, E, F) je jednoduchá síť bez izolovaných prvků, $B \cup E \neq \emptyset$
2. $C \subseteq 2^B$ je ekvivalenční třídou vzhledem k relaci dosažitelnosti $R_\Sigma = (r_\Sigma \cup r_\Sigma^{-1})^*$, kde $r_\Sigma \subseteq 2^B \times 2^B$ je dána vztahem $c_1 r_\Sigma c_2 \overset{def.}{\iff} \exists G \subseteq E\colon c_1\ [G\rangle\ c_2$ C se nazývá případová třída (angl. *case class*) sítě Σ.
3. $\forall e \in E\ \exists c \in C$ tak, že e je c-proveditelná

□

C/E systémy nemají počáteční stav a každý z případů z případové třídy vede k vytvoření stejné případové třídy. Na obrázku 4.8 je uveden příklad jednoduchého C/E systému. V grafickém vyjádření je jeho případová třída zadána jediným případem $\{b_1\}$ a celá má tvar $C = \{\{b_1\}, \{b_2\}, \{b_3\}, \{b_4\}\}$.

Věta

■ **Věta 4.1**

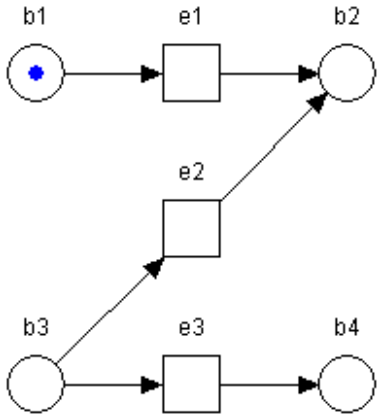
Nechť Σ je C/E systém.

1. $B_\Sigma \neq \emptyset \quad \wedge \quad E_\Sigma \neq \emptyset \quad \wedge \quad F_\Sigma \neq \emptyset$
2. Pro $c \in C_\Sigma$, $c' \subseteq B_\Sigma$ a $G \subseteq E_\Sigma$

   $c\ [G\rangle\ c' \ \Rightarrow\ c' \in C_\Sigma$

   $c'\ [G\rangle\ c \ \Rightarrow\ c' \in C_\Sigma$
3. $\forall b \in B_\Sigma\ \exists c, c' \in C_\Sigma$ tak, že $b \in c \quad \wedge \quad b \notin c'$
4. Σ je čistá síť

□



Obrázek 4.8: Jednoduchý C/E systém

■ **Věta 4.2** Věta

Nechť Σ je C/E systém a nechť $\hat{r} \subseteq 2^{B_\Sigma} \times 2^{B_\Sigma}$ je relace definována vztahem $c_1 \hat{r} c_2 \overset{def.}{\iff} \exists e \in E_\Sigma : c_1 \ [e\rangle \ c_2$. Je-li E_Σ konečná množina, pak $R_\Sigma = (\hat{r} \cup \hat{r}^{-1})^*$ □

Důkaz. Pro $\hat{R} = (\hat{r} \cup \hat{r}^{-1})^*$ platí triviálně $\hat{R} \subseteq R_\Sigma$. Protože E_Σ je konečná, každý krok sítě Σ je konečný, a proto z Lemma 4.1 plyne $r_\Sigma \subseteq \hat{r}^*$ a $r_\Sigma^{-1} \subseteq (r^{-1})^*$. Z toho pak dostaneme $R_\Sigma \subseteq \hat{R}$. □

4.4 Cyklické a živé systémy

DEF

■ **Definice 4.4** *Cyklický C/E systém*

C/E systém Σ se nazývá *cyklický*, jestliže $\forall c_1, c_2 \in C_\Sigma : \ c_1 r_\Sigma^* c_2$ □

■ **Věta 4.3** Věta

Nechť Σ je cyklický C/E systém a nechť $c \in C_\Sigma$. Pak $C_\Sigma = \{c' | c \, r_\Sigma^* c'\}$. □

DEF

■ **Definice 4.5** *Živý C/E systém*

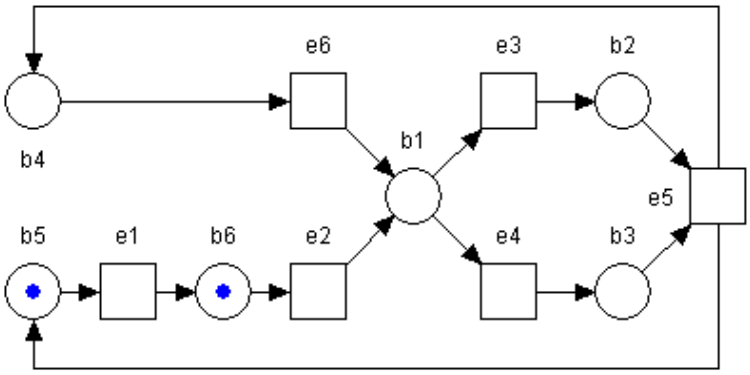
C/E systém Σ je *živý*, jestliže $\forall c \in C_\Sigma \ \ \forall e \in E_\Sigma \ \ \exists c' \in C_\Sigma$ takový, že $c \, r_\Sigma^* c'$ a e je c'-proveditelná. □

Věta

■ **Věta 4.4**

Každý cyklický C/E systém je živý. □

Důkaz. Nechť $c \in C_\Sigma$ a $e \in E_\Sigma$. Podle Definice 4.3 existuje $c' \in C_\Sigma$ takový, že e je c'-proveditelná. Podle Definice 4.4 platí $c\, r_\Sigma^* c'$. □



Obrázek 4.9: C/E systém, který je živý, ale není cyklický

Systém na obrázku 4.9 je živý, protože můžeme z každého případu pokračovat přes nějakou událost do dalšího případu. Tento systém ovšem není cyklický, protože případ $\{b_5, b_6\}$ není reprodukovatelný (již se do něj nelze vrátit).

4.5 Ekvivalence C/E systémů

Dva C/E systémy nazýváme *ekvivalentní*, jestliže si jejich případy a kroky navzájem odpovídají způsobem uvedeným v následující definici.

DEF

■ **Definice 4.6** *Ekvivalence C/E systémů*

Nechť Σ a Σ' jsou C/E systémy.

1. Jsou-li dány bijekce $\gamma : C_\Sigma \to C_{\Sigma'}$ a $\epsilon : E_\Sigma \to E_{\Sigma'}$, pak systémy Σ a Σ' nazýváme (γ, ϵ)-ekvivalentní, jestliže pro všechny případy $c_1, c_2 \in C_\Sigma$ a všechny množiny událostí $G \subseteq E_\Sigma$ platí: $c_1\ [G\rangle\ c_2 \ \Leftrightarrow\ \gamma(c_1)\ \ [\epsilon(G)\rangle\ \ \gamma(c_2)$

2. Σ a Σ' jsou *izomorfní* (angl. *isomorphic*), jestliže sítě $(B_\Sigma, E_\Sigma, F_\Sigma)$ a $(B_{\Sigma'}, E_{\Sigma'}, F_{\Sigma'})$ jsou izomorfní při bijekci β a jestliže $c \in C_\Sigma \ \Leftrightarrow\ \{\beta(b) \mid b \in c\} \in C_{\Sigma'}$

□

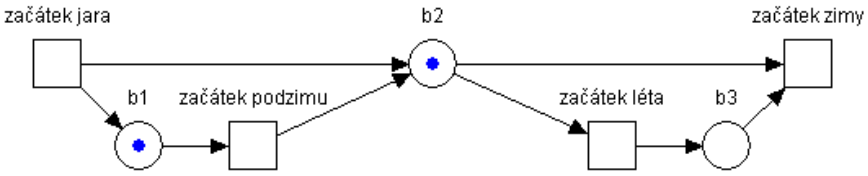
Notace: Jsou-li Σ a Σ' ekvivalentní, píšeme $\Sigma \sim \Sigma'$.

■ **Věta 4.5** Věta

$\sim$ je relace ekvivalence. □

■ **Věta 4.6** Věta

Ekvivalentní C/E systémy mají vždy stejný počet případů, událostí a kroků. Mohou se lišit v mohutnosti množin podmínek. □



Obrázek 4.10: C/E systém ekvivalentní se systémy z obrázku 4.1 a 4.2

Systémy na obrázku 4.1, 4.2 a 4.10 jsou ekvivalentní. Ekvivalence mezi systémy na obrázku 4.2 a 4.10 není na první pohled patrná, proto si ji popíšeme podrobněji. Oba systémy mají shodný počet kroků a případů. Kroky jsou tvořeny samostatnými událostmi a přímo si odpovídají i jejich názvy. Případová třída u systému na obrázku 4.10 je zvláštní tím, že v jednom z případů není splněna ani jedna podmínka. U událostí „začátek jara“ a „začátek zimy“ se využívá jejich prázdné vstupní, resp. výstupní množiny (značky se nám „generují“ a „ztrácejí“). Přiřazení případů ze systému z obrázku 4.1 k případům ze systému na obrázku 4.10 je následující:

$$\begin{array}{ccc} \{b_1, b_2\} & \equiv & \{\text{jaro}\} \\ \{b_1, b_3\} & \equiv & \{\text{léto}\} \\ \{b_2, b_3\} & \equiv & \{\text{podzim}\} \\ \emptyset & \equiv & \{\text{zima}\} \end{array}$$

■ **Věta 4.7** Věta

Nechť Σ a Σ' jsou ekvivalentní C/E systémy.

1. Σ je cyklický $\iff$ Σ' je cyklický
2. Σ je živý $\iff$ Σ' je živý

□

Sekvenční C/E systémy s jednoprvkovými případy (například systém z obrázku 4.1) odpovídají konečným automatům. Ekvivalence mezi dvěma takovými systémy se shoduje s izomorfismem.

Lemma

■ **Lemma 4.3**

Nechť Σ a Σ' jsou C/E systémy, pro které platí $\forall c \in C_\Sigma \cup C_{\Sigma'}: \ |c| = 1$.
Σ a Σ' jsou ekvivalentní, právě když jsou izomorfní. □

4.6 Bezkontaktní C/E systémy

V podkapitole 4.2 jsme si pověděli, v čem nám vadí kontaktní situace. Nyní si ukážeme, že kontaktu se lze vyhnout transformací C/E systému na ekvivalentní bezkontaktní C/E systém. K tomu přidáme pro každou podmínku b její komplement b' takový, že v každém z případů platí právě b nebo b'.

DEF

■ **Definice 4.7** *Komplement podmínky, úplný C/E systém*

Nechť Σ je C/E systém a nechť $b, b' \in B_\Sigma$.

1. b' se nazývá *komplement* (angl. *complement*) b, jestliže ${}^\bullet b = b'^\bullet$ a $b^\bullet = {}^\bullet b'$
2. Σ se nazývá *úplný* (angl. *complete*), jestliže každý prvek $b \in B_\Sigma$ má komplement $b' \in B_\Sigma$

□

Lemma

■ **Lemma 4.4**

Nechť Σ je C/E systém a nechť $b \in B_\Sigma$.

- b má nejvýše jeden komplement; označme jej $\widehat{b}$

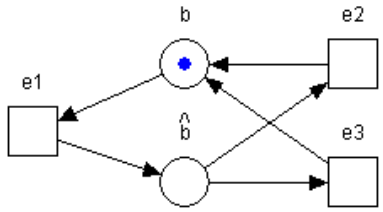
Jestliže b má komplement $\widehat{b}$, pak

- $\widehat{b}$ má komplement a $\widehat{\widehat{b}} = b$
- $\forall c \in C_\Sigma: \ b \in c \ \ \vee \ \ \widehat{b} \in c$

Je-li Σ úplný C/E systém, pak

- $\forall e \in E_\Sigma: \ |{}^\bullet e| = |e^\bullet|$
- $\forall c \in C_\Sigma: \ |c| = \frac{1}{2}|B_\Sigma|$

□



Obrázek 4.11: Podmínka b a její komplement $\widehat{b}$

■ **Definice 4.8** *Komplementace C/E systému* **DEF**

Nechť Σ je C/E systém a nechť $B \subseteq B_\Sigma$ je množina podmínek, které nemají komplement v B_Σ. Pro každé $b \in B$ nechť $\widehat{b}$ označuje nový prvek. Položme $F = \{(e,\widehat{b})|\ (b,e) \in F_\Sigma \ \wedge \ b \in B\} \cup \{(\widehat{b},e)|\ (e,b) \in F_\Sigma \ \wedge \ b \in B\}$. Pro $c \in C_\Sigma$ nechť $\varphi(c) = c \cup \{\widehat{b}|\ b \in B \ \wedge \ b \notin c\}$. Pak C/E systém $\widehat{\Sigma} = (B_\Sigma \cup \{\widehat{b}|\ b \in B\}, E_\Sigma, F_\Sigma \cup F, \varphi(C_\Sigma))$ je *komplementací systému* Σ. $\varphi(c)$ je komplementací c. □

■ **Věta 4.8** Věta

Nechť Σ je C/E systém a $c \in C_\Sigma$.

1. $\widehat{\widehat{\Sigma}} = \widehat{\Sigma}$
2. $\forall b \in B_\Sigma \ \forall c \in C_\Sigma\colon\ b \in \varphi(c) \ \Leftrightarrow \ \widehat{b} \notin \varphi(c)$
3. $c = \varphi(c) \cap B_\Sigma$

□

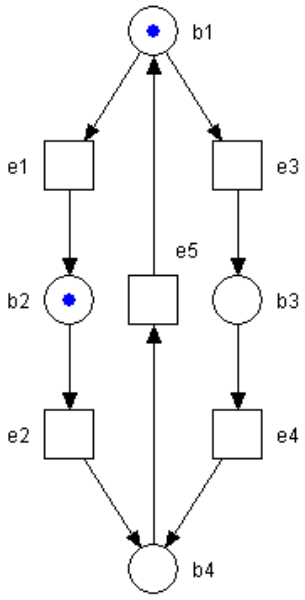
Notace: Nechť Σ je C/E systém a nechť $e \in E_\Sigma$. Označme ${}^-e$, resp. e^- *preset* resp. *postset* události e v $\widehat{\Sigma}$ (na rozdíl od ${}^\bullet e, e^\bullet$ v Σ).

■ **Věta 4.9** Věta

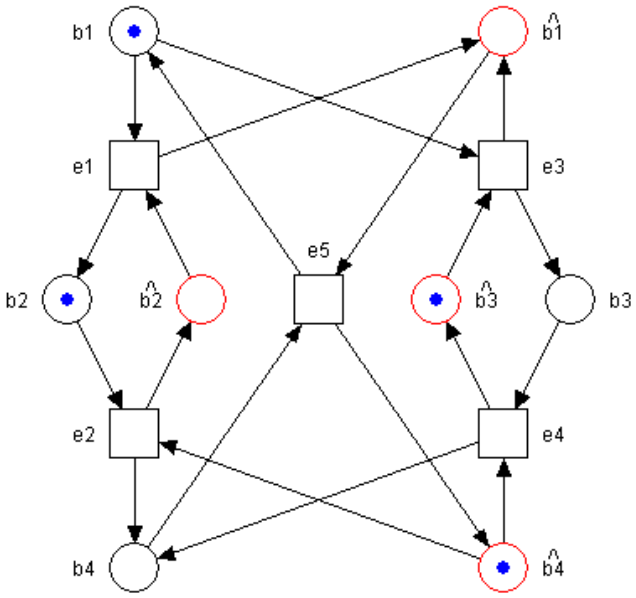
Nechť Σ je C/E systém a nechť $G \subseteq E_\Sigma$ a B je množina podmínek, které nemají komplement.

1. ${}^-G = {}^\bullet G \cup \{\widehat{b}|\ b \in B \ \wedge \ b \in G^\bullet\}$
   $G^- = G^\bullet \cup \{\widehat{b}|\ b \in B \ \wedge \ b \in {}^\bullet G\}$
2. ${}^\bullet G = {}^-G \cap B_\Sigma$, $G^\bullet = G^- \cap B_\Sigma$

□



Obrázek 4.12: Ilustrativní C/E systém Σ



Obrázek 4.13: Komplementovaný C/E systém $\widehat{\Sigma}$ od C/E systému Σ z obrázku 4.12

Na obrázku 4.13 jsou červeně zvýrazněny přidané komplementární podmínky.

■ **Věta 4.10** Věta

Je-li $\widehat{\Sigma}$ komplementací systému Σ, pak $\widehat{\Sigma}$ a Σ jsou ekvivalentní. □

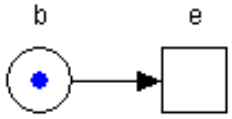
DEF

■ **Definice 4.9** *Bezkontaktní C/E systém*

Nechť Σ je C/E systém. Σ se nazývá *bezkontaktní* (angl. *contact-free*), jestliže pro každé $e \in E_\Sigma$ a každé $c \in C_\Sigma$ platí:

1. ${}^\bullet e \subseteq c \quad \Rightarrow \quad e^\bullet \subseteq B_\Sigma \backslash c$
2. $e^\bullet \subseteq c \quad \Rightarrow \quad {}^\bullet e \subseteq B_\Sigma \backslash c$

□



Obrázek 4.14: Podmínka 2. neplyne vždy s 1.

■ **Věta 4.11** Věta

1. Každý úplný C/E systém je bezkontaktní
2. Pro každý C/E systém existuje ekvivalentní bezkontaktní systém
3. Je-li Σ bezkontaktní, pak $\forall e \in E_\Sigma\colon \ {}^\bullet e \neq \emptyset \ \wedge \ e^\bullet \neq \emptyset$

□

Samozřejmě ne každý bezkontaktní C/E systém je úplný, například C/E systémy na obrázku 4.1, 4.2, 4.7, 4.5 a 4.4.

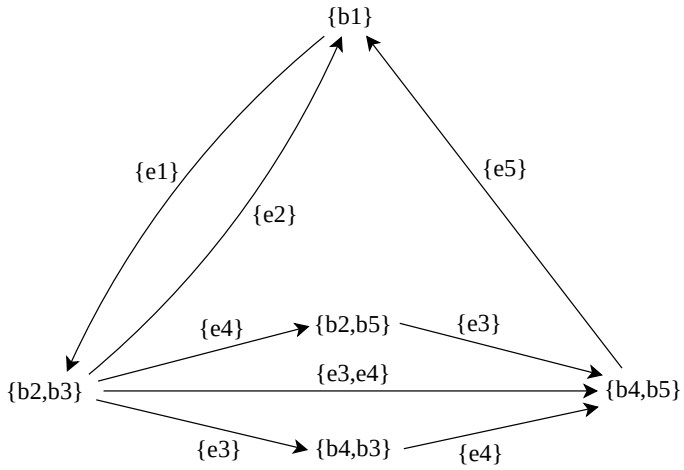
4.7 Případové grafy

Případový graf je orientovaný graf zobrazující vztah mezi případy z případové třídy daného C/E systému v závislosti na provádění událostí. Poskytuje nám ucelený pohled na všechny případy příslušného C/E systému. V grafické reprezentaci uzly představují případy systému a hrany reprezentují kroky mezi příslušnými případy. Hrany jsou ohodnoceny množinou událostí představující daný krok, uzly množinou podmínek představující daný případ.

DEF

■ **Definice 4.10** *Případový graf*

Nechť Σ je C/E systém, γ nechť je množina všech kroků systému Σ a nechť H je množina $H = \{(c_1, G, c_2) \in C_\Sigma \times \gamma \times C_\Sigma |\ c_1\ [G\rangle\ c_2\}$ Pak graf $\Phi_\Sigma = (C_\Sigma, H)$ se nazývá *případový graf* (angl. *case graph*) C/E systému Σ. □



Obrázek 4.15: Případový graf odpovídající C/E systému z příkladu 4.4

Věta

■ **Věta 4.12**

C/E systém Σ je cyklický, právě když je jeho případový graf silně souvislý. □

Důkaz. Σ je cyklický $\Leftrightarrow$ $\forall c, c' \in C_\Sigma :\ (c\ r_\Sigma^* c')$ $\Leftrightarrow$ $\forall c, c' \in C_\Sigma\ \exists G_1, \ldots, G_n \in \gamma\ \exists c_0, \ldots, c_n \in C_\Sigma :\ c_0\ [G_1\rangle\ c_1 \ldots\ [G_n\rangle\ c_n\ \wedge\ c_0 = c\ \wedge\ c_n = c'$ $\Leftrightarrow$ Φ_Σ je silně souvislý □

Věta

■ **Věta 4.13**

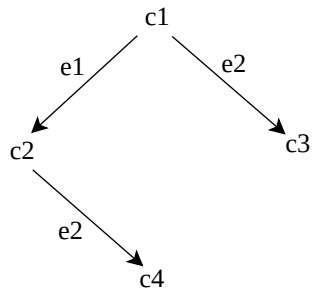
C/E systém Σ je živý, když a jen když pro každé $c_0 \in C_\Sigma$ a pro každé $e \in E_\Sigma$ existuje cesta v $\Phi_\Sigma :\ c_0 h_1 c_1 \ldots c_{n-1} h_n c_n$, kde $h_n = \{e\}$. □

Důkaz. Σ je živý $\Leftrightarrow$ $\forall c_0 \in C_\Sigma\ \forall e \in E_\Sigma\ \exists c, c' \in C_\Sigma :\ c_0 r_\Sigma^* c\ \wedge\ c\ [e\rangle\ c'$ $\Leftrightarrow$ v Φ existuje cesta $c_0 h_1 \ldots c_{n-1} h_n c_n$, kde $c_{n-1} = c$, $h_n = \{e\}$ a $c_n = c'$ □

■ **Věta 4.14** Věta

Dva C/E systémy jsou ekvivalentní, právě když jsou jejich případové grafy izomorfní. □

Ne každý graf můžeme chápat jako případový graf C/E systému. Například graf v příkladu 4.16 není případovým grafem žádného C/E systému. V případě c_1 jsou proveditelné události e_1 a e_2. Jestliže existuje konflikt mezi e_1 a e_2, pak e_2 není c_2-proveditelná a graf nesmí mít hranu $(c_2, \{e_2\}, c_4)$. Jestliže tento konflikt neexistuje, pak e_1 je proveditelná také v c_3 a tudíž chybí hrana $(c_3, \{e_1\}, c_4)$.



Obrázek 4.16: Graf, který není případovým grafem C/E systému

V silně paralelních systémech se případový graf stává velmi složitým. Krok, který obsahuje n událostí generuje $2^n - 1$ hran v případovém grafu.

■ **Věta 4.15** Věta

Nechť Σ je C/E systém, $c_1, c_2, c_3 \in C_\Sigma$ a $G_1, G_2 \subseteq E_\Sigma$.

1. Jestli $c_1 G_1 c_2 G_2 c_3$ je cesta v Φ_Σ, pak $G_1 \cap G_2 = \emptyset$
2. Nechť $G_1 \cap G_2 = \emptyset$. Jestli $c_1(G_1 \cup G_2)c_3$ je hrana v Φ_Σ, pak existuje $c \in C_\Sigma$ tak, že $c_1 G_1 c G_2 c_3$ je také cesta v Φ_Σ.

□

Důkaz.

1. $e \in G_1 \Rightarrow c_2 \cap {}^\bullet e = \emptyset \Rightarrow e$ není c_2-proveditelná $\Rightarrow e \notin G_2$
2. $c_1(G_1 \cup G_2)c_2$ je hrana $\Phi_\Sigma \Rightarrow c_1\ [G_1 \cup G_2\rangle\ c_2 \Rightarrow c_1\ [G_1\rangle\ c \ \wedge \ c\ [G_2\rangle\ c_2$, kde $c = (c_1 \setminus {}^\bullet G_1) \cup G_1^\bullet$

□

Pojmy k zapamatování

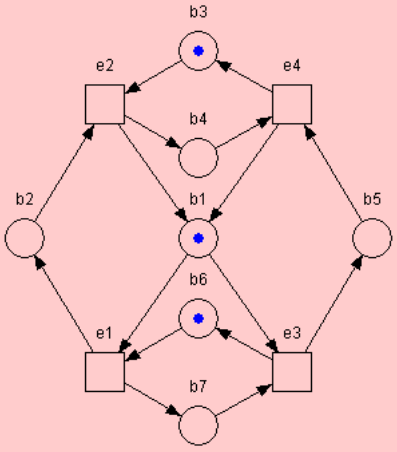
- C/E síť, případ, krok, konflikt
- C/E systém, případová třída, cyklický a živý C/E systém
- ekvivalence C/E systémů, komplementace, bezkontaktní C/E systém
- případový graf

Shrnutí

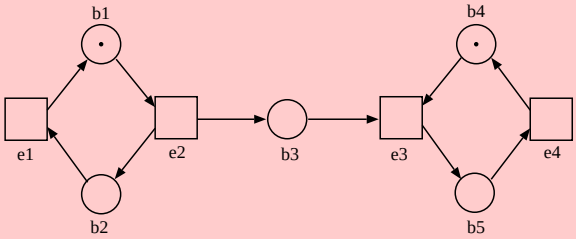
Tato kapitola si kladla za cíl uvést čtenáře do základů C/E Petriho sítí. Nadefinovali jsme si C/E systém jako jednoduchou síť bez izolovaných prvků rozšířenou o případovou třídu. Na konci kapitoly jsme zavedli případové grafy, které jsou jednou z nejčastěji používaných možností analýzy C/E systémů. Zobrazují uceleně všechny případy z případové třídy a všechny proveditelné události v jednotlivých případech. Lze z nich také snáze rozpoznat, zda jsou dva systémy ekvivalentní, takové systémy mají izomorfní případové grafy.

Příklady k procvičení

1. K C/E systému na obrázku vytvořte ekvivalentní C/E systém s minimálním počtem podmínek.



2. Sestrojte případový graf a vypište případovou třídu C/E systému na obrázku.



3. Definujte pojmy C/E síť a C/E systém.

4. Definujte pojem bezkontaktní C/E systém.

5. Definujte pojem případový graf a neformálně popište algoritmus jeho konstrukce.

Kapitola 5

Procesy C/E systémů

Čas potřebný ke studiu: 9 hodin

Cíle kapitoly

Cílem této kapitoly je definovat procesy C/E systémů pomocí výskytových sítí. K tomu si nejprve zavedeme pojmy z částečně uspořádaných množin a nadefinujeme s nimi výskytové sítě. Procesy potom definujeme jako zobrazení z omezené výskytové sítě do bezkontaktního C/E systému. Toto zobrazení musí ještě splňovat další požadavky vysvětlené dále v této kapitole. Budeme také hledat vztah mezi procesy a cestami v případovém grafu.

Průvodce studiem

Na tuto kapitolu navazují ostatní kapitoly věnované C/E Petriho sítím, proto je důkladnému pochopení jednotlivých termínů, včetně vzájemných souvislostí, věnovat náležitou pozornost.

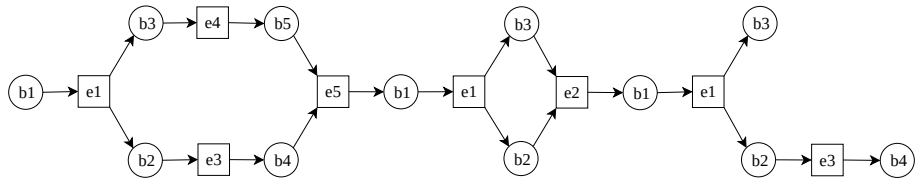
Obsah

5.1 Problém reprezentace procesů

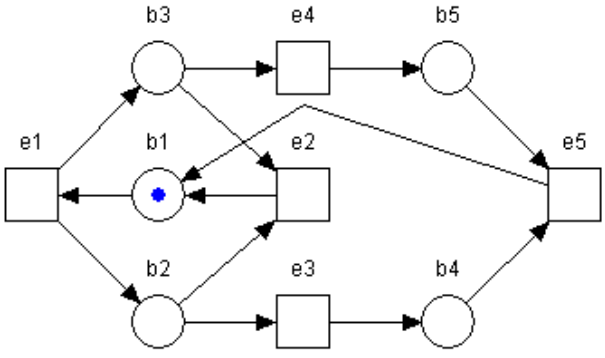
Na první pohled by se mohlo zdát jednoduché definovat proces C/E systému jako cestu v jeho případovém grafu. Taková reprezentace by ovšem nebyla zcela adekvátní. Úplné seřazení prvků nám nedává žádnou informaci o tom, zda se události ve skutečnosti vyskytovaly jedna za druhou nebo zda jsou na sobě nezávislé. Částečné uspořádání, ve kterém jsou události prováděny, je v případovém grafu popsáno pouze nepřímo a to množinou všech možných výskytů událostí jako posloupnost kroků.

Musíme proto hledat vhodnější popis procesů, který je především jednoznačný a zobrazuje explicitně, zda se události provedly souběžně. Uvažujme takový popis jako záznam výskytů událostí a změn příslušných podmínek. Položky v takovém záznamu jsou částečně uspořádané relací „a je kauzální předpoklad pro b“. Opakování stejné události nebo stejné podmínky je zaznamenáno jako nová položka.

Reprezentaci uvedeného záznamu provedeme sítí, jako například na obrázku 5.1, který odpovídá provedení posloupnosti
$\{b_1\}\ [e_1\rangle\ \{b_2, b_3\}\ [e_3, e_4\rangle\ \{b_4, b_5\}\ [e_5\rangle\ \{b_1\}\ [e_1\rangle\ \{b_2, b_3\}\ [e_2\rangle\ \{b_1\}\ [e_1\rangle\ \{b_2, b_3\}\ [e_3\rangle\ \{b_3, b_4\}$ v C/E systému na obrázku 5.2.



Obrázek 5.1: Síťová reprezentace procesu



Obrázek 5.2: Ilustrativní C/E systém

V síťové reprezentaci procesů odpovídají T-prvky provedení události. T-prvky jsou označené názvem příslušné události. Dva T-prvky se stejným označením představují dva rozdílné výskyty stejné události. Podobně S-prvky jsou označené názvem příslušné podmínky. Názvy T-prvků a S-prvků sítě neuvádíme, protože pro nás nejsou důležité.

Jak je vidět na obrázku 5.1, S-prvky nejsou větveny, protože konflikty v C/E systému na obrázku 5.2 (mezi událostmi $\{e_2\}$ a $\{e_3, e_4\}$) již byly vyřešeny provedením událostí v konkrétním pořadí.

5.2 Částečně uspořádané množiny

Ukážeme si nyní některé vlastnosti relace, která popisuje vztah „kauzální závislosti a nezávislosti“.

DEF

▪ **Definice 5.1** *Částečné uspořádaní*

Nechť M je množina. Binární relace $\rho \subseteq M \times M$ se nazývá *částečné uspořádání* (angl. *partial order*), jestliže $\forall a, b, c \in M$

1. $\neg(a \rho a)$ [ρ je ireflexivní]
2. $a \rho b \ \wedge \ b \rho c \ \Rightarrow \ a \rho c$ [ρ je tranzitivní]

Částečné uspořádání ρ budeme zapisovat symbolem $<$ (bez ohledu na nosič). Dále $a \leq b \ \Leftrightarrow \ a < b \ \vee \ a = b$ □

DEF

▪ **Definice 5.2** *Relace podobnosti, oblast*

Nechť A je množina. Binární relace $\rho \subseteq A \times A$ se nazývá *relací podobnosti* (angl. *similarity relation*), jestliže

1. $\forall a \in A\colon \ a \rho a$ [ρ je reflexivní]
2. $\forall a, b \in A\colon \ a \rho b \ \Rightarrow \ b \rho a$ [ρ je symetrická]

Podmnožina $B \subseteq A$ se nazývá *oblastí* (angl. *region*) relace podobnosti ρ, jestliže

1. $\forall a, b \in B\colon \ a \rho b$ [ρ je úplnou relací na B]
2. $\forall a \in A: \ a \notin B \ \Rightarrow \ \exists b \in B: \ \neg(a \rho b)$ [B je maximální podmnožina, na které je ρ úplná]

□

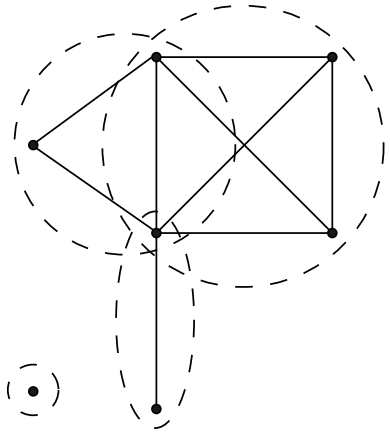
■ **Věta 5.1** Věta

Nechť ρ je relace podobnosti na A.

1. Každý prvek $a \in A$ patří alespoň do jedné oblasti relace ρ.
2. Žádná oblast není vlastní podmnožinou jiné oblasti. Oblasti neprázdné množiny A nejsou prázdné.
3. Je-li ρ zároveň ekvivalence, pak její oblasti jsou shodné s třídami rozkladu generovaného touto ekvivalencí.

□

Konečnou relaci podobnosti nad množinou A reprezentujeme graficky neorientovaným grafem, kde A je množina vrcholů a $K = \{(a,b)|\ a \neq b \quad \wedge \quad a\rho b\}$ je množina hran. Příklad grafické reprezentace relace podobnosti je na obrázku 5.3, jednotlivé oblasti jsou ohraničeny čárkovaně.



Obrázek 5.3: Relace podobnosti a její 4 oblasti

DEF

■ **Definice 5.3** *Relace $\underline{li}$ a $\underline{co}$*

Nechť A je částečně uspořádaná množina.

1. Nechť $\underline{li} \subseteq A \times A$ je binární relace $a \ \underline{li}\ b \quad \Leftrightarrow \quad a < b \quad \vee \quad b < a \quad \vee \quad a = b$
2. Nechť $\underline{co} \subseteq A \times A$ je binární relace $a \ \underline{co}\ b \quad \Leftrightarrow \quad \neg(a \ \underline{li}\ b) \quad \vee \quad a = b$ tzn. $a \ \underline{co}\ b \quad \Leftrightarrow \quad \neg(a < b) \quad \wedge \quad \neg(b < a) \quad \vee \quad a = b$

□

V relaci $\underline{li}$ jsou prvky řazeny lineárně (jsou na jedné přímce). V relaci $\underline{co}$ jsou prvky souběžné (jsou k sobě „paralelní“).

Věta

■ **Věta 5.2**

Nechť A je částečně uspořádaná množina a nechť $a, b \in A$.

1. $a \underline{li}\ b \quad \vee \quad a \underline{co}\ b$
2. $a \underline{li}\ b \quad \wedge \quad a \underline{co}\ b \quad \Leftrightarrow \quad a = b$

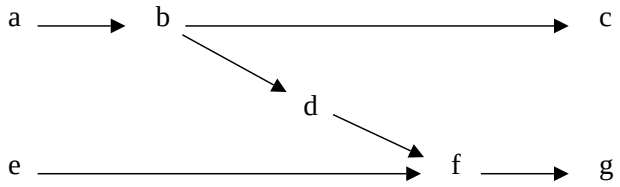
□

Věta

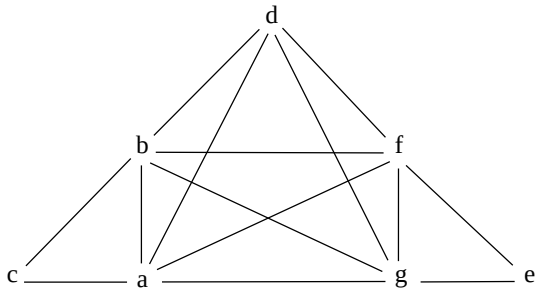
■ **Věta 5.3**

Pro každou částečně uspořádanou množinu jsou binární relace $\underline{li}$ a $\underline{co}$ relacemi podobnosti. □

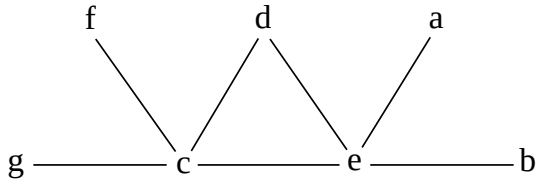
Obrázek 5.5 zobrazuje relaci $\underline{li}$ a obrázek 5.6 relaci $\underline{co}$ z částečně uspořádané množiny na obrázku 5.4.



Obrázek 5.4: Částečně uspořádaná množina



Obrázek 5.5: Odpovídající grafická reprezentace relace $\underline{li}$



Obrázek 5.6: Odpovídající grafická reprezentace relace co

■ **Definice 5.4** *Řetěz, řez* DEF

Nechť A je částečně uspořádaná množina a $B \subseteq A$.

1. B se nazývá *řetězcem* (angl. *line*), je-li oblastí li
2. B se nazývá *řezem* (angl. *cut*), je-li oblastí co

□

x+y

■ **Příklad 5.1**

Částečné uspořádání z obrázku 5.4 má:

- 3 řetězce: $\{a, b, c\}$, $\{e, f, g\}$, $\{a, b, d, f, g\}$
- 5 řezů: $\{e, a\}$, $\{e, b\}$, $\{e, d, c\}$, $\{f, c\}$, $\{g, c\}$

□

■ **Věta 5.4** Věta

Nechť A je částečně uspořádaná množina a $B \subseteq A$.

1. B je řetězcem, právě když
   (a) $\forall a, b \in B\colon\ a < b \ \vee \ b < a \ \vee \ a = b$
   (b) $\forall a \in A \backslash B\ \ \exists b \in B\colon\ \neg(a < b \ \vee \ b < a)$
2. B je řezem, právě když
   (a) $\forall a, b \in B\colon\ \neg(a < b \ \vee \ b < a)$
   (b) $\forall a \in A \backslash B\ \ \exists b \in B\colon\ (a < b \ \vee \ b < a)$

□

DEF

■ **Definice 5.5** *Omezená množina*

Nechť A je částečně uspořádaná množina a $B, C \subseteq A$.

1. Množina A se nazývá *omezená* (angl. *bounded*), jestliže existuje $n \in N$ takové, že pro každý řetězec množiny A platí $|L| \leq n$

2. B *předchází* (angl. *precedes*) C (píšeme $B \leq C$), jestliže $\forall b \in B \;\; \forall c \in C\colon \; b < c \;\; \vee \;\; b \underline{co} \; c$
($B < C$ značí $B \leq C$ a $B \neq C$)

3. Nechť $B^- = \{a \in A |\; \{a\} \leq B\}$ a $B^+ = \{a \in A |\; B \leq \{a\}\}$

4. Nechť ${}^\circ B = \{b \in B |\; \forall b' \in B\colon \; b \underline{co} \; b' \;\; \vee \;\; b < b'\}$ a $B^\circ = \{b \in B |\; \forall b' \in B\colon \; b \underline{co} \; b' \;\; \vee \;\; b' < b\}$.

□

Zřejmě ${}^\circ A$ obsahují „minimální prvky“ množiny A a A° pak „maximální prvky“ množiny A.

Věta

■ **Věta 5.5**

Je-li A omezená částečně uspořádaná množina, pak ${}^\circ A$ a A° jsou řezy.
□

Důkaz. Nechť a, b jsou libovolné prvky ${}^\circ A$. Pak $a \underline{co} \; b$, protože $\neg(a < b \;\; \vee \;\; b < a)$. Nechť $c \in A \backslash {}^\circ A$ a nechť L je řetězec a $c \in L$. Protože L je konečná množina, existuje $d \in L \cap {}^\circ A$, a proto $d < c$. Podle věty 5.4 (2) je ${}^\circ A$ řezem. Pro A° analogicky. □

Věta

■ **Věta 5.6**

Nechť A je částečně uspořádaná množina, L řetězec a D řez množiny A. Pak $|L \cap D| \leq 1$. □

Důkaz. Nechť $a, b \in L \cap D$. Pak $a \underline{li} \; b$, protože $a, b \in L$. Avšak $a \underline{co} \; b$, protože $a, b \in D$. Podle věty 5.2 (2) je $a = b$. □

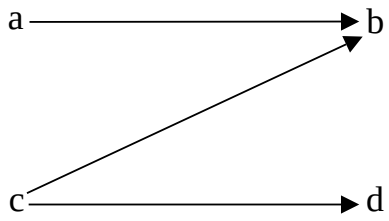
Věta 5.6 nám říká, že řez a řetěz mají nanejvýše jeden společný prvek.

DEF

■ **Definice 5.6** *K-hustá množina*

Částečně uspořádaná množina A se nazývá *K-hustá* (angl. *K-dense*), jestliže každý řetězec má neprázdný průnik s každým řezem. □

Částečné uspořádání na obrázku 5.4 je K-husté. Avšak ne každé částečné uspořádaní je K-husté, jak ukazuje obrázek 5.7,
kde $\{c, b\} \cap \{a, d\} = \emptyset$



Obrázek 5.7: Částečně uspořádaná množina, která není K-hustá

5.3 Výskytové sítě

Výskytové sítě si zavedeme jako sítě bez cyklů, kde S-prvky nejsou větveny. Dostáváme tak částečné uspořádání prvků výskytové sítě a budeme moci používat pojmy řetězce, řezy, omezenost a K-hustota, zavedené v předchozí podkapitole.

DEF

■ **Definice 5.7** *Výskytová síť*

Síť $K = (S_K, T_K, F_K)$ se nazývá *výskytová síť* (angl. *occurrence net*), jestliže

1. $\forall a, b \in K:\ aF_K^+b \ \Rightarrow\ \neg(bF_K^+a)$, tj. K nemá cykly

2. $\forall s \in S_K:\ |{}^\bullet s| \leq 1 \ \wedge\ |s^\bullet| \leq 1$, tj. místa sítě nejsou větvena

□

Věta

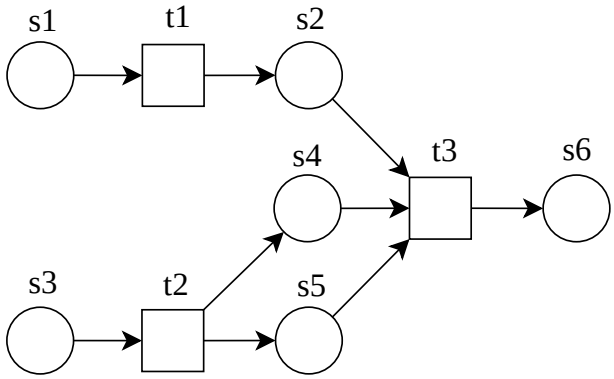
■ **Věta 5.7**

Nechť K je výskytová síť. Relace $<$ definována: $a < b \ \overset{def.}{\Longleftrightarrow}\ aF_K^+b$ pro všechna $a, b \in K$ je částečné uspořádání na K. □

DEF

■ **Definice 5.8** *S-řez*

S-řezem (angl. *slice*) nazýváme řez, který obsahuje pouze místa. Označme $\underline{sl}(K)$ množinu všech S-řezů sítě K. □



Obrázek 5.8: Příklad výskytové sítě

x+y

■ **Příklad 5.2**

Síť z obrázku 5.8 má 3 řetězce, 11 řezů a z toho 5 S-řezů:

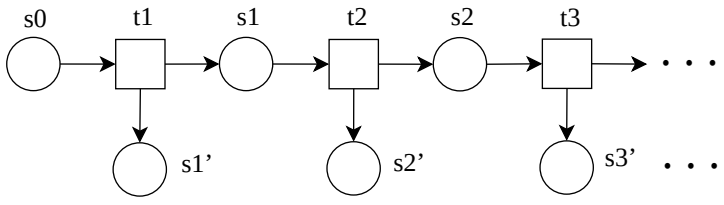
- řetězce: $\{s_1, t_1, s_2, t_3, s_6\}$, $\{s_3, t_2, s_4, t_3, s_6\}$, $\{s_3, t_2, s_5, t_3, s_6\}$
- řezy: $\{s_1, s_3\}$, $\{s_1, t_2\}$, $\{s_1, s_4, s_5\}$, $\{t_1, s_3\}$, $\{t_1, t_2\}$, $\{t_1, s_4, s_5\}$, $\{s_2, s_3\}$, $\{s_2, t_2\}$, , $\{s_2, s_4, s_5\}$, $\{t_3\}$, $\{s_6\}$
- S-řezy: $\{s_1, s_3\}$, $\{s_1, s_4, s_5\}$, $\{s_2, s_3\}$, $\{s_2, s_4, s_5\}$, $\{s_6\}$

□

Věta

■ **Věta 5.8**

Každá omezená neprázdná výskytová síť je K-hustá. □



Obrázek 5.9: Síť, která není K-hustá

Na obrázku 5.9 je uvedena neomezená výskytová síť, která tedy není K-hustá, protože $\{s_0, t_1, s_1, \ldots\} \cap \{s'_1, s'_2, \ldots\} = \emptyset$

Pokud by tato síť byla konečná, pak by muselo platit
$|\{s_0, t_1, s_1, \ldots, t_n, s'_n\} \cap \{s'_1, \ldots, s'_n\}| = 1$

5.4 Procesy C/E systémů

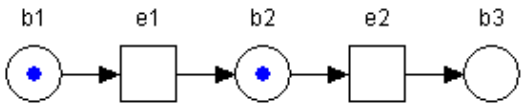
Proces definujeme jako zobrazení z omezené výskytové sítě do bezkontaktního C/E systému, které splňuje 2 požadavky:

1. Každý S-řez sítě je injektivně zobrazen na nějaký případ C/E systému

2. Zobrazení T-prvků výskytové sítě na události C/E systému respektuje okolí události

Omezení na bezkontaktní C/E systémy není nijak tvrdé, protože každý C/E systém můžeme převést na ekvivalentní bezkontaktní C/E systém, jak jsme si ukázali v předchozí kapitole o C/E systémech.

Proč musí být proces definován na bezkontaktním C/E systému si ukážeme na obrázku 5.10. Událost e_1 může být provedena až po provedení události e_2, protože se nachází v kontaktní situaci. V procesu popisujícím přesné pořadí provedení událostí e_1 a e_2 musí být jasně patrné, že před provedením události e_1 přestala platit podmínka b_2. A toho nelze dosáhnout jinak, než zavedením komplementu podmínky b_2.



Obrázek 5.10: Kontaktní situace

DEF

■ **Definice 5.9** *Proces*

Nechť K je omezená výskytová síť a nechť Σ je bezkontaktní C/E systém. Zobrazení $p\colon\ K \to \Sigma$ se nazývá *proces* (angl. *process*) systému Σ, jestliže pro každý S-řez D sítě K a každý přechod $t \in T_K$ platí:

1. $p|D$ je injektivní a $p(D) \in C_\Sigma$

2. $p({}^\bullet t) = {}^\bullet p(t) \quad \wedge \quad p(t^\bullet) = p(t)^\bullet$

□

V grafické reprezentaci procesů $p : K \to \Sigma$ je každý prvek x sítě K označen (angl. labelled) svým obrazem $p(x)$, jak jsme si již ukázali na obrázku 5.1 na začátku kapitoly.

Věta

■ **Věta 5.9**

Pro každý proces $p : K \to \Sigma$ platí:

1. $p(S_K) \subseteq B_\Sigma \quad \wedge \quad p(T_K) \subseteq E_\Sigma$, tj. p zachovává druhy
2. $\forall x, y \in K\colon\ xF_K y \quad \Rightarrow \quad p(x) F_\Sigma\, p(y)$, tj. p zachovává relaci toku
3. $\forall x, y \in K\colon\ p(x) = p(y) \quad \Rightarrow \quad x\ \underline{li}\ y$, tj. podmínky a události nejsou vzájemně souběžné
4. $\forall t \in T_K\colon\ {}^\bullet t \neq \emptyset \quad \wedge \quad t^\bullet \neq \emptyset$, tj. události mají předchůdce a následníky
5. pro každý řez D sítě $K\colon\ p|D$ je injektivní

□

Věta

■ **Věta 5.10**

Nechť $p : K \to \Sigma$ je proces, nechť pro $T \subseteq T_K$ platí $\forall t_1, t_2 \in T :$ $t_1\ \underline{co}\ t_2$. Pak $\exists c_1, c_2 \in C_\Sigma\colon\ c_1\ [p(T)\rangle\ c_2$. □

Důkaz. Zřejmě $\forall s_1, s_2 \in {}^\bullet T\colon\ s_1\ \underline{co}\ s_2$. Pak tedy existuje $D \in \underline{sl}(K)$, pro který ${}^\bullet T \subseteq D$. Podle definice 5.9 $p(D) \in C_\Sigma$ a ${}^\bullet p(T) = p({}^\bullet T) \subseteq p(D)$. Dále $\forall s \in T^\bullet\ \ \exists s_1 \in D$ takové, že $s_1 < s$. Proto $T^\bullet \cap D = \emptyset$ a také $p(D) \cap p(T^\bullet) = p(D) \cap p(T)^\bullet = \emptyset$. Tudíž $p(T)$ je $p(D)$-proveditelný krok. □

DEF

■ **Definice 5.10** *Izomorfní procesy*

Dva procesy $p_1\colon\ K_1 \to \Sigma$ a $p_2\colon\ K_2 \to \Sigma$ od C/E systému Σ nazýváme *izomorfní* (angl. *isomorphic*), jestliže K_1 je β-izomorfní k K_2 a $\forall x \in K_1 : p_1(x) = p_2(\beta(x))$. □

Věta

■ **Věta 5.11**

Každý bezkontaktní C/E systém je plně charakterizován množinou svých procesů. Proces $p\colon\ K \to \Sigma$ je skutečně reprezentován množinou dvojic $\{(x, p(x))|\ x \in K\}$. □

■ **Věta 5.12** Věta

Nechť Σ_1, Σ_2 jsou dva bezkontaktní C/E systémy a nechť P_i jsou množiny procesů systému Σ_i $(i = 1, 2)$. Pak $P_1 = P_2 \Leftrightarrow \Sigma_1 = \Sigma_2$. □

Důkaz. Nechť $\Sigma_i = (B_i, E_i, F_i, C_i)$, $i = 1, 2$ a nechť $\Sigma_1 \neq \Sigma_2$. Pak existuje $b \in B_1 \cup B_2$ nebo $e \in E_1 \cup E_2$ nebo $c \in C_1 \cup C_2$ takový, že $b \in B_1 \backslash B_2$ nebo $e \in E_1 \backslash E_2$ nebo $(b, e) \in F_1 \backslash F_2$ nebo $(e, b) \in F_1 \backslash F_2$ nebo $c \in C_1 \backslash C_2$. Pak ale existuje krok $c_1\ [e'\rangle\ c_2$ v Σ_1, který není možný v Σ_2 (vyber $b \in c_1 \cup c_2$ nebo $e' = e$ nebo $c = c_1$ nebo $c = c_2$). Pro $K = (S, \{t\}, F)$ nechť $p\colon K \to \Sigma_1$ je proces, pro který $p({}^\circ K) = c_1$ a $p(K^\circ) = c_2$ a $p(t) = e'$. Pak $p \in P_1 \backslash P_2$. □

5.5 Kompozice procesů

Za předpokladu, že proces p_1 končí ve stejném případu (stavu), ve kterém proces p_2 začíná, definujeme kompozici jako $p_1 \circ p_2$

■ **Lemma 5.1** Lemma

Je-li $p\colon K \to \Sigma$ proces, pak ${}^\circ K$ a K° jsou S-řezy množiny K. □

Důkaz. Podle věty 5.5 jsou ${}^\circ K$ a K° řezy množiny K. Protože Σ je bezkontaktní, pak pro každé $e \in E_\Sigma$ je ${}^\bullet e \neq \emptyset$ a $e^\bullet \neq \emptyset$. Z definice 5.9 (2) plyne ${}^\circ K \cup K^\circ \subseteq S_K$. □

■ **Lemma 5.2** Lemma

Nechť $p_i : K_i \to \Sigma$, $i = 1, 2$ jsou dva procesy, pro které $p_1(K_1^\circ) = p_2({}^\circ K_2)$. Pak existuje právě jedna (až na izomorfizmus) výskytová síť K a S-řez D této sítě a proces $p : K \to \Sigma$ takový, že $p|D^- = p_1$ a $p|D^+ = p_2$ □

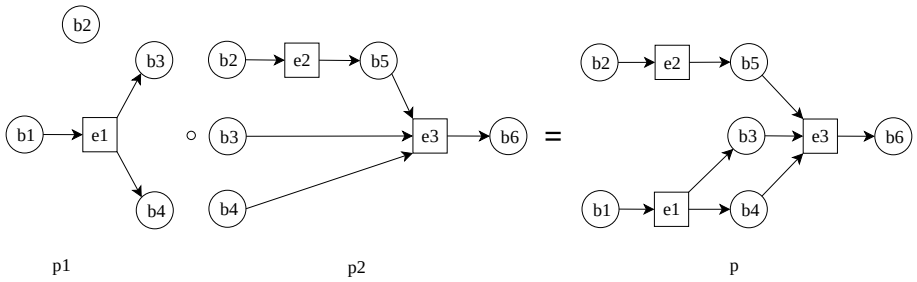
Důkaz. Nechť $K_i = (S_i, T_i, F_i)$, $i = 1, 2$ a nechť $(S_1 \cup T_1) \cap (S_2 \cup T_2) = K_1^\circ = {}^\circ K_2$. Pak síť $K = (S_1 \cup S_2, T_1 \cup T_2, F_1 \cup F_2)$, $D = K_1^\circ = {}^\circ K_2$ a proces p definovaný předpisem $p(x) = p_i(x) \overset{def.}{\Longleftrightarrow} x \in K_i$, $i = 1, 2$ splňuje tvrzení lemmy. □

DEF

■ **Definice 5.11** *Kompozice procesů*

Nechť p_1, p_2, p jsou procesy splňující lemma 5.2. Proces p se nazývá kompozicí procesů p_1 a p_2. Kompozici procesů zapisujeme $p = p_1 \circ p_2$ □

Příklad kompozice procesů je uveden na obrázku 5.11.



Obrázek 5.11: Kompozice procesů

Věta

■ **Věta 5.13**

Nechť $p: K \to \Sigma$ je proces a nechť D je S-řez množiny K. Nechť $p^- = p|D^-$ a $p^+ = p|D^+$. Pak p^- a p^+ jsou procesy a $p = p^- \circ p^+$. (Každý S-řez dělí proces na zkomponovatelné podprocesy.) □

Věta

■ **Věta 5.14**

Nechť p_1, p_2, p_3 jsou procesy, pro které jsou definovány $p_1 \circ p_2$ a $p_2 \circ p_3$. Pak $p_1 \circ (p_2 \circ p_3) = (p_1 \circ p_2) \circ p_3$. (Kompozice procesů je asociativní.) □

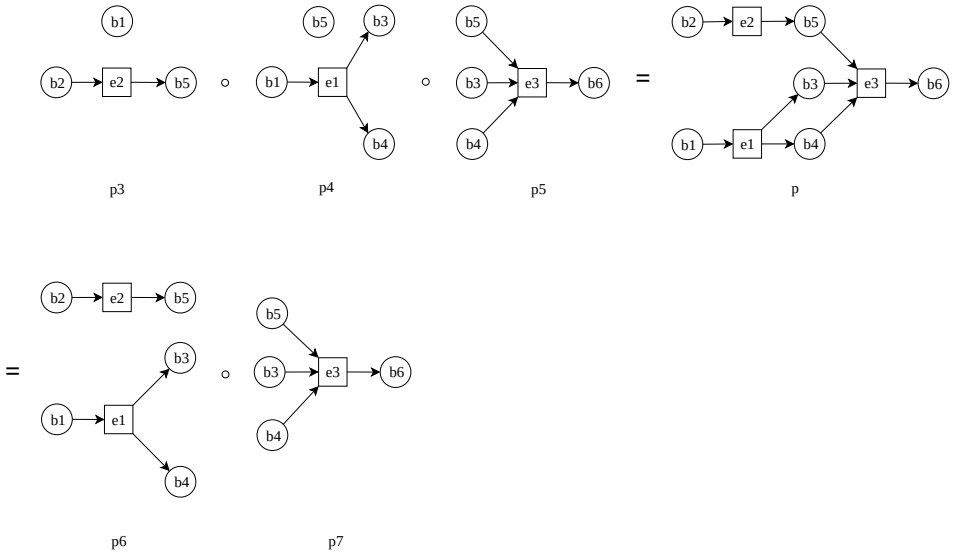
Proces nazýváme *elementární* (angl. *elementary*), jestliže popisuje jednoduchý krok. Procesy jsou rozložitelné na konečně mnoho elementárních procesů.

DEF

■ **Definice 5.12** *Elementární proces*

Proces $p: K \to \Sigma$ je *elementární*, jestliže $S_K = {}^\circ K \cup K^\circ$ □

Na obrázku 5.12 vidíme příklad kompozice procesu p užitím elementárních procesů p_3, p_4, p_5 nebo p_6, p_7. Proces p můžeme v tomto příkladě rozložit na dva nebo na tři elementární procesy díky tomu, že elementární proces odpovídá jednomu kroku (viz. věta 5.15). Události e_1 a e_2 tedy můžeme provést odděleně v procesech p_3 a p_4 nebo společně v procesu p_6.



Obrázek 5.12: Kompozice procesů

■ **Věta 5.15** Věta

1. $p\colon\ K \to \Sigma$ je elementární proces $\Leftrightarrow$ $p(^\circ K)\ \ [p(T_K)\rangle\ \ p(K^\circ)$ je krok v Σ
2. Jestliže $p\colon\ K \to \Sigma$ je elementární, pak $\forall t_1, t_2 \in T_K\colon\ t_1\ \underline{co}\ t_2$

□

■ **Definice 5.13** *Prázdný proces*

DEF

Proces $p\colon\ K \to \Sigma$ se nazývá *prázdný* (angl. *empty*), jestliže $T_K = \emptyset$.
□

■ **Věta 5.16** Věta

1. Každý prázdný proces je elementární
2. Je-li p' prázdný proces a je-li definováno $p \circ p'$ (resp. $p' \circ p$), pak $p = p \circ p'$ (resp. $p = p' \circ p$)

□

Věta

■ **Věta 5.17**

Je-li $p\colon\ K \to \Sigma$ proces, pak existuje konečně mnoho elementárních procesů $p_1, p_2, \ldots, p_n$ takových, že $p = p_1 \circ p_2 \circ \ldots \circ p_n$ □

Důkaz. Existuje největší celé číslo m ohraničující počet T-prvků každého řetězce množiny K. Důkaz provedeme indukcí na m. Je-li $m = 0$, pak $T_K = \emptyset$ a p je prázdný. Jestliže nejdelší řetězec v K má $m{+}1$ T-prvků, pak p je rozložitelný na p' a p'' tak, že $p = p' \circ p''$; nejdelší řetězec p' obsahuje m T-přechodů; a p'' je elementární neprázdný proces. Indukčním předpokladem p' je komponovatelné z elementárních procesů $p_1, \ldots, p_n$, $p' = p_1 \circ \ldots \circ p_n$ a odtud
$p = p_1 \circ \ldots \circ p_n \circ p''$. □

5.6 Procesy a případové grafy

Nyní budeme hledat vztah mezi procesy a cestami v případovém grafu.

Lemma

■ **Lemma 5.3**

Nechť Σ je bezkontaktní C/E systém. Proces $p\colon\ K \to \Sigma$ je elementární proces, právě když existuje hrana $v = (c_1, G, c_2)$ grafu Φ_Σ taková, že $p(^\circ K) = c_1$, $p(K^\circ) = c_2$ a $p(T_K) = G$. □

Důkaz. Jestliže $p\colon\ K \to \Sigma$ je elementární proces, pak $p(^\circ K)\ [p(T_K)\rangle\ p(K^\circ)$ je krokem Σ a $(p(^\circ K), p(T_K), p(K^\circ))$ je hranou grafu Φ_Σ. Naopak, je-li (c_1, G, c_2) hrana Φ_Σ, pak $c_1\ [G\rangle\ c_2$. Nechť $K = (c_1 \cup c_2, G, F_\Sigma \cap (c_1 \cup c_2 \cup G)^2)$. Pak $id\colon\ K \to \Sigma$ je elementární proces. □

DEF

■ **Definice 5.14**

Nechť Σ je bezkontaktní C/E systém.

1. Je-li v hrana grafu Φ_Σ, pak $\tilde{v}$ označuje proces odpovídající hraně v. $\tilde{v}$ se nazývá *procesem hrany* v; v je *hranou procesu* $\tilde{v}$.

2. Nechť $v_1, v_2, \ldots, v_n$ jsou hrany a $w = v_1 v_2 \ldots v_n$ je cesta v Φ_Σ. Pak $\tilde{w} = \tilde{v_1} \circ \tilde{v_2} \circ \ldots \circ \tilde{v_n}$ se nazývá *procesem cesty* w; w je *cestou procesu* $\tilde{w}$.

3. Pro $v = (c_1, G, c_2)$ a $e \in G$ nechť $t(v, e) = \tilde{v}^{-1}(e)$ a $\tau(v) = \{t(v, e) |\ e \in G\}$ $t(v, e)$ a $\tau(v)$ označuje jednoduchý přechod, resp. množinu přechodů odpovídající výskytové sítě.

□

DEF

■ **Definice 5.15**

Nechť Σ je C/E systém, $c_1, c_2, c_3 \in C_\Sigma$ a $G_1, G_2 \subseteq E_\Sigma$.

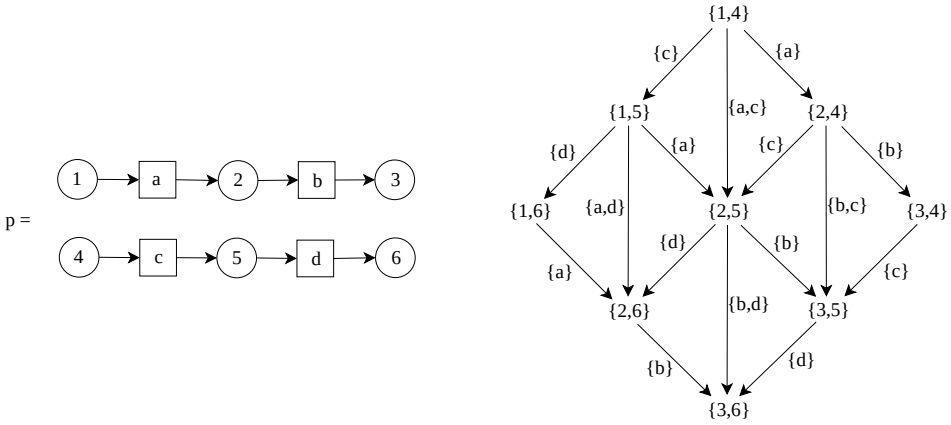
1. Jestliže $u_1 = c_1 G_1 c_2$, $u_2 = c_2 G_2 c_3$ a $v = c_1(G_1 \cup G_2)c_3$ jsou hrany případového grafu Φ_Σ, pak cesta $u_1 u_2$ se nazývá *dekompozicí cesty* v a v se nazývá *unifikací cesty* $u_1 u_2$.

2. Nechť w, w' jsou cesty v Φ_Σ. w' se nazývá *permutací cesty* w, jestliže existují cesty $u_1, \ldots, u_4$ takové, že $w = u_1 u_2 u_3$, $w' = u_1 u_4 u_3$ a u_4 je dekompozicí nebo unifikací cesty u_2

3. Nechť $w_1, w_2, \ldots, w_n$ jsou cesty v Φ_Σ. $(w_1, \ldots, w_n)$ se nazývá *permutační posloupností*, jestliže pro $i = 1, \ldots, n-1$ je cesta w_{i+1} *permutací cesty* w_i

□

Každé cestě v případovém grafu odpovídá přesně jeden proces. Naopak existuje několik cest odpovídající jednomu procesu. Příklad vidíme na obrázku 5.13, kde každá z 13 cest v případovém grafu z případu $\{1, 4\}$ do případu $\{3, 6\}$ odpovídá procesu p.



Obrázek 5.13: Proces a případový graf

Věta

■ **Věta 5.18**

Nechť Σ je bezkontaktní C/E systém, $c_1, c_2, c_3 \in C_\Sigma$ a nechť $G_1, G_2 \subseteq E_\Sigma$ jsou disjunktní a neprázdné.

1. Jestliže $v = c_1(G_1 \cup G_2)c_2$ je hrana v Φ_Σ, pak existuje dekompozice cesty v ve tvaru $c_1 G_1 c G_2 c_2$ pro nějaké $c \in C_\Sigma$
2. Nechť $u_1 = c_1 G_1 c_3$ a $u_2 = c_3 G_2 c_2$ jsou hrany grafu Φ_Σ a nechť $\tilde{u_1} \circ \tilde{u_2} \colon K \to \Sigma$. Pak $\forall t_1, t_2 \in T_K \colon t_1 \underline{co}\, t_2 \Leftrightarrow c_1(G_1 \cup G_2)c_2$ je hrana v Φ_Σ.

□

Důkaz. (ad. 2) $\forall t_1, t_2 \in T_K \colon t_1 \underline{co}\, t_2$ právě když existuje elementární proces $p \colon K \to \Sigma$, pro který $p({}^\circ K) = c_1$, $p(K^\circ) = c_2$. A $p(T_K) = G_1 \cup G_2$, právě když $c_1(G_1 \cup G_2)c_2$ je hrana v Φ_Σ (viz. Lemma 5.1). □

Věta

■ **Věta 5.19**

Dvě cesty w, w' v grafu Φ_Σ přísluší stejnému procesu, právě když existuje permutační posloupnost z w do w'. □

Pojmy k zapamatování

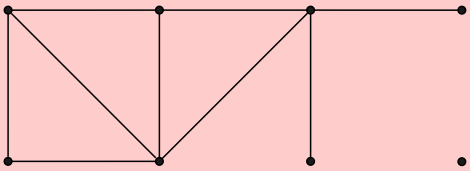
- relace podobnosti, řetěz, řez, S-řez
- výskytová síť
- proces C/E systému, kompozice procesů
- elementární proces

Shrnutí

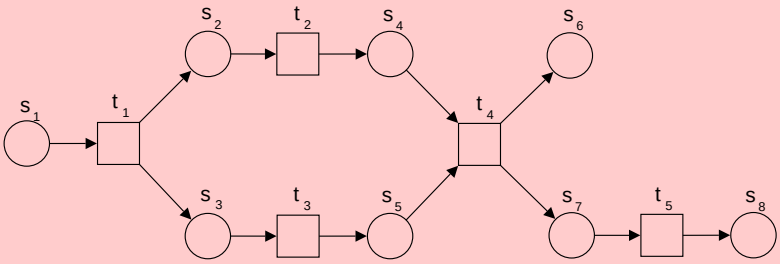
V této kapitole jsme se soustředili na vysvětlení procesů C/E systémů a to pomocí výskytových sítí. K tomu jsme si nejprve zavedli pojmy z částečně uspořádaných množin a nadefinovali s nimi výskytové sítě. Zkoumali jsme vztah mezi procesy a cestami v případovém grafu.

Příklady k procvičení

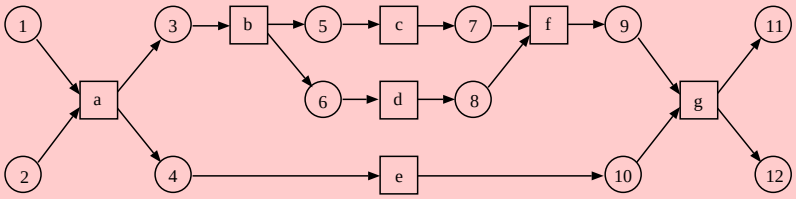
1. Vyznačte všechny oblasti relace podobnosti na obrázku.



2. Uveďte, kolik řetězců, řezů a S-řezů obsahuje výskytová síť na obrázku.



3. Rozložte proces z obrázku na elementární procesy.



4. Nechť Σ je konečný cyklický C/E systém a $E_1, E_2 \subseteq E_\Sigma$. Dokažte, že $\sigma(E_1, E_2) = \omega$, když existuje proces $p\colon K \to \Sigma$ takový, že $p(^\circ K) = p(K^\circ)$ a $\nu'(p, E_1, E_2) = ||p^{-1}(E_1)| - |p^{-1}(E_2)|| > 0$.

Kapitola 6

Vlastnosti C/E systémů

Čas potřebný ke studiu: 8 hodin

Cíle kapitoly

Cílem této kapitoly je definovat synchronizační vzdálenost, což je vlastnost C/E systému, která udává stupeň závislosti mezi výskyty jeho událostí, tj. jak je výskyt jedné události závislý na předchozím výskytu jiné události(í). Speciálně se také budeme věnovat synchronizačním vzdálenostem v sekvenčních a cyklických C/E systémech. S využitím podmínek C/E systému je možno konstruovat formule výrokové logiky. Ukážeme, jak lze reprezentaci a vyhodnocení těchto formulí začlenit do „síťového kalkulu“. Pro reprezentaci libovolné pravdivostní formule sestavené z podmínek systému obohatíme C/E systém o nové přechody, které nejsou proveditelné v žádném případu systému. Takové „mrtvé“ přechody nazveme fakta.

Průvodce studiem

Touto kapitolou uzavřeme C/E Petriho sítě a budeme se dále věnovat P/T Petriho sítím. C/E Petriho sítě jsou podtřídou P/T Petriho sítí, základní pojmy se tedy shodně objevují v obou sítích a je vhodné před pokračováním v další kapitole provést rekapitulaci a projít si znovu předchozí kapitoly.

Obsah

6.1 Synchronizační vzdálenosti

Důležitou vlastností systému je stupeň závislosti mezi výskyty jeho událostí, tj. jak je výskyt jedné události závislý na předchozím výskytu jiné události(í). Například v systému na obrázku 4.1 jsou události „konec zimy“ a „začátek jara“ silně svázány (silně synchronizovány), říkáme, že *koincidují* (angl. *coincident*). Události ovšem mohou být i méně svázány. Například události e_2 a e_3 na obrázku 4.7 *alternují* (angl. *alternate*), události e_1 a e_2 na obrázku 4.5 jsou *souběžné* (angl. *concurrent*), události e_1 a e_2 na obrázku 4.7 jsou zcela *nezávislé* (angl. *independent*) nebo se události mohou vyskytovat v *libovolném pořadí* (angl. *arbitrary order*).

V této kapitole zavedeme míru synchronizace událostí. Budeme uvažovat obecně dvojici množin událostí $E_1, E_2 \subseteq E_\Sigma$. Pozorujme, jak často se události množiny E_1 a události množiny E_2 objevují v každém procesu p systému. Absolutní rozdíl počtu jejich vzájemných výskytů nazýváme *variancí* E_1 a E_2 v procesu p. Suprémum variancí ve všech procesech systému se nazývá *synchronizační vzdálenost* $\sigma(E_1, E_2)$ množin E_1 a E_2. Dá se ukázat, že σ má vlastnost metriky. Synchronizační vzdálenost je tak prostředkem pro získání kvantitativní informace o dynamickém chování systému, aniž je třeba zavádět pojem „času“.

synchronizační vzdálenost

Znovu se omezíme na bezkontaktní C/E systémy, protože teorie synchronizační vzdálenosti je založena na procesech. K zavedení synchronizační vzdálenosti dvou množin $E_1, E_2 \subseteq E_\Sigma$ uvažujme proces $p : K \to \Sigma$ a počítejme prvky $p^{-1}(E_1)$ a $p^{-1}(E_2)$. Protože nás zajímají největší rozdíly výskytů událostí z E_1 a E_2, spočítáme pro všechny S-řezy D_1, D_2 sítě K prvky $p^{-1}(E_1)$ a $p^{-1}(E_2)$ mezi D_1 a D_2. K tomu účelu, pro všechny $M \subseteq T_K$ zavedeme nejprve míru $\mu(M, D_1, D_2)$ pro „počítání událostí“. Je-li $D_1 < D_2$, položme $\mu(M, D_1, D_2) = |M \cap D_1^+ \cap D_2^-|$. Je-li $D_2 < D_1$, potom $\mu(M, D_1, D_2) = |M \cap D_1^- \cap D_2^+|$.

Problém je však v tom, že S-řezy mohou být nesrovnatelné, proto definujeme míru μ následovně:

DEF

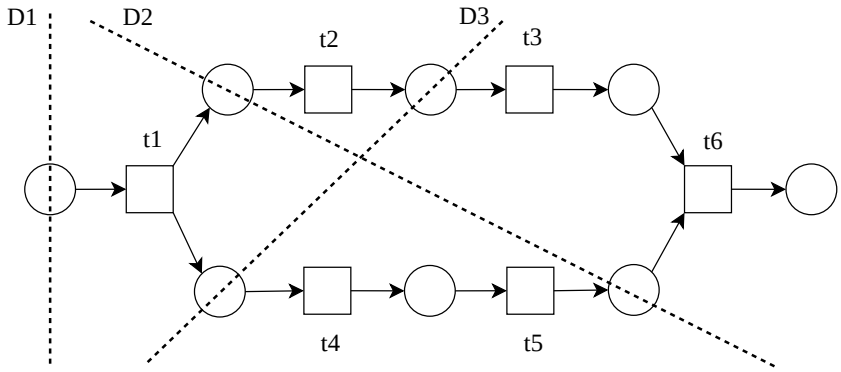
■ **Definice 6.1** *Míra μ*

Nechť K je výskytová síť a D_1, D_2 její dva S-řezy. Nechť $M \subseteq T_K$ je konečná množina. Pak nechť
$\mu(M, D_1, D_2) = |M \cap D_1^+ \cap D_2^-| - |M \cap D_1^- \cap D_2^+|$ □

Věta

■ **Věta 6.1**

Pro všechny konečné podmnožiny $M \subseteq T_K$ a S-řezy D_1, D_2 z výskytové sítě K platí: $\mu(M, D_1, D_2) = -\mu(M, D_2, D_1)$ □



Obrázek 6.1: Ilustrativní proces

x+y

■ **Příklad 6.1**

Vybrané hodnoty míry μ z procesu na obrázku 6.1:

- $\mu(\{t_1\}, D_1, D_2) = 1$
- $\mu(\{t_4, t_5\}, D_2, D_3) = -2$
- $\mu(\{t_2, t_3\}, D_2, D_3) = 1$
- $\mu(\{t_2, t_3\}, D_3, D_2) = -1$
- $\mu(\{t_2, t_4\}, D_3, D_2) = 0$

□

S využitím míry μ nyní definujeme varianci dvou množin událostí v procesu.

DEF

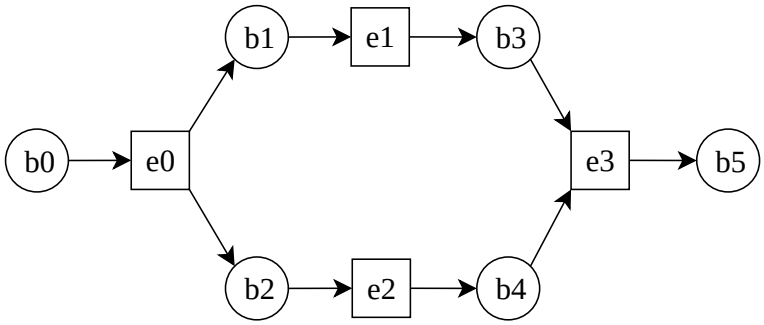
■ **Definice 6.2**

Nechť Σ je bezkontaktní C/E systém. Označme π_Σ množinu všech jeho procesů. □

DEF

■ **Definice 6.3** *Variance ν*

Nechť Σ je bezkontaktní C/E systém a nechť $p : K \to \Sigma \in \pi_\Sigma$ a $E_1, E_2 \subseteq E_\Sigma$. Potom $\nu(p, E_1, E_2) = \max\{\ \mu(p^{-1}(E_1), D_1, D_2) - \mu(p^{-1}(E_2), D_1, D_2) \mid\ D_1, D_2 \in \underline{sl}(K)$ se nazývá *variancí* (angl. *variance*) množin událostí E_1 a E_2 v procesu p. □



Obrázek 6.2: Ilustrativní proces

x+y

■ **Příklad 6.2**

Vybrané hodnoty variance ν z procesu na obrázku 6.2:

- $\nu(p, \{e_0\}, \{e_3\}) = 1$
- $\nu(p, \{e_0\}, \{e_1, e_2\}) = 2$
- $\nu(p, \{e_0, e_1\}, \{e_2\}) = 2$
- $\nu(p, \{e_1\}, \{e_2\}) = 2$

□

■ **Věta 6.2** Věta

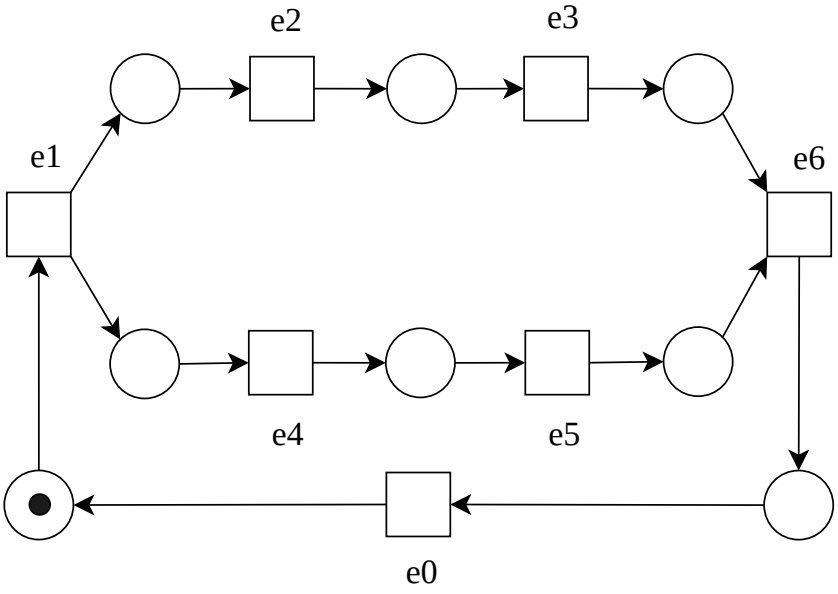
Pro každý proces $p : K \to \Sigma$ a každý pár $E_1, E_2 \subseteq E_\Sigma$ platí:
$\nu(p, E_1, E_2) = \nu(p, E_2, E_1)$ □

Synchronizační vzdálenost dvou množin událostí můžeme nyní definovat jako supremum variancí ve všech konečných procesech.

DEF

■ **Definice 6.4** *Synchronizační vzdálenost* σ

Nechť Σ je bezkontaktní C/E systém a nechť $E_1, E_2 \subseteq E_\Sigma$ jsou konečné množiny. $\sigma(E_1, E_2) = \sup\{\ \nu(p, E_1, E_2) \mid\ p \in \pi_\Sigma\}$ se nazývá *synchronizační vzdálenost* (angl. *synchronic distance*) množin událostí E_1 a E_2. □



Obrázek 6.3: Ilustrativní C/E systém

x+y

■ **Příklad 6.3**

Vybrané hodnoty synchronizační vzdálenosti σ z C/E systému na obrázku 6.3:

- $\sigma(\{e_4\}, \{e_0\}) = 1$
- $\sigma(\{e_2\}, \{e_4\}) = 2$
- $\sigma(\{e_2, e_3\}, \{e_4, e_5\}) = 4$
- $\sigma(\{e_2, e_4\}, \{e_3, e_5\}) = 2$
- $\sigma(\{e_4, e_5\}, \{e_3\}) = \omega$

□

6.2 Grafická reprezentace synchronizační vzdálenosti

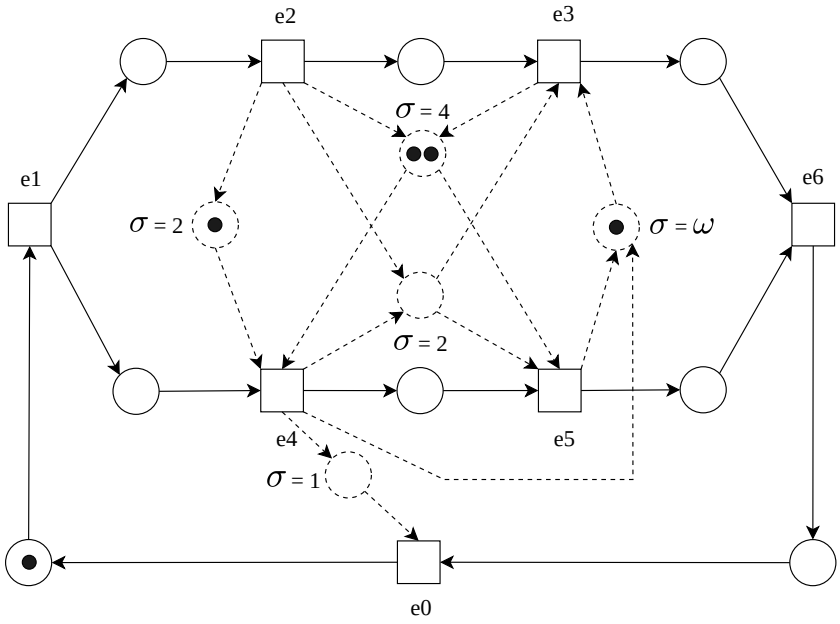
Pro reprezentaci synchronizační vzdálenosti množin událostí E_1 a E_2 v C/E systému Σ zavedeme nové místo s: ${}^\bullet s = E_1$ a $s^\bullet = E_2$. Místo s není podmínka C/E systému Σ, ale může obsahovat libovolný počet

značek. V každém případu c systému Σ obsahuje s určitý počet značek (dostatečně mnoho, aby nebránilo provedení událostí). Kdykoliv se provede událost z E_1, resp. E_2, počet značek se zvětší, resp. zmenší o 1. Pak $\sigma(E_1, E_2)$ je supremum přes největší změny počtu značek v místě s při provádění sítě.

Graficky jsou místa s a nové hrany sítě kresleny čárkovaně. Místo s má označení „$\sigma = x$“, jestliže $\sigma({}^\bullet s, s^\bullet) = \sigma(E_1, E_2) = x$.

Poznámka: Grafickou reprezentací synchronizační vzdálenosti v C/E síti získáme P/T síť, ve které nové místo s má neomezenou kapacitu značenou symbolem ω ($\omega = \inf(\mathbb{N})$).

Na obrázku 6.4 jsou graficky vyznačeny synchronizační vzdálenosti z příkladu 6.3.

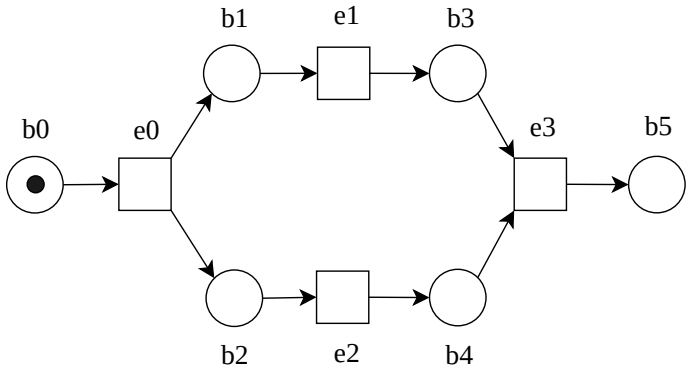


Obrázek 6.4: Grafická reprezentace synchronizační vzdálenosti

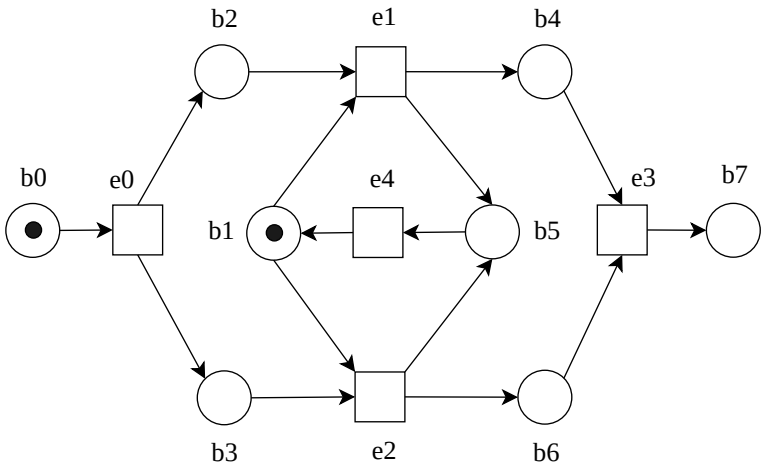
6.3 Některé speciální synchronizační vzdálenosti

Zřejmě, je-li $E_1 = E_2$, pak $\sigma(E_1, E_2) = 0$ a naopak, je-li $\sigma(E_1, E_2) = 0$, pak $E_1 = E_2$. Speciálně $e_1 = e_2 \Leftrightarrow \sigma(e_1, e_2) = 0$, události e_1 a e_2 koincidují.

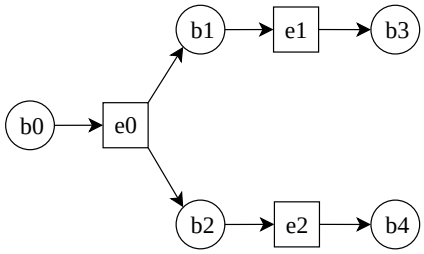
Uvažujme nyní dva C/E systémy Σ a Σ' na obrázcích 6.5 a 6.6. Události e_1 a e_2 jsou v Σ souběžné (paralelní) a tedy nezávislé. Podle definice je $\sigma_\Sigma(e_1, e_2) = 2$. V systému Σ' na obrázku 6.6 je zaveden „řídící mechanizmus“, který neumožňuje, aby e_1 a e_2 byly provedeny souběžně, musí tedy proběhnout v nějakém pořadí. V procesech p_1 (obrázek 6.8) a p_2 (obrázek 6.9) systému Σ' leží $p_i^{-1}(e_1)$ a $p_i^{-1}(e_2)$, $i = 1, 2$ v jednom řetězci. Kdežto $p^{-1}(e_1)$ a $p^{-1}(e_2)$ v procesu p systému Σ na obrázku 6.7 jsou souběžné. Koncepční rozdíl mezi systémy Σ a Σ' je vyjádřen právě synchronizační vzdáleností mezi událostmi e_1 a e_2. V C/E systému Σ' je $\sigma_{\Sigma'}(e_1, e_2) = 1$.



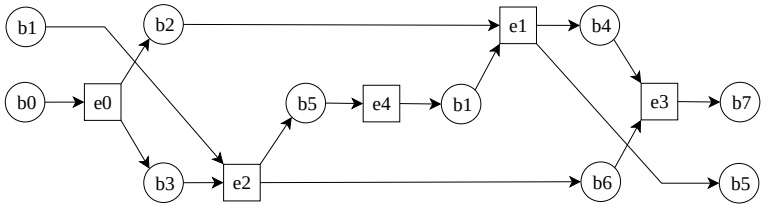
Obrázek 6.5: C/E systém Σ



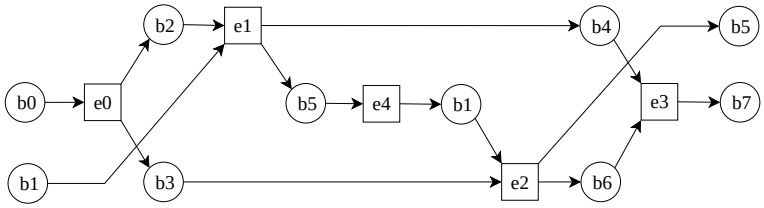
Obrázek 6.6: C/E systém Σ'



Obrázek 6.7: Proces p systému Σ z obrázku 6.5

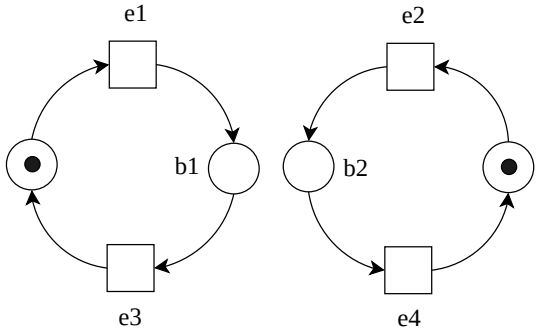


Obrázek 6.8: Proces p_1 systému Σ' z obrázku 6.6

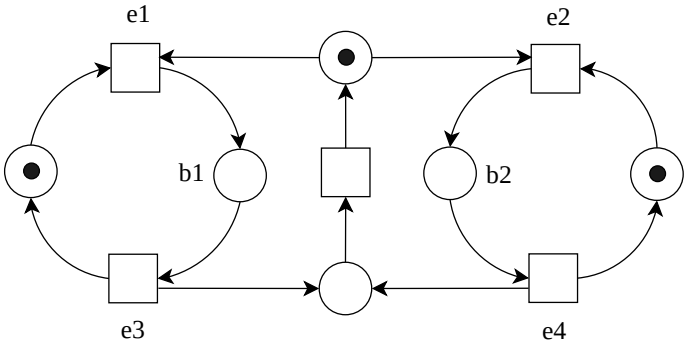


Obrázek 6.9: Proces p_2 systému Σ' z obrázku 6.6

Odpovídající dvojice událostí C/E systému Σ_1 (obrázek 6.10) a C/E systému Σ_2 (obrázek 6.11) mají shodnou synchronizační vzdálenost: $\sigma(e_1, e_2) = \sigma(e_1, e_4) = \omega$ a $\sigma(e_1, e_3) = \sigma(e_2, e_4) = 1$ v obou C/E systémech.

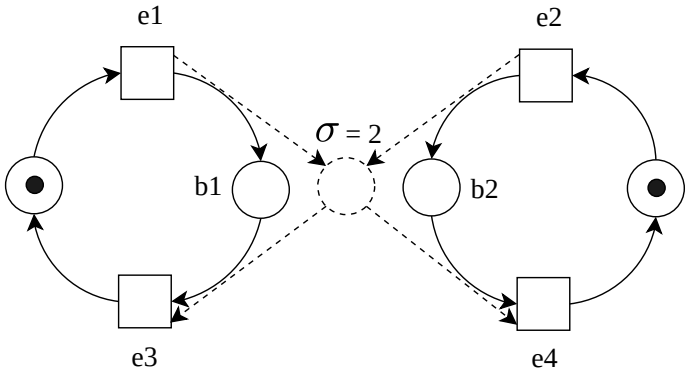


Obrázek 6.10: C/E systém Σ_1

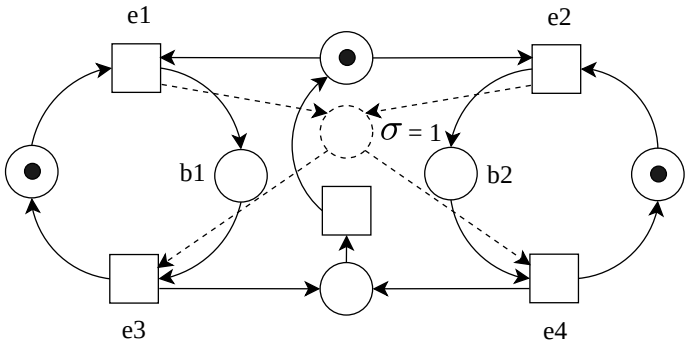


Obrázek 6.11: C/E systém Σ_2

Intuitivně však cítíme, že C/E systém Σ_2 je více synchronizován, protože v C/E systému Σ_2 nemohou být dvě události provedeny souběžně. To je vyjádřitelné synchronizační vzdáleností množin $\{e_1, e_2\}$ a $\{e_3, e_4\}$, která má hodnotu $\sigma(\{e_1, e_2\}, \{e_3, e_4\}) = 2$ v C/E systému Σ_1 (obrázek 6.12), avšak hodnotu $\sigma(\{e_1, e_2\}, \{e_3, e_4\}) = 1$ v C/E systému Σ_2 (obrázek 6.13).



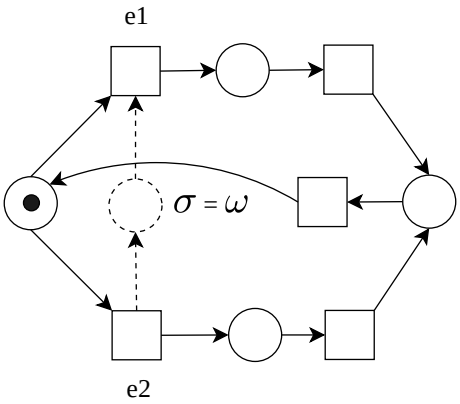
Obrázek 6.12: C/E systém Σ_1



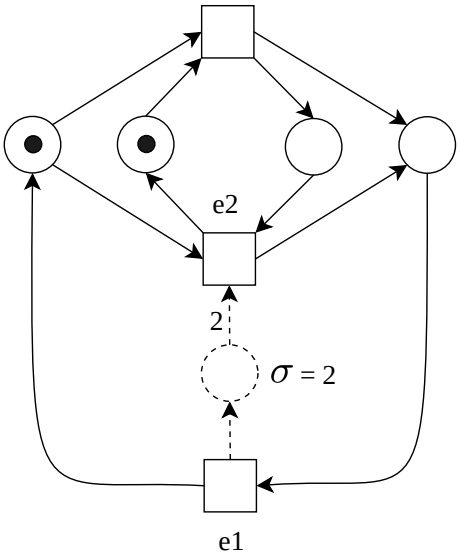
Obrázek 6.13: C/E systém Σ_2

V C/E systému na obrázku 6.14 jsou události e_1 a e_2 vzájemně neomezeně často v konfliktu, jejich synchronizační vzdálenost je tedy nekonečná: $\sigma(e_1, e_2) = \omega$. V C/E systému na obrázku 6.15 je také synchronizační vzdálenost mezi událostmi e_1 a e_2 nekonečná. Ale na rozdíl od C/E systému na obrázku 6.14 jsou však výskyty e_1 a e_2 na sobě závislé (událost e_1 se provádí dvakrát častěji než událost e_2). K vyjádření tohoto rozdílu je třeba zobecnění synchronizační vzdálenosti na *váhovanou synchronizační vzdálenost* (angl. *weighted synchronic distance*). Například v C/E systému na obrázku 6.15 uvedeme, že provedení události e_2 sníží počet značek v nově přidaném místě dvakrát. V grafické reprezentaci mají přidané hrany celočíselnou váhu.

váhovaná synchronizační vzdálenost



Obrázek 6.14: C/E systém



Obrázek 6.15: C/E systém

6.4 Některé kvantitativní vlastnosti synchronizační vzdálenosti

■ **Věta 6.3** Věta

Nechť Σ je bezkontaktní C/E systém a nechť $E_1, E_2, E_3 \subseteq E_\Sigma$. Pak

1. $\sigma(E_1, E_2) = 0 \Leftrightarrow E_1 = E_2$
2. $\sigma(E_1, E_2) = \sigma(E_2, E_1)$
3. $\sigma(E_1, E_2) \leq \sigma(E_1, E_3) + \sigma(E_3, E_2)$

□

Synchronizační vzdálenost je tedy *metrikou* na množinách událostí.

■ **Věta 6.4** Věta

Nechť Σ je bezkontaktní C/E systém a nechť $E_1, E_2 \subseteq E_\Sigma$. Pak $\sigma(E_1, E_2) = \sigma(E_1 \setminus E_2, E_2 \setminus E_1)$ □

6.5 Synchronizační vzdálenosti v sekvenčních systémech

V čistě sekvenčních systémech není synchronizační vzdálenost příliš zajímavá, protože pro každou dvojici událostí můžeme dostat pouze hodnoty $0, 1$ a ω.

■ **Definice 6.5** *Stavový stroj* DEF

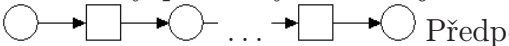
C/E systém Σ se nazývá *stavový stroj* (angl. *state machine*), jestliže

1. $\forall e \in E_\Sigma : \ |{}^\bullet e| = |e^\bullet| = 1$
2. $\forall c \in C_\Sigma : \ |c| = 1$

□

■ **Věta 6.5** Věta

Nechť Σ je stavový stroj a nechť $e_1, e_2 \in E_\Sigma$. Pak $\sigma(e_1, e_2) \in \{0, 1, \omega\}$
□

Důkaz. Každý proces systému Σ je tvořen řetězcem tvaru na obrázku  Předpokládejme, že existuje proces $p : K \to \Sigma$ se dvěma přechody $t_1, t_2 \in T_K$ takovými, že pro $i = 1, 2$, $p(t_1) = p(t_2) = e_i$ a $\forall t \in t_1^+ \cap t_2^- : p(t) \neq e_i$ □

Pak pro $p_1 = p \mid (\bullet t_1^+ \cap \bullet t_2^-)$ je $p_n = \overbrace{p_1 \circ \ldots \circ p_1}^{\texttt{n-krát}}$ také proces a $\nu(p_n, \{e_1\}, \{e_2\}) \geq n$. Potom $\sigma(e_1, e_2) = \omega$. Jinak je pro všechny procesy $p : \nu(p, e_1, e_2) \leq 1$ a tedy $\sigma(e_1, e_2) \leq 1$.

6.6 Synchronizační vzdálenosti v cyklických systémech

Definujme nyní jednodušší funkci σ', která je pro speciální případy cyklických C/E systémů ekvivalentní se synchronizační vzdáleností σ.

DEF

■ **Definice 6.6**

Nechť Σ je bezkontaktní C/E systém, $E_1, E_2 \subseteq E_\Sigma$ a nechť $p \in \pi_\Sigma$.

1. $\nu'(p, E_1, E_2) = \mid |p^{-1}(E_1)| - |p^{-1}(E_2)| \mid$
2. $\sigma'(E_1, E_2) = \sup\{\ \nu'(p, E_1, E_2) \mid\ p \in \pi_\Sigma)\ \}$

□

Věta

■ **Věta 6.6**

Pro libovolný C/E systém Σ a $E_1, E_2 \subseteq E_\Sigma$ platí $\sigma'(E_1, E_2) \leq \sigma(E_1, E_2)$
□

Například v C/E systému na obrázku 6.5 je
$\sigma'(\{e_1\}, \{e_2\}) = 1 \quad < \quad \sigma(\{e_1\}, \{e_2\}) = 2.$

Věta

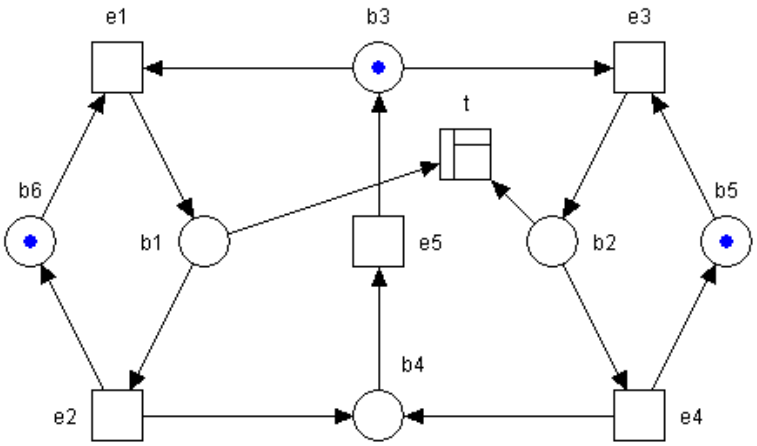
■ **Věta 6.7**

Nechť Σ je bezkontaktní a cyklický systém.
Pak pro všechny $E_1, E_2 \subseteq E_\Sigma$ je $\sigma'(E_1, E_2) = \sigma(E_1, E_2)$ □

6.7 Fakta

S využitím podmínek C/E systému je možno konstruovat formule výrokové logiky. Tyto formule budou pravdivé, či nepravdivé v závislosti na tom, ve kterém případu se C/E systém nachází. Nejvíce zajímavé jsou formule, které jsou pravdivé ve všech případech z C/E systému, protože popisují jeho invariantní vlastnosti. Nyní si ukážeme, jak lze reprezentaci a vyhodnocení těchto formulí začlenit do „síťového kalkulu“.

Uvažujme C/E systém Σ_1 z obrázku 6.10, který se skládá ze dvou sekvenčních cyklů. Přidejme navíc požadavek, aby podmínky b_1 a b_2 nikdy neplatily současně. Toho lze dosáhnout konstrukcí C/E systému Σ_2 uvedeném na obrázku 6.11. Tato nová vlastnost systému může být vyjádřena zavedením nového přechodu t takového, že ${}^\bullet t = \{b_1, b_2\}$ a $t^\bullet = \emptyset$, který není proveditelný v žádném případu C/E systému Σ_2. Výsledek vidíme na obrázku 6.16.



Obrázek 6.16: C/E systém s přidaným faktem

Nejprve budeme studovat vztahy mezi formulemi obsahujícími podmínky (například $\neg(b_1 \wedge b_2)$ v C/E systému na obrázku 6.16) a prováděním událostí. K tomu účelu uvažujme podmínku b jako prvotní (atomickou) formuli, která je pravdivá v daném případu c, právě když b patří do c. Pak můžeme konstruovat formule výrokové logiky a vyhodnocovat jejich pravdivostní hodnoty.

DEF

■ **Definice 6.7**

Nechť Σ je C/E systém.

1. Množina A_Σ formulí (výrokové logiky) nad B_Σ je nejmenší množina, pro kterou
   (a) $B_\Sigma \subseteq A_\Sigma$
   (b) $a_1, a_2 \in A_\Sigma \Rightarrow (a_1 \wedge a_2) \in A_\Sigma, (a_1 \vee a_2) \in A_\Sigma, (a_1 \rightarrow a_2) \in A_\Sigma, (\neg a_1) \in A_\Sigma$
2. V každém $c \in C_\Sigma$ přísluší každé formuli $a \in A_\Sigma$ hodnota $\hat{c}(a)$ definovaná valuací $\hat{c} : A_\Sigma \rightarrow \{0, 1\}$:
   - $b \longmapsto 1$, jestliže $b \in c$
   - $b \longmapsto 0$, jestliže $b \notin c$
   - $(a_1 \wedge a_2) \longmapsto \min(\hat{c}(a_1), \hat{c}(a_2))$
   - $(a_1 \vee a_2) \longmapsto \max(\hat{c}(a_1), \hat{c}(a_2))$
   - $(a_1 \rightarrow a_2) \longmapsto \hat{c}((\neg a_1) \vee a_2)$
   - $(\neg a_1) \longmapsto 1 - \hat{c}(a_1)$
3. Dvě formule $a_1, a_2 \in A_\Sigma$ jsou ekvivalentní v Σ, jestliže pro všechny $c \in C_\Sigma : \ \hat{c}(a_1) = \hat{c}(a_2)$

□

Nadbytečné závorky můžeme vynechávat. Hodnotu 1 interpretujeme jako „true“ a hodnotu 0 jako „false“. Formuli a nazýváme platnou v případu c, když $\hat{c}(a) = 1$. Připomeňme, že operátor $\vee$ a $\wedge$ je asociativní.

Nyní ke každé události $e \in E_\Sigma$ přiřadíme formuli $a(e)$ tak, že ve všech případech c: $a(e)$ je platná v c, právě když e není c-proveditelná.

DEF

■ **Definice 6.8**

Nechť Σ je konečný C/E systém a nechť $e \in E_\Sigma$. Nechť ${}^\bullet e = \{b_1, b_2, \ldots, b_n\}$, $e^\bullet = \{b'_1, b'_2, \ldots, b'_m\}$. Pak $a(e)$ je formule $(b_1 \wedge b_2 \wedge \ldots \wedge b_n) \rightarrow (b'_1 \vee b'_2 \vee \ldots \vee b'_m)$. Je-li ${}^\bullet e = \emptyset$, pak $a(e)$ je formule $(b'_1 \vee \ldots \vee b'_m)$, je-li $e^\bullet = \emptyset$, pak $a(e)$ je formule $\neg(b_1 \wedge b_2 \wedge \ldots \wedge b_n)$. □

Lemma

■ **Lemma 6.1**

Nechť Σ je konečný C/E systém a nechť $e \in E_\Sigma$. Pak pro každé $c \in C_\Sigma$, $a(e)$ platí v c, právě když e není c-proveditelná. □

Důkaz. $\hat{c}(a(e)) = 1 \Leftrightarrow \exists b \in {}^\bullet e$, kde $\hat{c}(b) = 0$ nebo $\exists b' \in e^\bullet : c(b') = 1 \Leftrightarrow \exists b \in {}^\bullet e$ a $b \notin c$ nebo $\exists b' \in e^\bullet$ a $b' \in c \Leftrightarrow e$ není c-proveditelná. □

Ukázali jsme, jak spojovat formule s událostmi C/E systému. Teď uvažujme, jak reprezentovat libovolnou pravdivostní formuli sestavenou z podmínek systému.

K tomu účelu obohatíme C/E systém Σ o nové přechody, které nejsou proveditelné v žádném případu ze Σ (přidáme tzv. „mrtvé“ přechody). Proto neovlivní chování systému. Pokud s každým novým přechodem t spojíme formuli $a(t)$ tak, jak jsme to prováděli pro události, potom $a(t)$ je platná v systému Σ (platí pro každý jeho případ). Takto je možné reprezentovat všechny platné formule pro Σ určitým počtem „mrtvých“ přechodů. Tyto přechody nazýváme fakta.

DEF

■ **Definice 6.9** *Platná formule, fakt*

Nechť Σ je C/E systém.

1. Formule $a \in A_\Sigma$ se nazývá *platnou* (angl. *valid*) v C/E systému Σ, jestliže $\forall c \in C_\Sigma : \ \hat{c}(a) = 1$

2. Pro $B_1, B_2 \subseteq B_\Sigma$ nechť $t = (B_1, B_2)$ je nový přechod: ${}^\bullet t = B_1$ a $t^\bullet = B_2$. Přechod t se nazývá *faktem* (angl. *fact*) systému Σ, jestliže t není proveditelný pro žádné $c \in C_\Sigma$.

□

V grafické reprezentaci je fakt t označen přechodem  , který znázorňuje velké „F“.

Pro fakt t formuli $a(t)$ definujeme stejně jako pro událost e formuli $a(e)$. Například jestliže ${}^\bullet t = \{b_1, \ldots, b_n\}, \ t^\bullet = \{b'_1, \ldots, b'm\}$, pak $a(t) = (b_1 \wedge b_2 \wedge \ldots \wedge b_n) \rightarrow (b'_1 \vee b'_2 \vee \ldots \vee b'_m)$

■ **Věta 6.8** Věta

Nechť Σ je konečný C/E systém a nechť $a \in A_\Sigma$. Formule a je platná v Σ, právě když existují fakta $t_1, t_2, \ldots, t_k$ taková, že a je logicky ekvivalentní formuli $a(t_1) \wedge a(t_2) \wedge \ldots \wedge a(t_n)$. □

Jak reprezentovat formule, které platí jen pro některé případy systému? Pro $c \in C_\Sigma$ nechť c' označuje konjunkci všech podmínek ze Σ tvořících c. Pak a platící pro případy $c_1, \ldots, c_k$ a lze jej popsat formulí $(c'_1 \wedge c'_2 \wedge \ldots \wedge c'_k) \rightarrow a$.

Pojmy k zapamatování

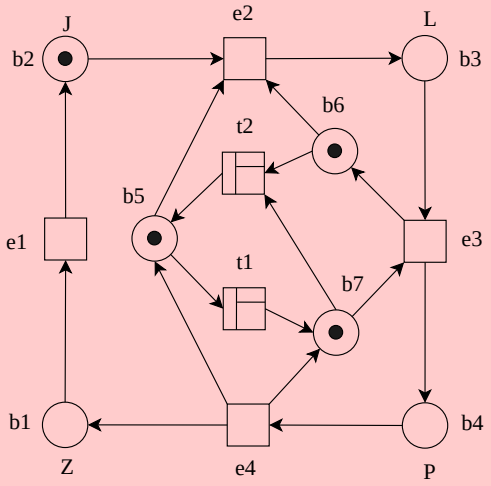
- míra, variance, synchronizační vzdálenost
- fakta

Shrnutí

V této kapitole jsme se soustředili na definici synchronizační vzdálenosti jako prostředku pro získání kvantitativní informace o dynamickém chování systému, aniž bylo třeba zavádět pojem „času“. Ukázali jsme si některé speciální synchronizační vzdálenosti a také její grafickou reprezentaci. Studovali jsme vztahy mezi podmínkami a prováděním událostí. Ukázali jsme, jak spojovat formule s událostmi systému. Možnost reprezentovat všechny platné formule v C/E systému jsme vyřešili vložením „mrtvých“ přechodů.

Příklady k procvičení

1. Definujte pojem synchronizační vzdálenosti dvou událostí C/E systému.
2. Jakým způsobem reprezentujeme události C/E systému formulemi výrokové logiky?
3. Definujte pojem fakt a uveďte jeho grafickou reprezentaci.
4. Pro C/E systém na obrázku uveďte formule výrokové logiky pro podmínky b_5, b_6, b_7 a pro fakta t_1, t_2.



Část II

P/T sítě

Kapitola 7

Definice P/T Petriho sítí

Čas potřebný ke studiu: 7 hodin

Cíle kapitoly

Cílem této kapitoly je definovat P/T Petriho sítě a jejich evoluční pravidla. Dále pak pochopit stavový prostor P/T Petriho sítí ve vztahu ke konečným automatům. Zavedeme maticovou reprezentaci Petriho sítě, která umožňuje zjednodušit formální práci s pojmy, které souvisejí se značením Petriho sítě, a které se úzce dotýkají chování modelovaného systému. Kapitola stanovuje rámce teorie, které jsou využívány i v následujících částech učebního textu.

Průvodce studiem

Na tuto kapitolu navazují ostatní kapitoly věnované P/T Petriho sítím, proto je důkladnému pochopení jednotlivých termínů, včetně vzájemných souvislostí, věnovat náležitou pozornost.

Obsah

7.1 Definice P/T Petriho sítě

Již dříve v textu jsme se setkali se symbolem ω, který vyjadřoval neomezenou kapacitu místa, nebo neomezený (nekonečný) počet značek obsažených v místě. Následující definice nám umožní s tímto symbolem korektně zacházet v aritmetických operacích a relacích.

DEF

■ **Definice 7.1** *Rozšíření* $\mathbb{N}$ *na* $\mathbb{N} \cup \{\omega\}$.

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

1. $\forall n \in \mathbb{N} : n < \omega$
2. $\forall m \in \mathbb{N} \cup \{\omega\} : m + \omega = \omega + m = \omega;\ \omega - m = \omega$
3. Nechť $A \subseteq \mathbb{N}$. Pak

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

□

Nyní již můžeme uvést definici Petriho sítě.

DEF

■ **Definice 7.2** *P/T Petriho síť.*

Šestici $N = (P, T, F, W, K, M_0)$ nazýváme *P/T Petriho síť (Place/Transition Petri Net)*, jestliže

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

Dále budeme Petriho sítí rozumět P/T Petriho síť. □

Následující definice vymezuje přesně pravidla pro provádění přechodů Petriho sítě. V některé literatuře jsou tato pravidla označována jako *evoluční pravidla* Petriho sítě.

DEF

■ **Definice 7.3** *Značení Petriho sítě.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Zobrazení $M : P \to \mathbb{N} \cup \{\omega\}$ se nazývá *značení (marking)* Petriho sítě N, jestliže $\forall p \in P : M(p) \leq K(p)$. □

Konvence

V grafické reprezentaci Petriho sítě budeme dodržovat následující konvence.

- Hranu $f \in F$ budeme popisovat ohodnocením $W(f)$ pouze v případě $W(f) \geq 1$
- Kapacitu místa $p \in P$ budeme popisovat ve tvaru $\kappa = K(p)$.
- Neoznačená kapacita místa p bude znamenat $\kappa = \omega$.
- Hodnotu značení místa p vykreslujeme počtem černých teček uvnitř místa p, případně číslem $M(p)$, je-li tato hodnota obtížně zobrazitelná tečkami, nebo symbolem ω.

Je-li $P = \{p_1, p_2, \ldots, p_n\}$ množina míst Petriho sítě N a M nějaké značení téže sítě, pak toto značení budeme často popisovat vektorem $(M(p_1), M(p_2), \ldots, M(p_n))$. □

DEF

■ **Definice 7.4** *Proveditelnost přechodu.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Přechod $t \in T$ je *proveditelný (enabled)* při značení M, stručněji *M-proveditelný (M-enabled)*, jestliže

$$\forall p \in {}^\bullet t : M(p) \geq W(p, t)$$

$$\forall p \in t^\bullet : M(p) \leq K(p) - W(t, p)$$

□

DEF

■ **Definice 7.5** *Provedení přechodu, následné značení.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a M její značení. Je-li přechod $t \in T$ proveditelný při značení M, pak jeho *provedením* získáme následné značení M' ke značení M, které je definováno takto:

$$\forall p \in P : M'(p) = \begin{cases} M(p) - W(p,t), & \text{jestliže } p \in {}^\bullet t \setminus t^\bullet \\ M(p) + W(t,p), & \text{jestliže } p \in t^\bullet \setminus {}^\bullet t \\ M(p) - W(p,t) + W(t,p), & \text{jestliže } p \in {}^\bullet t \cap t^\bullet \\ M(p), & \text{jinak} \end{cases}$$

Provedení přechodu (transition firing) t ze značení M do značení M' zapisujeme symbolicky $M\ [t\rangle\ M'$. □

DEF

■ **Definice 7.6** *Množina dosažitelných značení.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a M její značení. Symbolem $[M\rangle$ označme nejmenší množinu různých značení sítě N takovou, že platí:

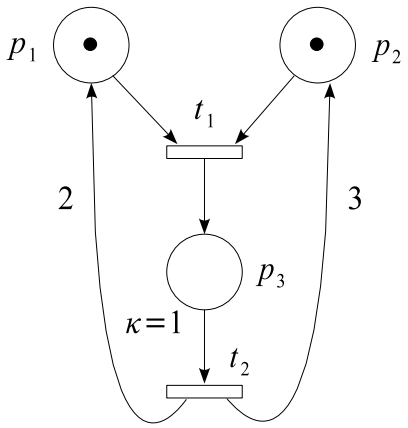
1. $M \in [M\rangle$
2. je-li $M_1 \in [M\rangle$ a pro nějaké $t \in T$ platí $M_1\ [t\rangle\ M_2$, pak $M_2 \in [M\rangle$

Množina $[M\rangle$ se označuje jako *množina dosažitelných značení (reachability set)* ze značení M. Množina $[M_0\rangle$ dosažitelných značení z počátečního značení se označuje jako *množina dosažitelných značení (reachability set)* sítě N. □

x+y

■ **Příklad 7.1**

Na obrázku 7.1 je znázorněn graf Petriho sítě s počátečním značením.



Obrázek 7.1: Graf sítě N

Pro tuto Petriho síť platí:

- $K(p_1) = K(p_2) = \omega$, $K(p_3) = 1$
- $W(p_1, t_1) = W(p_2, t_1) = W(t_1, p_3) = W(p_3, t_2) = 1$, $W(t_2, p_1) = 2$, $W(t_2, p_2) = 3$

- $M_0(p_1) = M_0(p_2) = 1,\ M_0(p_3) = 0$

V počátečním značení M_0 je proveditelný přechod t_1. Opakovaným prováděním přechodů t_1 a t_2 v posloupnosti $(t_1 t_2)^+$ získáváme posloupnost značení:

$$(1,1,0)\ [t_1\rangle\ (0,0,1)\ [t_2\rangle\ (2,3,0)\ [t_1\rangle\ (1,2,1)\ [t_2\rangle\ (3,5,0)\ [t_1\rangle\ (2,4,1)\ldots$$

Množina dosažitelných značení této sítě je

$$[M_0\rangle\ = \{(i, 2i-1, 0) | i \in \mathbb{N} \setminus \{0\}\} \cup \{(j, 2j, 1) | j \in \mathbb{N}\}$$

□

7.2 Komplementace Petriho sítě

DEF

■ **Definice 7.7** *Bezkontaktní Petriho síť.*

Petriho síť $N = (P, T, F, W, K, M_0)$ se nazývá *bezkontaktní (contact-free)*, jestliže $\forall M \in [M_0\rangle$ a $\forall t \in T$ platí následující tvrzení:

$$\forall p \in {}^\bullet t : M(p) \geq W(p,t) \Longrightarrow \forall p \in t^\bullet : M(p) \leq K(p) - W(t,p)$$

□

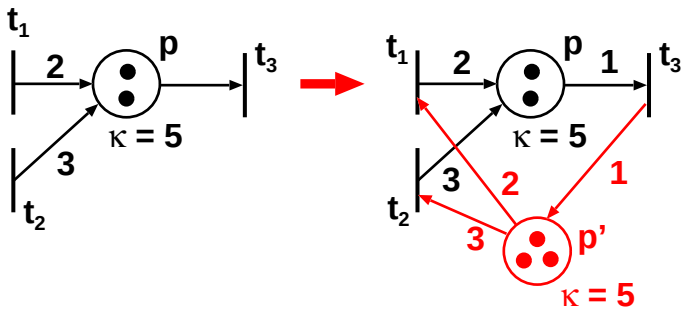
Vyjádřeno slovně, pokud je ve všech místech vstupní množiny přechodu t dostatečně vysoký počet značek (tedy větší nebo roven váze příslušné hrany vstupující do t), pak ve všech místech výstupní množiny téhož přechodu je dostatečně nízký počet značek, aby byl přechod t proveditelný.

Srovnáme-li definici proveditelnosti přechodu (definice 7.4) s výrazem na levé a pravé straně implikace v předchozí definici bezkontaktní Petriho sítě (definice 7.7), pak zjistíme jejich úzkou souvislost. Definice proveditelnosti přechodu definuje 2 podmínky, které musí být splněny, aby byl přechod proveditelný. Definice bezkontaktní sítě mezi tytéž podmínky klade implikaci platnou pro všechny přechody ve všech dosažitelných stavech sítě.

U bezkontaktní sítě nemůže nastat situace, kdy by provedením přechodu byla překročena kapacita místa, kdy by tedy došlo ke kontaktu značek, které mají být přidány, se značkami, jež měly být odebrány.

Nyní ukážeme, že každá Petriho síť může být rozšířena o určitá místa nazvaná *komplementární* tak, že výsledná Petriho síť má stejné chování, avšak je bezkontaktní. Této transformaci říkáme *komplementace* Petriho sítě.

Příklad komplementace Petriho sítě je na obrázku 7.2.



Obrázek 7.2: Komplementace Petriho sítě

Je-li dána Petriho síť $N = (P, T, F, W, K, M_0)$, pak komplementární Petriho síť $N' = (P', T', F', W', K', M'_0)$ k síti N získáme přidáním nových míst a hran takto:

1. Pro každé místo $p \in P$ s konečnou kapacitou, tj. $K(p) \neq \omega$, vytvoříme nové místo p' a pokračujeme následujícími body.

2. Kapacita místa $K(p') = K(p)$.

3. Počáteční značení $M'_0(p') = K(p) - M_0(p)$.

4. Pro všechny hrany $\langle t, p\rangle$ vcházející do místa p vytvoříme nové hrany $\langle p', t\rangle$ tak, že $W'(p', t) = W(t, p)$. Obdobně, pro všechny hrany $\langle p, t\rangle$ vycházející z místa p vytvoříme nové hrany $\langle t, p'\rangle$ tak, že $W'(t, p') = W(p, t)$.

Výsledná síť N' je bezkontaktní, protože pro všechna dosažitelná značení M platí $M(p) + M(p') = K(p)$. Jestliže provedeme restrikci (ve smyslu restrikce zobrazení) značení M' sítě N' na místa P sítě N, pak tato značení korespondují se značeními M sítě N. Tato korespondence je jednoznačná. Skutečně, jsou-li dána odpovídající značení M sítě N a M' sítě N', pak každý přechod t je M-proveditelný v síti N právě tehdy, je-li M'-proveditelný v síti N'.

Z vlastností značení sítě N' plyne, že každou konečnou kapacitu $\kappa = K(p)$ v síti N' můžeme nahradit neomezenou kapacitou ω, aniž bychom změnili chování sítě N'. Tohoto faktu spolu s možností převodu libovolné Petriho sítě na bezkontaktní Petriho síť se často využívá. Některé definice Petriho sítě z tohoto důvodu vůbec nezavádějí složku K (kapacity míst). Příkladem je definice popsaná v následující podkapitole.

7.3 Alternativní definice Petriho sítě

Zvláště v americké odborné literatuře se používá definice Petriho sítě, která je založena na ryzím chápání grafu Petriho sítě jako bipartitního orientovaného multigrafu. Výhodou této definice je odstranění pomocného pojmu síť (definice 3.1), avšak na druhé straně vzniká nutnost pracovat se zobecněním pojmu množina na *multimnožinu (bag)*.

Multimnožina se od množiny liší tím, že připouští vícenásobné výskyty téhož prvku. Typickým příkladem multimnožiny je rozklad přirozeného čísla na prvočinitele, $\underline{180} = \{2, 2, 3, 3, 5\}$ je multimnožina reprezentující číslo 180. Pro používání multimnožiny zavedeme tyto konvence:

1. Nechť B je multimnožina a b její prvek. Symbolem $\#(b, B)$ budeme značit počet výskytů prvku b v B. Např. $\#(3, \underline{180}) = 2$. Zřejmě platí následující implikace: $\#(b, B) = 0 \Rightarrow b \notin B$

2. Je-li A množina, pak symbolem A^∞ značíme systém multimnožin vytvořených z prvků množiny A.

Nyní již můžeme přistoupit k alternativní definici Petriho sítě.

DEF

▪ **Definice 7.8** *Alternativní definice P/T Petriho sítě.*

P/T Petriho síť je pětice $N = (P, T, I, O, M_0)$, kde:

1. P je neprázdná konečná *množina míst*, T je neprázdná konečná *množina přechodů*, $P \cap T = \emptyset$

2. zobrazení $I : T \to P^\infty$ je *vstupní funkce*, přiřazující každému přechodu multimnožinu vstupních míst

3. zobrazení $O : T \to P^\infty$ je *výstupní funkce*, přiřazující každému přechodu multimnožinu výstupních míst

4. zobrazení $M_0 : P \to \mathbb{N} \cup \{\omega\}$ je *počáteční značení* Petriho sítě N

□

DEF

▪ **Definice 7.9** *Alternativní definice grafu P/T Petriho sítě.*

Grafem Petriho sítě N je orientovaný bipartitní multigraf

$$G_N = (H, P \cup T, \sigma),\ \sigma : H \to (P \cup T) \times (P \cup T)$$

kde množina hran H a incidenční relace σ je definována takto:

$$\forall h \in H : \sigma(h) = \begin{cases} \langle p, t\rangle \Leftrightarrow p \in I(t) \\ \langle t, p\rangle \Leftrightarrow p \in O(t) \end{cases} \quad \text{pro jisté } t \in T \text{ a } p \in P$$

a dále platí

$$\forall p \in P, t \in T : |\sigma^{-1}(\langle p, t\rangle)| = \#(p, I(t)) \quad \text{a} \quad |\sigma^{-1}(\langle t, p\rangle)| = \#(p, O(t))$$

□

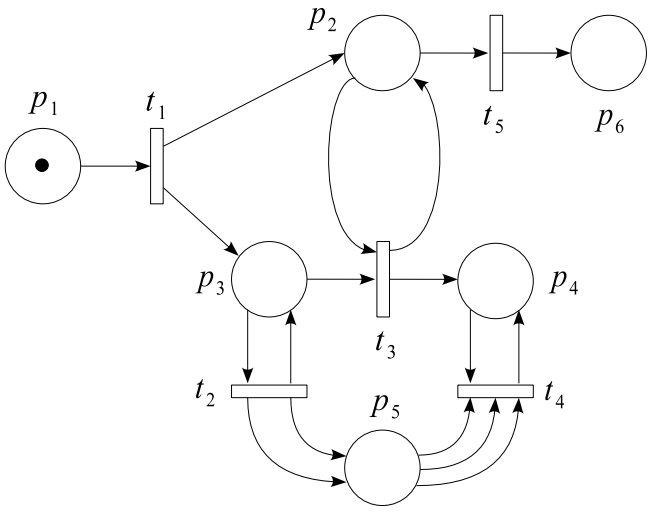
x+y

■ **Příklad 7.2**

Uvažujme definici Petriho sítě $N = (P, T, I, O, M_0)$, kde:

$$\begin{aligned} P &= \{p_1, p_2, p_3, p_4, p_5, p_6\} \\ T &= \{t_1, t_2, t_3, t_4, t_5\} \\ I &= \{t_1 \mapsto \{p_1\}, t_2 \mapsto \{p_3\}, t_3 \mapsto \{p_2, p_3\}, t_4 \mapsto \{p_4, p_5, p_5, p_5\}, t_5 \mapsto \{p_2\}\} \\ O &= \{t_1 \mapsto \{p_2, p_3\}, t_2 \mapsto \{p_3, p_5, p_5\}, t_3 \mapsto \{p_2, p_4\}, t_4 \mapsto \{p_4\}, t_5 \mapsto \{p_6\}\} \\ M_0 &= \{p_1 \mapsto 1, p_2 \mapsto 0, p_3 \mapsto 0, p_4 \mapsto 0, p_5 \mapsto 0, p_6 \mapsto 0\} \end{aligned}$$

Graf této sítě je na obrázku 7.3. □



Obrázek 7.3: Graf Petriho sítě z příkladu 7.2 ve tvaru orientovaného multigrafu

Realizační pravidla pro alternativně definovanou Petriho síť mají jednoduchý tvar,

DEF

■ **Definice 7.10** *Proveditelnost a provedení přechodu v alternativní definici.*

Přechod $t \in T$ je proveditelný při značení M, jestliže

$$\forall p \in P : M(p) \geq \#(p, I(t))$$

Je-li t proveditelný při značení M, pak $M\ [t\rangle\ M'$, kde značení M' je popsané takto:

$$\forall p \in P : M'(p) = M(p) - \#(p, I(t)) + \#(p, O(t))$$

□

x+y

■ **Příklad 7.3**

V Petriho síti na obrázku 7.3 je při počátečním značení proveditelný pouze přechod t_1. V této síti můžeme získat např. tyto posloupnosti realizací přechodů:

$(1,0,0,0,0,0)\ [t_1\rangle\ (0,1,1,0,0,0)\ [t_3\rangle\ (0,1,0,1,0,0)\ [t_5\rangle\ (0,0,0,1,0,1)$

nebo

$(1,0,0,0,0,0)\ [t_1\rangle\ (0,1,1,0,0,0)\ [t_2\rangle\ (0,1,1,0,2,0)\ [t_2\rangle\ (0,1,1,0,4,0)\ [t_3\rangle$

$(0,1,0,1,4,0)\ [t_4\rangle\ (0,1,0,1,1,0)\ [t_5\rangle\ (0,0,0,1,1,1)$

□

7.4 Stavový prostor a přechodová funkce Petriho sítě

V úvodu této kapitoly jsme konstatovali příbuznost Petriho sítí a konečných automatů. Ukažme si nyní tuto příbuznost v termínech, které jsou vlastní konečným automatům.

Petriho síť je automat, jehož množina stavů, *stavový prostor sítě*, je tvořena množinou všech značení, která jsou dosažitelná z počátečního značení, tj. množinou $[M_0\rangle$. Na této množině může být definována přechodová funkce určující na základě přítomného stavu a proveditelného přechodu příští stav sítě.

DEF

■ **Definice 7.11** *Přechodová funkce, výpočetní posloupnost.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a $[M_0\rangle$ její množina dosažitelných značení. *Přechodovou funkcí* Petriho sítě N nazveme (parciální) funkci δ:

$$\delta : [M_0\rangle \times T \rightarrow [M_0\rangle$$

pro kterou

$$\forall t \in T \wedge \forall M, M' \in [M_0\rangle : \delta(M,t) = M' \Leftrightarrow M\ [t\rangle\ M'$$

Přechodová funkce δ může být zobecněna na posloupnost přechodů:

$$\delta : [M_0\rangle \times T^* \rightarrow [M_0\rangle$$

takto:

$$\begin{aligned} \delta(M, t\tau) &= \delta(\delta(M,t),\tau), \text{ kde } \tau \in T^* \\ \delta(M, e) &= M, \text{ kde } e \text{ označuje prázdnou sekvenci přechodů} \end{aligned}$$

Posloupnost $\tau \in T^*$ nazveme *výpočetní posloupností* Petriho sítě N, je-li pro ni definována přechodová funkce $\delta(M_0, \tau)$. Množina všech výpočetních posloupností charakterizuje chování Petriho sítě. □

x+y

■ **Příklad 7.4**

Na obrázku 7.4 je zobrazena Petriho síť N a její přechodová funkce δ ve tvaru diagramu přechodů.

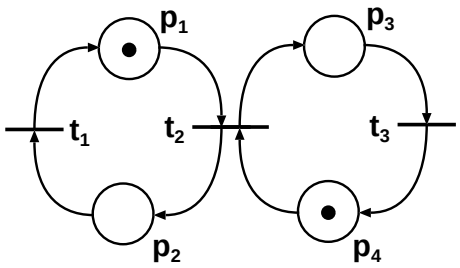
Množina výpočetních posloupností Petriho sítě N může být charakterizována regulárním výrazem

$$(t_2(e + t_3t_1(t_2t_3t_1)^*)t_1t_3$$

Každý neprázdný prefix řetězce specifikovaného tímto výrazem tvoří výpočetní posloupnost. □

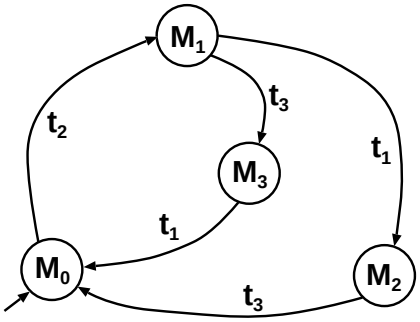
Dosud jsme nediskutovali jeden z důležitých rysů Petriho sítě, její *nedeterminismus*. Objevuje se při provádění přechodů Petriho sítě tehdy, jestliže pro jisté značení $M \in [M_0\rangle$ existují alespoň dva různé přechody $t_1, t_2 \in T$, které jsou M-proveditelné. Pak mohou nastat dvě odlišné situace

1. Existují výpočetní posloupnosti $\alpha t_1 t_2 \beta$ a $\alpha t_2 t_1 \beta$ pro jisté $\alpha, \beta \in T^*$. V tomto případě přechody t_1 a t_2 modelují dvě vzájemně nezávislé události (operace) a mohou být provedeny v libovolném pořadí.



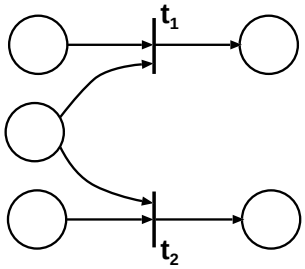
$[M_0\rangle = \{M_0, M_1, M_2, M_3\}$, kde

$$\begin{aligned} M_0 &= (1,0,0,1) \\ M_1 &= (0,1,1,0) \\ M_2 &= (1,0,1,0) \\ M_3 &= (0,1,0,1) \end{aligned}$$



Obrázek 7.4: Petriho síť a její přechodová funkce

2. Provedení přechodu t_1, resp. t_2 znemožní následné provedení přechodu t_2, resp. t_1. Přechody t_1 a t_2 se v takovém případě nazývají *konfliktní přechody*. Konfliktní přechody ilustruje obrázek 7.5.



Obrázek 7.5: Konfliktní přechody Petriho sítě

Petriho síť, jako každý matematický model systému, je popisem, který abstrahuje od určitých detailů reálného systému; je *abstraktním modelem*. Z tohoto hlediska je případ (1) abstraktním vyjádřením *paralelismu* událostí (operací), kdežto případ (2) je vyjádřením *nedeterminismu* pořadí, v jakém modelované události nastávají nebo v jakém jsou modelované operace prováděny.

Ne vždy je možné (a žádoucí) nedeterminismus Petriho sítě, vznikající v důsledku konfliktních přechodů, odstranit. Nedeterminismus Petriho sítě, podobně jako např. v případě zásobníkových automatů, obecně zvětšuje modelovací schopnost Petriho sítí. Při aplikacích Petriho sítí při modelování a simulaci dynamických diskrétních systémů se setkáváme často s dodatečným prostředkem pro snížení abstrakce popisu, kterým je *interpretace Petriho sítě*. Interpretace Petriho sítě, neformálně vyjádřeno, dodává každému přechodu sítě konkretizovanou událost či operaci a místům sítě jisté podmínky (predikáty) a množiny proměnných, jejichž hodnoty tvoří stavový prostor interpretované Petriho sítě. U interpretovaných Petriho sítí je pak možné nedeterminismus konfliktních přechodů řešit v rámci interpretace sítě.

V dalším studiu Petriho sítí zůstaneme na úrovni Petriho sítí bez interpretace, nazývaných, při zdůraznění, také *neinterpretované Petriho sítě*.

7.5 Lineární algebraická reprezentace Petriho sítě

Maticová reprezentace Petriho sítě umožňuje zjednodušit formální práci s pojmy, které souvisejí se značením Petriho sítě, a které se úzce dotýkají chování modelovaného systému. Budeme pracovat s celočíselnými vektory a maticemi, jejímiž indexy mohou být prvky libovolných konečných množin (na rozdíl od obvyklých indexů z množiny přirozených čísel).

DEF

■ **Definice 7.12** *Maticová reprezentace Petriho sítě*

Nechť $N = (P, T, F, K, W, M_0)$ je Petriho síť.

1. Pro přechod $t \in T$ nechť je definován vektor $\underline{t} : P \to \mathbb{Z}$ takto:

$$\underline{t}(p) = \begin{cases} W(t,p), & \text{jestliže } p \in t^\bullet \setminus {}^\bullet t \\ -W(p,t), & \text{jestliže } p \in {}^\bullet t \setminus t^\bullet \\ W(t,p) - W(p,t), & \text{jestliže } p \in {}^\bullet t \cap t^\bullet \\ 0, & \text{jinak} \end{cases}$$

2. Matice Petriho sítě $\underline{N}$ je definována jako matice $\underline{N} : P \times T \to \mathbb{Z}$, kde $\underline{N}(p,t) = \underline{t}(p)$.

□

Na obrázku 7.6 je uveden graf Petriho sítě, příslušná matice Petriho sítě a počáteční značení. Je zřejmé, že každé značení Petriho sítě lze reprezentovat vektorem $M : P \to \mathbb{N}$ stejně jako složku $K : P \to \mathbb{N} \cup \{\omega\}$ popisující kapacity míst.

$\underline{N} =$

	t_1	t_2	t_3	t_4	M_0
p_1	1	−1	0	0	1
p_2	−1	1	0	0	0
p_3	0	1	0	0	0
p_4	0	3	−1	0	0
p_5	0	0	−1	1	1
p_6	0	0	1	−1	1

Obrázek 7.6: Matice Petriho sítě

Poznamenejme, že Petriho síť na obrázku 7.6 je modifikací modelu producent – konzument z obrázku 3.8, v níž jsou dva konzumenti reprezentováni pouze značkami, nikoliv samostatnými podsítěmi. Cenou za méně rozsáhlý model je nerozlišitelnost individuálních konzumentů (nelze určit, zda provedením přechodu t_3 je aktivován konzument 1 nebo konzument 2).

Matice Petriho sítě $\underline{N}$ je jednoznačnou reprezentací Petriho sítě N pouze v případě, že N je čistá Petriho síť (bez vlastních cyklů). V tom případě z ní mohou být snadno odvozeny složky P, T, F i W. Je-li navíc N bezkontaktní Petriho síť, pak chování N je plně určeno maticí $\underline{N}$ a vektorem M_0. Následující věta dává stručná pravidla provádění přechodů v případě, že pracujeme s maticí Petriho sítě.

Věta

■ **Věta 7.1**

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a $M, M' : P \to \mathbb{N} \cup \{\omega\}$ jsou dvě značení této sítě. Pak pro každý přechod $t \in T$ platí:

1. Jestliže je přechod t M-proveditelný, pak

$$M\ [t\rangle\ M' \Leftrightarrow M + \underline{t} = M'$$

2. Je-li N navíc čistá síť, pak

$$t \text{ je } M\text{-proveditelný} \Leftrightarrow \mathbf{0} \leq M + \underline{t} \leq K$$

$$N \text{ je bezkontaktní } \Leftrightarrow (\forall M \in [M_0\rangle : \mathbf{0} \leq M + \underline{t} \Rightarrow M + \underline{t} \leq K)$$

□

Skutečně, v síti na na obrázku 7.6 např. platí:

$$M_0\,[t_2\rangle\,M_1 \Leftrightarrow M_1 = M_0 + \underline{t_2} = (1,0,0,0,1,1)+(-1,1,1,3,0,0) = (0,1,1,3,1,1)$$

nebo

$$M_1\,[t_3\rangle\,M_2 \Leftrightarrow M_2 = M_1 + \underline{t_3} = (0,1,1,3,1,1)+(0,0,0,-1,-1,1) = (0,1,1,2,0,0)$$

Petriho síť s nekonečnými kapacitami míst splňuje vlastnost monotónnosti značení:

■ **Lemma 7.1** *Monotónnost značení.* Lemma

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť, pro kterou $K(p) = \omega$ pro všechna $p \in P$. Nechť dále M, M_1, M_2 jsou libovolná značení této sítě, $M, M_1, M_2 : P \to \mathbb{N} \cup \{\omega\}$.

1. $M_1\,[t\rangle\,M \Rightarrow (M_1 + M_2)\,[t\rangle\,(M + M_2)$
2. $M \in [M_1\rangle \Rightarrow (M + M_2) \in [M_1 + M_2\rangle$

Důkaz. Tvrzení (1) je zřejmé, tvrzení (2) je důsledkem tvrzení (1). □ □

V některých případech, např. v programových systémech pro práci s Petriho sítěmi, se setkáváme s takovou maticovou reprezentací Petriho sítě, která je jednoznačná i pro sítě, jež nejsou čisté.

DEF

■ **Definice 7.13** *Toková matice.*

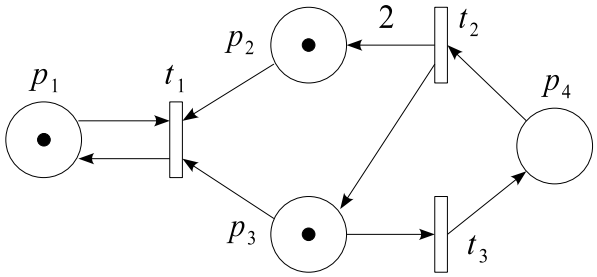
Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. *Tokovou maticí* (flow matrix) Petriho sítě N nazýváme matici $F : P \times T \to \mathbb{N} \times \mathbb{N}$, $F(p,t) = \langle w(p,t), w(t,p)\rangle$, kde

$$w(x,y) = \begin{cases} W(x,y), & \text{je-li } \langle x,y\rangle \in F \\ 0, & \text{jinak} \end{cases}$$

Toková matice je maticovou reprezentací vážené tokové relace F. Vztah mezi maticí $\underline{N}$ a $\underline{F}$ je zřejmý:

$$\forall p \in P, \forall t \in T : \underline{N}(p,t) = w(t,p) - w(p,t)$$

Matice $\underline{N}$ je, pro odlišení od matice $\underline{F}$, nazývaná někdy *maticí změn* (change matrix) Petriho sítě N. □



Obrázek 7.7: Petriho síť a její matice

x+y

■ **Příklad 7.5**

Uvažujme Petriho síť, která obsahuje cyklus:

Toková matice $\underline{F}$ a matice změn $\underline{N}$ pro tuto síť mají tvar:

$\underline{F}=$

	t_1	t_2	t_3
p_1	$\langle 1,1\rangle$	$\langle 0,0\rangle$	$\langle 0,0\rangle$
p_2	$\langle 1,0\rangle$	$\langle 0,2\rangle$	$\langle 0,0\rangle$
p_3	$\langle 1,0\rangle$	$\langle 0,1\rangle$	$\langle 1,0\rangle$
p_4	$\langle 0,0\rangle$	$\langle 1,0\rangle$	$\langle 0,1\rangle$

$\underline{N}=$

	t_1	t_2	t_3
p_1	0	0	0
p_2	-1	2	0
p_3	-1	1	-1
p_4	0	-1	1

□

Můžeme se rovněž setkat i s případem, kde je matice $\underline{F}$ reprezentována dvěma celočíselnými maticemi $\underline{F}^-$ a $\underline{F}^+$:

$\underline{F}^-=$

	t_1	t_2	t_3
p_1	1	0	0
p_2	1	0	0
p_3	1	0	1
p_4	0	1	0

$\underline{F}^+=$

	t_1	t_2	t_3
p_1	1	0	0
p_2	0	2	0
p_3	0	1	0
p_4	0	0	1

Pojmy k zapamatování

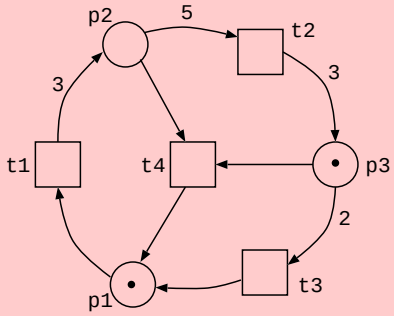
- P/T Petriho síť, provedení přechodu, následné značení
- bezkontaktní Petriho síť, komplementace Petriho sítě
- přechodová funkce, výpočetní posloupnost
- toková matice, matice změn

Shrnutí

Tato kapitola si kladla za cíl uvést čtenáře do základů P/T Petriho sítí. Definovali jsme P/T Petriho sítě a jejich evoluční pravidla. Dále bezkontaktní sítě, kde nemůže nastat situace, kdy by provedením přechodu byla překročena kapacita místa. Ukázali jsme si alternativní definici Petriho sítě, která odstraňuje nutnost definovat pojem síť, avšak na druhé straně vzniká nutnost pracovat s multimnožinou. Zavedli jsme maticovou reprezentaci Petriho sítě, která umožňuje zjednodušit formální práci s pojmy, které souvisejí se značením Petriho sítě.

Příklady k procvičení

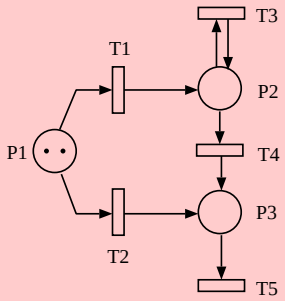
1. Zapište složky P/T sítě z obrázku ve shodě s matematickou definicí P/T sítí.



2. Uvažujte Petriho síť na obrázku a výpočtovou posloupnost $t_1 t_2 t_3 t_4 t_5$.

   (a) Zapište odpovídající posloupnost značení.

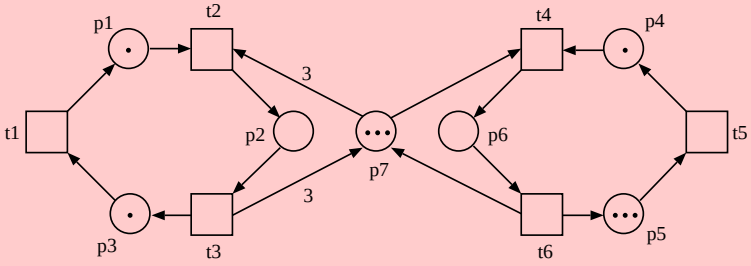
   (b) Jaká výpočtová posloupnost odpovídá posloupnosti značení: $(1,0,0),(0,0,1),(0,0,0)$?



3. Uvažujte Petriho síť na obrázku.

   (a) Určete minimální kapacity míst, tak aby neovlivnily chování sítě.

   (b) Reprezentujte tuto síť tokovou maticí a maticí změn.



Kapitola 8

Analýza P/T Petriho sítí

Čas potřebný ke studiu: 8 hodin

Cíle kapitoly

Jednou z hlavních předností P/T Petriho sítí, jakožto modelovacího nástroje, je poměrně vyvážený vztah mezi jejich modelovací a rozhodovací mocností (viz také sekce 11.1)). Pomocí P/T Petriho sítí lze tedy modelovat poměrně širokou třídu problémů, přesto však lze modely reprezentované P/T sítěmi poměrně snadno analyzovat, tedy určovat jejich vlastnosti.

Stanovování vlastností modelu na základě jeho *analýzy* dává globální pohled na model, tj. informaci o splnění dané vlastnosti ve všech dosažitelných stavech modelu.

Naproti tomu *simulace modelu*, která není obsahem této kapitoly, dává možnost sledovat určitou vlastnost pouze ve stavech, kterými model projde při použití daného simulačního (testovacího) scénáře. Je pochopitelné, že při rozsáhlém (případně nekonečném) stavovém prostoru nelze simulovat všechny možné scénáře, proto se vlastnosti modelů v takových případech odvozují právě pomocí analytických metod.

Tato kapitola studuje jednak vlastnosti, které lze u jednotlivých modelů specifikovaných pomocí P/T Petriho sítí stanovovat, a jednak metody, pomocí nichž lze tyto vlastnosti z daného modelu odvodit. Pro studium některých vlastností je použit koncept tzv. *stromu dosažitelných značení*, kterému je věnována samostatná podkapitola.

Průvodce studiem

Kapitola úzce navazuje na základní definice obsažené v kapitole 7.

Obsah

Nyní se budeme zabývat metodami, jež umožňují získat důležité informace o vlastnostech dané sítě a mohou být použity k ověření určitých žádoucích (či nežádoucích) rysů modelovaného systému. Nejprve však vymezíme základní problémy analýzy Petriho sítí.

8.1 Základní problémy analýzy

8.1.1 Bezpečnost, omezenost

DEF

■ **Definice 8.1** *Bezpečné místo, bezpečná síť.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Místo $p \in P$ se nazývá *bezpečné* (angl. *safe*), jestliže platí

$$\forall M \in [M_0\rangle : M(p) \leq 1$$

Je-li každé místo Petriho sítě N bezpečné, pak tuto síť označujeme jako *bezpečná Petriho síť* (angl. *safe Petri net*). □

Bezpečnost (angl. *safeness*) sítě tedy zaručuje, že počet značek žádného místa v libovolném stavu Petriho sítě nepřevýší hodnotu 1. Tato vlastnost je důležitá v mnoha aplikacích Petriho sítí, např. při modelování logických obvodů, kdy místa jsou realizována dvoustavovými klopnými obvody, nebo v případech, kdy všechna místa sítě modelují logické podmínky (C/E Petriho sítě).

Následující podmínka *není* postačující podmínkou pro to, aby Petriho síť byla bezpečná:

1. $M : P \to \{0, 1\}$, tedy počáteční značení je bezpečné
2. $\forall f \in F : W(f) \leq 1$, tedy váhy všech hran nejsou větší než 1

Petriho síť, která však splňuje tyto podmínky a která není bezpečná, může být doplněna o stanovení kapacity všech míst na 1, tedy $\forall p \in P : K(p) = 1$, čímž bude bezpečnost sítě explicitně zaručena. Taková síť může být převedena na komplementární bezpečnou síť postupem popsaným v podkapitole 7.2. Na obrázku 8.1 je uvedena Petriho síť, která není bezpečná a jí odpovídající komplementární bezpečná síť.

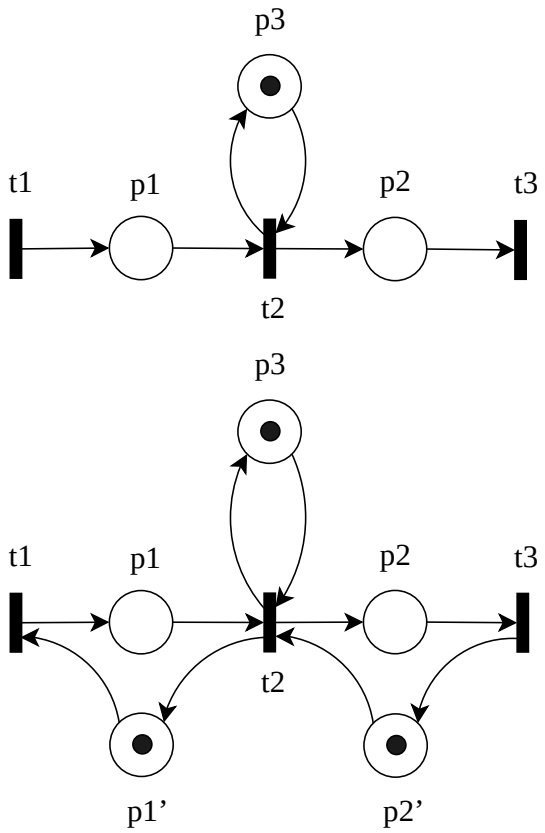
Bezpečnost sítě je speciálním případem obecnější vlastnosti, nazývané *omezenost* (angl. *boundedness*) Petriho sítě.

DEF

■ **Definice 8.2** *Omezené místo, omezená síť.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Místo $p \in P$ se nazývá *k-omezené* (angl. *k-bounded*), jestliže platí

$$\exists k \in \mathbb{N} : \forall M \in [M_0\rangle : M(p) \leq k$$



Obrázek 8.1: Petriho síť, která není bezpečná, a jí odpovídající bezpečná síť

Místo $p \in P$ se nazývá *omezené* (angl. *bounded*), je-li k-omezené pro určité $k \in \mathbb{N}$. Je-li každé místo Petriho sítě N omezené, pak tuto síť označujeme jako *omezená Petriho síť* (angl. *bounded Petri net*). □

Omezená Petriho síť je vyžadována v případech, kdy modelujeme konečné zdroje či prostředky systému, např. konečnou kapacitu vyrovnávací paměti, konečnou délku fronty apod. Každá bezpečná Petriho síť je 1-omezená Petriho síť.

Nutnou podmínkou pro k-omezenost Petriho sítě je *konečnost počátečního značení*:

$$M_0 : P \to \{0, 1, \ldots, k\}$$

Stejně jako v případě bezpečných sítí může být každá neomezená Petriho síť převedena na k-omezenou Petriho síť metodou komplementace míst.

8.1.2 Konzervativnost

Petriho sítě se často používají k modelování různých strategií přidělování zdrojů v modelovaném systému. Např. pomocí Petriho sítě modelujeme požadavky na přidělení zdroje (procesoru nebo vstup/výstupního zařízení), vlastní přidělení zdroje a vrácení zdroje. Tyto operace v Petriho síti modelujeme pomocí odebírání a navrácení značky do místa, jehož značení reprezentuje počet volných procesorů nebo vstup/výstupních zařízení. Chceme, aby značky reprezentující jednotlivé prostředky nebyly ani vytvářeny (generovány), ani ničeny. Požadujeme tedy, aby jejich celkový počet v síti zůstával stále konstantní. Vlastnost Petriho sítě, která stanovuje, že počet značek se prováděním přechodů nemění, se nazývá *konzervativnost* (angl. *conservation*).

DEF

■ **Definice 8.3** *Striktně konzervativní síť.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Síť N se nazývá *striktně konzervativní* (angl. *strictly conservative*), jestliže platí

$$\forall M \in [M_0\rangle \ : \sum_{p \in P} M(p) = \sum_{p \in P} M_0(p)$$

□

Striktní konzervativnost Petriho sítě je velmi silná vlastnost. Snadno lze nahlédnout, že pro striktně konzervativní síť musí platit:

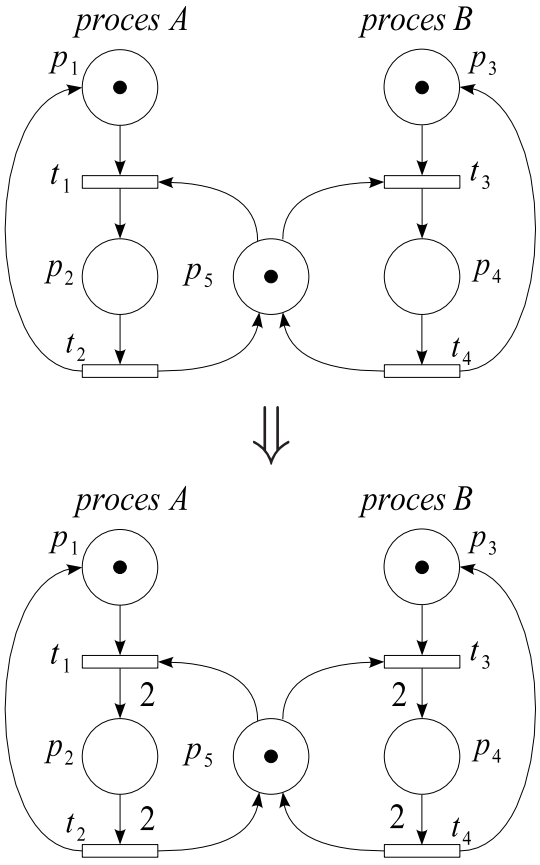
$$\forall t \in T : \sum_{p \in {}^\bullet t} W(p,t) = \sum_{p \in t^\bullet} W(t,p)$$

Pokud by tato podmínka nebyla pro nějaký přechod t splněna, pak by jeho provedení změnilo celkový počet značek sítě. Na obrázku 8.2 je příklad Petriho sítě, která popisuje dva procesy, sdílející jediný procesor (modelovaný místem p_5) a jí odpovídající striktně konzervativní Petriho síť.

Obrázek 8.2 ilustruje časté situace, ve kterých není přiřazení mezi značkami a prostředky zcela jednoznačné. Např. značka v místě p_2 reprezentuje společně proces A i přidělený procesor. Vedle toho jsou v síti obvykle místa, kterých se tok značek reprezentujících prostředky systémů vůbec nedotýká. Abychom zjednodušili popis konzervace prostředků a vyhnuli se transformacím podobným jako na obrázku 8.2, zavedeme pojem konzervativní sítě vzhledem ke stanoveným vahám jednotlivých míst sítě.

DEF

■ **Definice 8.4** *Síť konzervativní vzhledem k váhovému vektoru.*



Obrázek 8.2: Striktně konzervativní Petriho síť

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a $v : P \to \mathbb{N}$ vektor. Síť N je *konzervativní vzhledem k váhovému vektoru* v, jestliže platí

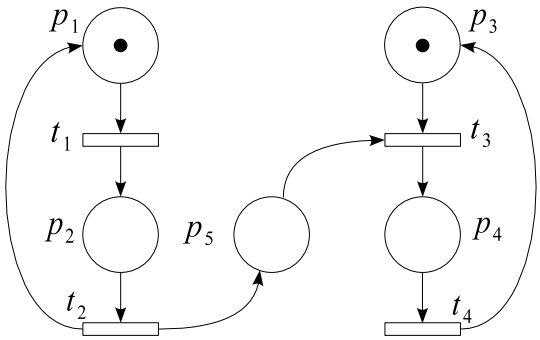
$$\forall M \in [M_0\rangle \ : \sum_{p \in P} v(p)M(p) = \sum_{p \in P} v(p)M_0(p)$$

Petriho síť N nazveme *konzervativní* (angl. *conservative*), je-li konzervativní vzhledem k váhovému vektoru $v > \mathbf{0}$. Připoměňme, že

$$v > \mathbf{0} \Leftrightarrow p \in P : v(p) > 0$$

□

První Petriho síť na obrázku 8.2, protože je konzervativní vzhledem k váhovému vektoru $(1, 2, 1, 2, 1)$ (nebo přesnější $\{p_1 \mapsto 1, p_2 \mapsto 2, p_3 \mapsto 1, p_4 \mapsto 1, p_5 \mapsto 1\}$). Příklad nekonzervativní sítě je na obrázku 8.3.



Obrázek 8.3: Nekonzervativní Petriho síť

8.1.3 Živost, uváznutí

Jednou z nejdůležitějších vlastností, které studujeme u Petriho sítí, souvisí s problémem *uváznutí* (zablokování) modelovaného systému, který je v anglicky psané literatuře označován termínem *deadlock*. Pomocí modelu Petriho sítě si vysvětlíme situaci, kdy může k uváznutí dojít.

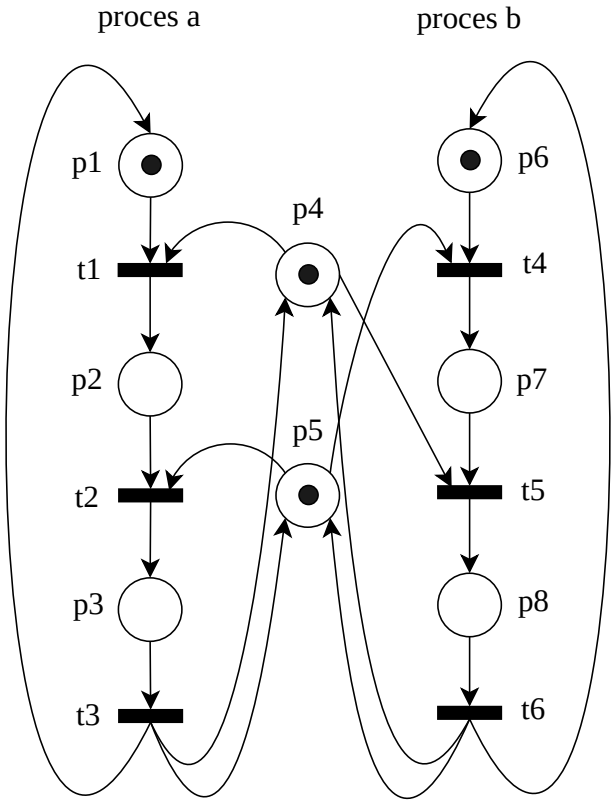
Na obrázku 8.4 je Petriho síť, která je modifikací sítě z obrázku 8.2. V tomto případě procesy A a B požadují přidělení dvou společných (sdílených) zdrojů q a r, které jsou reprezentovány místy p_4 a p_5. Jednotlivé procesy však vyžadují tyto zdroje v opačném pořadí: zatímco proces A vyžaduje zdroje v pořadí q, r, proces B je požaduje v pořadí r, q.

Na počátku jsou oba zdroje volné a oba procesy jsou připraveny k provádění. Uvažujeme-li např. výpočetní posloupnost $t_1t_2t_3t_4t_5t_6$, nebo posloupnost $t_4t_5t_6t_1t_2t_3$, pak Petriho síť modeluje korektní provádění procesů A a B. Jiná situace však nastane při provádění výpočetní posloupnosti začínající prefixem t_1t_4. Po provedení přechodů t_1 a t_4 je procesu A dostane zdroj q a čeká na uvolnění zdroje r, zatímco proces B dostane zdroj r a čeká na uvolnění prostředku q. Procesy A a B tedy čekají na sebe vzájemně. Tento stav, nazývaný *uváznutí* (angl. *deadlock*), znamená úplné zablokování systému. Žádný z procesů A a B nemůže pokračovat a žádný přechod sítě není v tomto okamžiku proveditelný.

DEF

■ **Definice 8.5** *Živé značení, živá síť.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Značení $M \in [M_0\rangle$ je *živé*, jestliže $\forall t \in T$ existuje nějaké značení $M_1 \in [M\rangle$ takové, že



Obrázek 8.4: Přidělování zdrojů – deadlock

přechod t je M_1-proveditelný.

Jsou-li všechna značení z $[M_0\rangle$ živá, pak Petriho síť N je *živá* (angl. *live*). □

Po provedení přechodů t_1 a t_4 Petriho sítě na obrázku 8.4 dostaneme načení $\{p_1 \mapsto 0, p_2 \mapsto 1, p_3 \mapsto 0, p_4 \mapsto 0, p_5 \mapsto 1, p_6 \mapsto 0\}$, nebo zjednodušeně $(0, 1, 0, 0, 1, 0)$. Toto značení není živé, tudíž ani Petriho síť není živá.

Někdy se setkáváme s definicí pojmu živosti vztažené k přechodům sítě, přičemž je možné rozlišit různé kvalitativní hladiny živosti.

DEF

■ **Definice 8.6** *Hladiny živosti.*

Nechť N je Petriho síť s množinou přechodů T a počátečním značením M_0. Přechod t je:

- *živý na hladině 0*, není-li proveditelný v žádném značení z $[M_0\rangle$, je tedy *neživý*
- *živý na hladině 1*, jestliže existuje značení $M \in [M_0\rangle$ takové,

že přechod t je M-proveditelný

- *živý na hladině 2*, jestliže pro každé $n \in \mathbb{N}$ existuje výpočetní posloupnost, ve které se přechod t vyskytuje alespoň n-krát.
- *živý na hladině 3*, jestliže existuje výpočetní posloupnost, ve které se přechod t vyskytuje nekonečněkrát.
- *živý na hladině 4*, jestliže pro každé $M_1 \in [M_0\rangle$ existuje značení $M_2 \in [M\rangle$ takové, že přechod t je M_2-proveditelný.

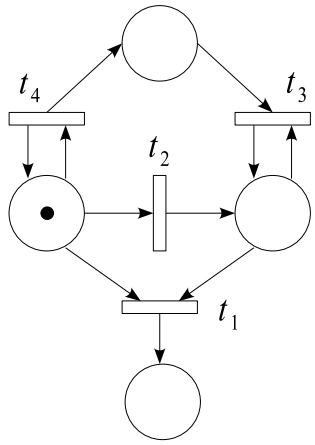
Hladiny živosti přechodu rozšíříme na hladinu živosti Petriho sítě: Petriho síť *živá na hladině* h, jestliže všechny přechody jsou živé na hladině h, nebo vyšší.

Živá Petriho síť dle definice 8.5 odpovídá Petriho síti živé na hladině 4 dle definice 8.6. □

x+y

■ **Příklad 8.1** *Hladiny živosti.*

Na obrázku 8.5 je zobrazena Petriho síť, jejíž přechody jsou klasifikovány do různých hladin živosti.



Obrázek 8.5: Petriho síť ilustrující různé hladiny živosti přechodů

- Přechod t_1 nemůže být proveden nikdy, je živý na hladině 0.
- Přechod t_2 lze provést právě jednou, je živý na hladině 1.
- Přechod t_3 může být proveden n-krát pro libovolné n, avšak počet jeho provedení závisí na počtu provedení přechodu t_4.

Chceme-li provést přechod t_3 5-krát, pak nejprve 5-krát provedeme přechod t_4, poté t_2 a pak 5-krát t_3. Jakmile však je proveden přechod t_2 (a ten musí být proveden, aby t_3 byl proveditelný), pak je počet provedení přechodu t_3 pevně dán. Proto je t_3 živý na hladině 2 a ne na hladině 3.

- Přechod t_4 může být proveden libovolně krát a je tedy živý na hladině 3, avšak ne na hladině 4, protože po provedení přechodu t_2 nemůže již být nikdy proveden.

□

8.1.4 Dosažitelnost a pokrytí

Všechny dosud diskutované vlastnosti Petriho sítě souvisely s dosažitelnými značeními Petriho sítě. Zda je zadané značení dosažitelné z počátečního značení, tedy zda je prvkem stavového prostoru Petriho sítě, patří tedy mezi základní problémy analýzy Petriho sítí.

DEF

▪ **Definice 8.7** *Problém dosažitelnosti.*

Problém dosažitelnosti (angl. *reachability problem*) je formulován takto: Je dána Petriho síť $N = (P, T, F, W, K, M_0)$ a značení $M : P \rightarrow \mathbb{N} \cup \{\omega\}$.

$$\text{Platí } M \in [M_0\rangle \text{ ?}$$

□

Řešením tohoto problému můžeme např. rozhodnout, zda síť není živá, protože může dosáhnout jistého neživého značení, nebo zda je dosažitelný určitý jiný významný stav sítě. Řada problémů analýzy Petriho sítě může být formulována právě s využitím problému dosažitelnosti.

V některých otázkách analýzy Petriho sítě nás nemusí zajímat konkrétní hodnoty značení, kterých lze či nelze dosáhnout, ale pouze otázka, zda lze dosáhnout značení, jehož některé složky jsou větší než jistá celočíselná hodnota. Tento problém, nazývaný problémem pokrytí, vymezují následující definice.

DEF

▪ **Definice 8.8** *Pokrytí značení.*

Nechť N je Petriho síť a M_1, M_2 dvě její značení. Značení M_1 je pokryto značením M_2, pokud $M_1 \leq M_2$, tedy $\forall p \in P : M_1(p) \leq M_2(p)$. □

DEF

■ **Definice 8.9** *Problém pokrytí.*

Problém pokrytí (angl. *coverability problem*) je formulován takto: Je dána Petriho síť $N = (P, T, F, W, K, M_0)$ a značení $M : P \to \mathbb{N} \cup \{\omega\}$. Je dosažitelné takové značení, které pokrývá M, tedy

$$\exists M_1 \in [M\rangle : M \leq M_1?$$

□

8.1.5 Ekvivalence a inkluze

Jinou třídou problémů, o níž se zmíníme, jsou problémy, které se váží na transformace a optimalizace Petriho sítě. Často chceme danou Petriho síť modifikovat s cílem dosáhnout určitých vlastností, např. zvýšit míru paralelismu nebo dosáhnout nižší cenu realizace (implementace) systému. V takovém případě nás přirozeně zajímá, v jakém vztahu jsou původní a transformovaná Petriho sítě. Rozlišujeme dva typy vztahů, které jsou formulovány dvěma problémy:

- *problémem ekvivalence* Petriho sítí (angl. *equivalence problem*)
- *problémem inkluze* Petriho sítí (angl. *subset problem*)

Konkrétní formulace těchto problémů může mít řadu variant. Vztah ekvivalence dvou Petriho sítí může např. požadovat shodnost množin výpočetních posloupností obou sítí, nebo shodnost množin dosažitelných značení, nebo obojí. Vztah inkluze dvou Petriho sítí může, kromě požadavků inkluze množin výpočetních posloupností a množin značení, zohledňovat i transformace, při kterých se mění množina míst transformované sítě (např. vypuštěním místa, které nemůže být nikdy označeno). V tom případě lze použít projekci mezi vektory značení původní a výsledné sítě. V každém případě je však třeba konkrétní problém ekvivalence i inkluze pečlivě a přesně definovat.

8.1.6 Další okruhy problému analýzy

Poslední okruh problémů analýzy se zaměřuje na otázky spojené s výpočetními posloupnostmi dané Petriho sítě. Zajímá nás, zda je proveditelná v Petriho síti daná posloupnost přechodů nebo, zda se v některé výpočetní posloupnosti mohou objevit vedle sebe určité přechody apod. Tyto problémy vedou k pojmu jazyka generovaného danou Petriho sítí, podobně jako u konečných automatů nebo Turingových strojů. Vlastnostmi jazyků generovaných Petriho sítí se budeme zabývat v samostatné podkapitole.

Uvedený soubor problémů analýzy Petriho sítí není vyčerpávající; popisuje však problémy, které se nejčastěji diskutují v odborné literatuře a odrážejí důležité praktické aspekty aplikací Petriho sítí. Nyní se budeme zabývat některými technikami analýzy Petriho sítí.

8.2 Strom dosažitelných značení

Celá řada metod analýzy Petriho sítě vychází z grafové reprezentace stavového prostoru Petriho sítě, nazývané *strom dosažitelných značení* (angl. *reachability tree*) resp. *strom pokrytí* (angl. *coverability tree*). Strom dosažitelných značení je ve skutečnosti určitou abstrakcí přechodové funkce Petriho sítě (definice 7.11) definované na potenciálně nekonečném stavovém prostoru možných značení sítě. Strom dosažitelných značení připouští pouze konečný počet stavů a diagram modifikované přechodové funkce vyjadřuje ve tvaru orientovaného kořenového stromu, jehož kořenem je počáteční značení Petriho sítě. Principem zobrazení nekonečného stavového prostoru na konečnou množinu vrcholů stromu je agregace určitých nekonečných podmnožin značení a jejich zobrazení v konečném počtu vrcholů stromu. Jestliže v průběhu konstrukce stromu dosažitelných značení zjistíme, že jistá složka vektoru značení roste nade všechny meze, pak tato složka nabude hodnoty ω a příslušný vektor reprezentuje nekonečnou množinu značení, pro která tato složka nabývá libovolné nezáporné celočíselné hodnoty.

Dříve, než formálně popíšeme algoritmus konstrukce stromu dosažitelných značení, ukážeme si tvar postupně získávaných stromů a tvar výsledného stromu dosažitelných značení.

x+y

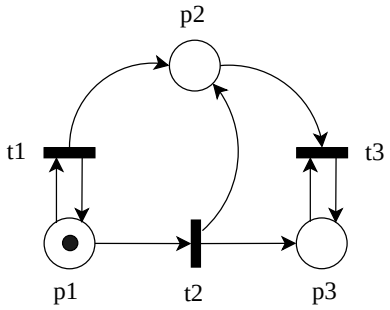
■ **Příklad 8.2** *Konstrukce stromu dosažitelných značení.*

Na obrázku 8.6 je dána Petriho síť, pro kterou hledáme strom dosažitelných značení. Začátek postupu (v prvních třech krocích) a výsledek ilustruje obrázek 8.7.

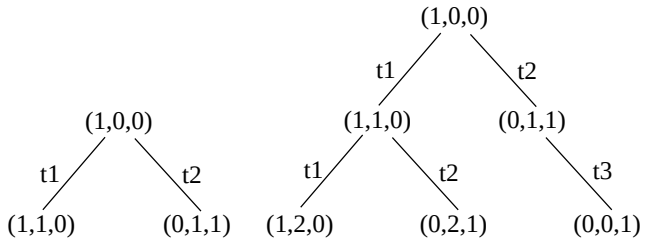
□

8.2.1 Algoritmus konstrukce stromu dosažitelných značení

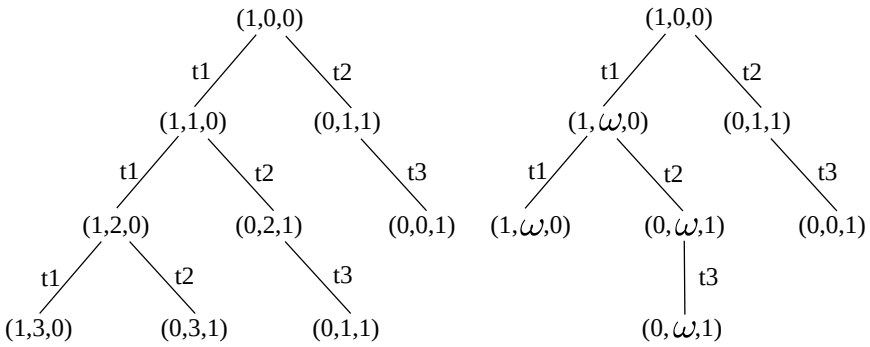
Každý vrchol x stromu bude ohodnocen značením $M_x : P \rightarrow \mathbb{N} \cup \{\omega\}$. Vrcholy budou dále klasifikovány do 4 typů : *čelní, koncový, duplikovaný a vnitřní vrchol*. Čelní vrcholy jsou ty vrcholy, které jsou algoritmem právě zpracovávány. Jsou přeměněny na koncové, nebo duplikované, nebo vnitřní. Algoritmus začíná od kořene stromu, kterému je



Obrázek 8.6: Petriho síť pro ilustraci stromu dosažitelných značení



(a) 1. krok (b) 2. krok



(c) 3. krok (d) výsledný strom

Obrázek 8.7: Konstrukce stromu dosažitelných značení

přiřazeno počáteční značení M_0, a který je jediným čelním vrcholem. Algoritmus pracuje tak dlouho, pokud zůstávají čelní vrcholy, podle těchto pravidel:

Nechť x je právě zpracovávaný čelní vrchol s ohodnocením M_x

1. Existuje-li jiný vrchol y vytvářeného stromu, který není čelní a je ohodnocen stejným značením jako vrchol x, tj. $M_x = M_y$, pak se vrchol M_x stává *duplikovaným vrcholem.*

2. Jestliže pro značení M_x neexistuje přechod $t \in T$, pro který by t byl M_x-proveditelný (tj. $\delta(M_x, t)$ není definováno), pak M_x je *koncový vrchol* stromu.

3. Pro přechod t, který je M_x-proveditelný, vytvoříme nový čelní vrchol z stromu. Ohodnocení M_z je definováno následovně. Pro všechna $p \in P$:

   (a) je-li $M_x(p) = \omega$, pak $M_z(p) = \omega$

   (b) existuje-li na cestě z kořene stromu do vrcholu x vrchol y takový, že $M_y \leq \delta(M_x, t)$, (tj. značení, které vznikne provedením přechodu t ze značení M_x, pokrývá značení M_y), a jestliže $M_y(p) < \delta(M_x, t)(p)$, pak $M_z(p) = \omega$

   (c) jinak $M_z(p) = \delta(M_x, t)(p)$

   Hrana vedená z vrcholu x do vrcholu z je označena přechodem t a vrchol z se stává čelním vrcholem.

Jakmile jsou všechny vrcholy stromu převedeny na koncové, duplikované nebo vnitřní, algoritmus končí. Výsledný strom je stromem dosažitelných značení dané Petriho sítě.

Věta

■ **Věta 8.1** *Konečnost stromu dosažitelných značení*

Nechť N je Petriho síť. Strom dosažitelných značení sítě N vytvořený podle předchozího algoritmu je vždy konečný.

Důkaz. Důkaz věty vyžaduje několik pomocných tvrzení. Lze ho nalézt v [8]. □ □

Nyní si ukážeme, jak lze některé z problémů analýzy Petriho sítí řešit na základě příslušného stromu dosažitelných značení.

8.2.2 Omezenost a bezpečnost sítě

Jestliže pro danou síť N nalezneme příslušný strom dosažitelných značení, pak rozhodnutí omezenosti či bezpečnosti sítě je jednoduché. Je-li některý z vrcholů stromu označen značením obsahujícím symbol ω, pak daná síť není omezená. Naopak není-li daná sít omezená, pak její množina dosažitelných značení je nekonečná, a proto musí existovat alespoň jeden vrchol stromu, jenž je ohodnocen značením obsahujícím symbol ω.

Absence symbolu ω ve značeních vrcholů stromu tedy jednoznačně indikuje omezenost sítě a lze snadno určit hodnotu k (maximální hodnotu libovolné složky značení) určující k-omezenost sítě. Je-li $k = 1$, pak je síť bezpečná.

Omezená Petriho síť má konečný stavový prostor, její strom dosažitelných značení je v podstatě stromovým vyjádřením přechodové funkce konečného automatu a může být použit nejen pro analýzu vlastností značení sítě, ale také pro analýzu výpočetních posloupností sítě.

Strom dosažitelných značení je užitečný pro další analýzu i v případě neomezené sítě. Může nás zajímat, která místa jsou neomezená nebo jaká je hodnota k udávající k-omezenost význačných míst sítě.

8.2.3 Konzervativnost

Stromu dosažitelných značení lze použít i pro efektivní analýzu konzervativnosti sítě, která předpokládá že součet značek všech míst, vážený váhovým vektorem $v : P \to \mathbb{N} \setminus \{0\}$, je vždy konstantní. Máme-li tedy Petriho síť a její strom dosažitelných značení, pak rozhodnutí konzervativnosti sítě vyžaduje nalezení tohoto váhového vektoru.

Nechť množina míst uvažované Petriho sítě je $P = \{p_1, \ldots, p_n\}$ a množina vrcholů stromu dosažitelných značení je $U = \{u_1, \ldots, u_m\}$ a nechť M_{u_i} je značení, kterým je ohodnocen vrchol u_i. Dále předpokládejme, že Petriho síť je konzervativní, tj. platí

$$\forall p \in [M_0\rangle : \sum_{p \in P} M(p).v(p) = konst$$

M_0 je počáteční značení sítě a *konst* je určitá konstanta

Pak musí existovat řešení soustavy lineárních rovnic a nerovností tvaru:

$$\begin{aligned} v.M_{u_1} &= konst \\ v.M_{u_2} &= konst \\ &\vdots \\ v.M_{u_m} &= konst \\ v(p_i) &> 0 \qquad \forall i = 1, \ldots, n \end{aligned}$$

pro $n + 1$ neznámých $v(p_i)$ a *konst* v oboru kladných celých čísel.

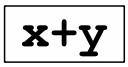
Řešení soustavy je známý problém, na který lze aplikovat metodu lineárního programování, nebo modifikované metody řešení soustavy lineárních rovnic. Pokud existuje jeho řešení v oboru kladných racionálních čísel, pak výsledné celočíselné váhy (i hodnotu *konst*) získáme

převedením výsledku na společného jmenovatele. Pokud existuje řešení ve tvaru $v(p_i) = 1$ pro všechny $i = 1, \ldots, n$, pak Petriho síť je *striktně konzervativní*.

Řešení uvedené soustavy však nemusí existovat. V tom případě uvažovaná Petriho síť není konzervativní. Tato situace zřejmě nastane vždy, není-li Petriho síť omezená, protože výskyt ω v některém vektoru M_{u+i} představuje libovolný celočíselný příspěvek k číslu *konst*. Prakticky však můžeme problém konzervativnosti řešit pro určitou podsíť vyloučením neomezených míst tak, že položíme $v(p_i) = 0$ pro ta místa, pro která existuje $u \in U : M_u(p_i) = \omega$.

8.2.4 Problém pokrytí

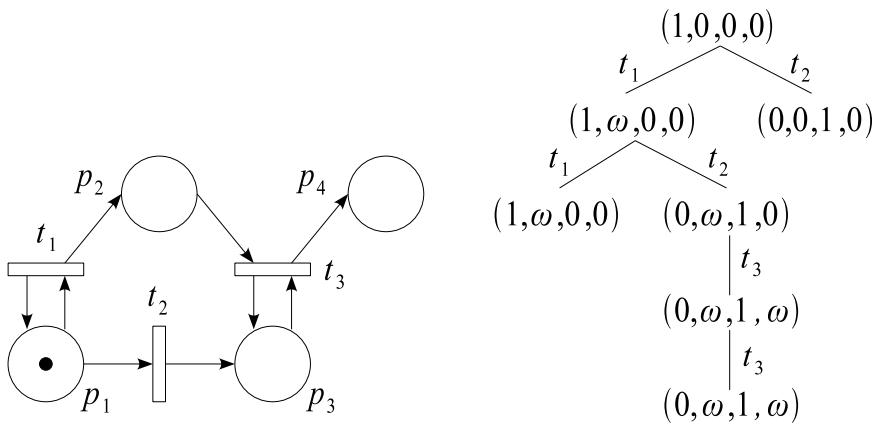
Další problém, který lze úspěšně řešit s využitím stromu dosažitelných značení je problém pokrytí: existuje pro dané značení M dostupné značení M', tak že $M \leq M'$? Algoritmus nalezení značení M' je zřejmý; procházíme všechny vrcholy stromu dosažitelných značení a testujeme nerovnost $M \leq M'$ (připomeňme, že platí $k < \omega$ pro libovolné $k \in \mathbb{N}$). Pokud takový vrchol neexistuje, pak rovněž neexistuje pokrytí M' a algoritmus končí v konečném počtu kroků. Nalezneme-li vrchol x, pro který $M \leq M_x$, pak M_x je hledané pokrytí a cesta z kořene do vrcholu x specifikuje výpočetní posloupnost realizující toto pokrytí. Tato specifikace však není jednoznačná v případě, že M_x obsahuje symbol ω; pro $M_x \in \mathbb{N}^n$ je nutné příslušný přechod výpočetní posloupnosti v dostatečném počtu iterovat. Pokud více složek vektoru M_x je rovno symbolu ω, pak se zde mohou objevit vzájemné interakce mezi změnami značení a iteracemi příslušných přechodů, jak ilustruje příklad 8.3.



■ **Příklad 8.3** *Řešení problému pokrytí pomocí stromu dosažitelných značení*

Na obrázku 8.8 je Petriho síť a její strom dosažitelných značení. Ptáme se, zda existuje pokrytí značení např. $(0, 14, 1, 7)$ a jaký tvar mají výpočetní posloupnosti, které k tomuto pokrytí vedou. Odpověď na první část otázky je zřejmá; vrchol $(0, \omega, 1, \omega)$ stromu dosažitelných značení zaručuje existenci nekonečně mnoha pokrytí. K tomuto vrcholu vede cesta označená přechody $t_1 t_2 t_3$; výpočetní posloupnost vedoucí k pokrytí značení $(0, 14, 1, 7)$ proto bude mít tvar $t_1^+ t_2 t_3^+$. Počet iterací přechodu t_1 však není 14, jak bychom se mohli domnívat, ale minimálně 21, protože nezbytných 7 iterací přechodu t_3 odstraní 7 značek z místa p_2. □

I v takových případech však existuje algoritmus nalezení výpočetní posloupnosti realizující žádané pokrytí, pokud toto pokrytí sa-



Obrázek 8.8: Ilustrace řešení problému pokrytí

mozřejmě existuje.

8.2.5 Omezení aplikace stromu dosažitelných značení

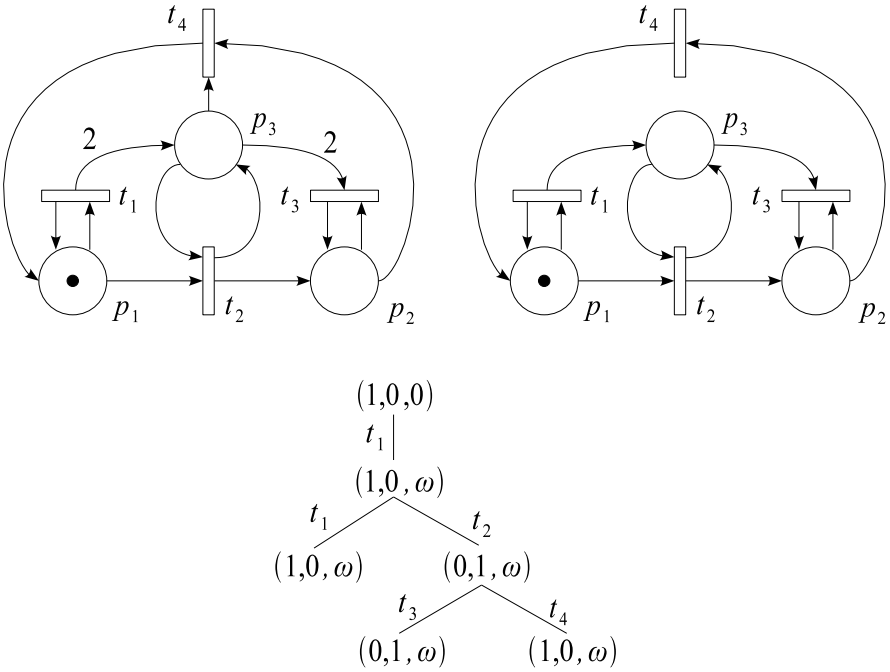
Na rozdíl od dosud diskutovaných úloh analýzy Petriho sítě je použití stromu dosažitelných značení při řešení problému živosti sítě a problému dosažitelnosti značení omezené. To vyplývá ze skutečnosti, že strom dosažitelných značení je abstrakcí přechodové funkce, v níž dochází ke ztrátě určité informace. Vztah mezi Petriho sítí a příslušným stromem dosažitelných značení není pak jednojednoznačný; dvěma různým Petriho sítím může příslušet stejný strom, jak ukazuje obrázek 8.9. Obrázek 8.9 můžeme využít i pro ilustraci podstaty neřešitelnosti obecného problému živosti sítě a problému dosažitelnosti na základě stromu dosažitelných značení.

Síť N_1 není živá. Např. výpočetní posloupnost $t_1t_2t_3$ vede ke značení $(0,1,0)$:

$$(1,0,0)\ [t_1\rangle\ (1,0,2)\ [t_2\rangle\ (0,1,2)\ [t_3\rangle\ (0,1,0)$$

pro které není již žádný z přechodů sítě N_1 proveditelný. Na druhé straně síť N_2 je živá; v její množině dosažitelných značení neleží značení $(0,1,0)$. Vrchol $(0,1,\omega)$ sice reprezentuje nekonečnou množinu značení lišících se poslední složkou, avšak obecně neplatí, že libovolný její prvek je dosažitelný.

Přesto může být strom dosažitelných značení někdy využit i pro vyřešení otázek týkajících se živosti sítě, či dosažitelnosti značení.



Obrázek 8.9: Stejný strom dosažitelných značení pro různé Petriho sítě

Například, obsahuje-li strom dosažitelných značení koncový vrchol (vrchol bez následníků), pak příslušná síť nemůže být živá. Podobně, je-li značením $M \in \mathbb{N}^n$ ohodnocen jistý vrchol stromu, pak je M dostupné. Na druhé straně, není-li M pokrytelné žádným značením stromu, pak je jistě nedostupné.

Jiné, vážnější omezení aplikovatelnosti stromu dosažitelných značení souvisí s jeho výpočtovou složitostí. Zvláště kritické jsou paměťové nároky algoritmu jeho konstrukce. Pro modifikaci stromu dosažitelných značení, zvanou *graf dosažitelných značení* (připouštějící cykly grafu) byla dokázána následující vlastnost [10]:

Existuje posloupnost Petriho sítí $N_1, N_2, \ldots$, jejichž velikost (celkový počet vrcholů, hran a počátečních značek) roste lineárně, taková, že odpovídající grafy dosažitelných značení $G_1, G_2, \ldots$ rostou (vzhledem k počtu vrcholů) rychleji než libovolná primitivně rekurzivní funkce. Toto tvrzení má značně nepříznivé důsledky pro univerzální využití stromu dosažitelných značení:

Nechť N_1 a N_2 jsou Petriho sítě s počátečním značením M_0^1 resp. M_0^2, které mají identické množiny míst a konečné množiny dosažitelných značení $[M_0^1\rangle$ resp. $[M_0^2\rangle$.

1. Je rozhodnutelné jestli $[M_0^1\rangle \subseteq [M_0^2\rangle$, avšak ne v rekurzivně primitivním čase nebo prostoru. Podobný výsledek platí pro pro-

blém $[M_0^1\rangle = [M_0^2\rangle$.

2. Jsou-li $[M_0^1\rangle$ a $[M_0^2\rangle$ nekonečné množiny, pak oba problémy jsou nerozhodnutelné.

Pro Petriho síť velikosti n je rozhodnutelné s paměťovou složitostí $2^{c.n.log(n)}$, zda je množina $[M_0\rangle$ dosažitelných značení konečná [10]. Stejnou složitost pak má i problém pokrytí.

K řešení těchto problémů tedy není nutné konstruovat strom dosažitelných značení. Navíc je zde ukázáno, že oba problémy nemohou být rozhodnuty v prostoru $2^{\sqrt{n}}$.

Pojmy k zapamatování

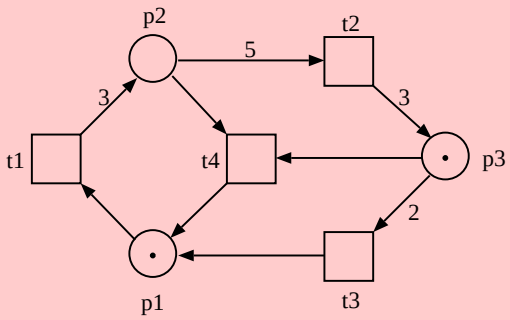
- bezpečná síť, omezená síť
- striktně konzervativní síť, živá síť
- strom dosažitelných značení

Shrnutí

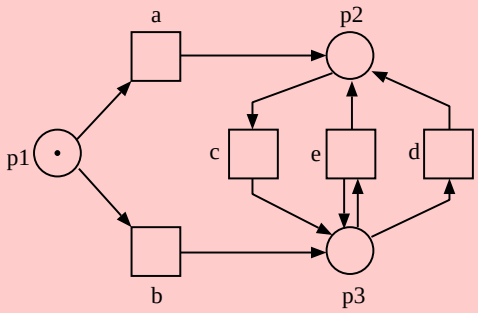
V této kapitole jsme se soustředili na problém analýzy P/T Petriho sítí, která nám umožňuje získat důležité informace o vlastnostech dané sítě a může být použita k ověření určitých žádoucích (či nežádoucích) rysů modelovaného systému. Jednou z nejdůležitějších vlastností, které studujeme u Petriho sítí, souvisí s problémem uváznutí modelovaného systému. Mezi základní problémy analýzy Petriho sítí patří také, zda je zadané značení dosažitelné z počátečního značení, tedy zda je prvkem stavového prostoru Petriho sítě. Poslední okruh problémů analýzy se zaměřuje na otázky spojené s výpočetními posloupnostmi dané Petriho sítě.

Příklady k procvičení

1. Uvažujte Petriho síť na obrázku. Na základě stromu dosažitelných značení rozhodněte:
   (a) Která místa sítě jsou neomezená? Zdůvodněte.
   (b) Je tato síť živá? Zdůvodněte.



2. Pro Petriho síť na obrázku sestrojte strom dosažitelných značení. Dále ukažte (nebo vyvraťte), že:
   (a) $\neg\exists M' \in [M_0\rangle$ takové, že $(1,2,3) < M'$
   (b) síť je omezená
   (c) síť je živá



Kapitola 9

Invarianty

Čas potřebný ke studiu: 6 hodin

Cíle kapitoly

Nyní se budeme zabývat metodami analýzy, které jsou založeny na lineární algebraické reprezentaci Petriho sítě (sekce 7.5). Budou nás zajímat množiny míst, které nemění svoje značky v průběhu provádění přechodů. Jejich znalost (identifikace) nám pomůže v analýze živosti sítě a v ověřování určitých specifických vlastností, které má modelovaný systém zachovávat. Množiny takových míst se nazývají *P-invarianty* .
T-invarianty udávají kolikrát je třeba, počínaje určitým značením, provést každý přechod sítě, abychom získali nazpět toto značení (reprodukovali dané značení sítě).

Průvodce studiem

Ke studiu této kapitoly budete potřebovat nejen znalosti z kapitol 7 a 8, ale rovněž základní znalosti lineární algebry, použití matic a řešení soustav rovnic.

Obsah

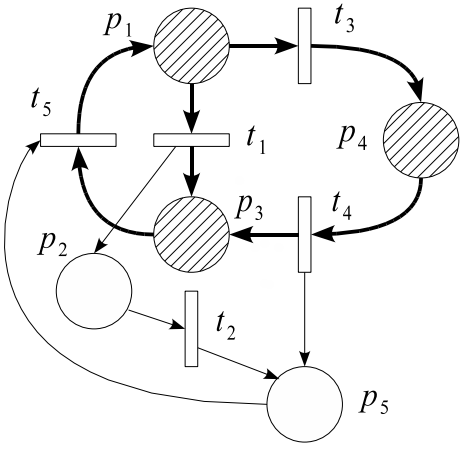
9.1 P-invarianty

Nejdříve uvažujme speciální třídu P-invariantů. Nechť N je Petriho síť, $N = (P, T, F, W, K, M_0)$, pro kterou $\forall f \in F : W(f) = 1$, tj. váhy všech hran jsou rovny 1. Chceme charakterizovat množiny $\Pi \subseteq P$, které zachovávají celkový součet značek po libovolném provedení přechodu. Aby tomu tak bylo, pak, je-li Π taková množina míst a $p \in \Pi$, pak pro každý přechod $t \in p^\bullet$ musí existovat místo $p' \in t^\bullet$, které je také obsaženo v Π. Jinak řečeno, značka se přemisťuje po hranách (p, t) a (t, p') z místa p do místa p'. Analogicky, pro každý proveditelný přechod $t \in {}^\bullet p$ existuje místo $p' \in {}^\bullet t$ takové, že značka se přemísťuje po hranách (p', t) a (t, p) z místa p' do p. Tedy množina Π může být charakterizována množinou hran $\Phi \subseteq F$, které splňují tyto podmínky:

1. Jestliže hrana $f \in \Phi$ začíná nebo končí v místě p, pak všechny hrany vedoucí z místa nebo do místa p patří do množiny Φ.

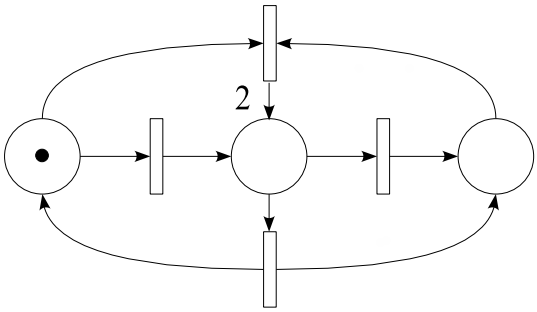
2. Pro každou hranu $f \in \Phi$ končící v určitém přechodu t, existuje právě jedna hrana $f' \in \Phi$ začínající v t.

Na obrázku 9.1 je uvedena taková množina míst Π; odpovídající množinu Φ tvoří silně vytažené hrany. Jiná množina míst splňující požadavek neměnného součtu značek je množina $\{p_1, p_2, p_4, p_5\}$. Popsaný



Obrázek 9.1: Ilustrace specifických podmnožin míst

způsob určování množiny Π, resp. Φ není dostačující v případě, že Petriho síť obsahuje hrany s váhou větší než 1. Příkladem může být síť na obrázku 9.2, která je striktně konzervativní. Hledaná množina Π



Obrázek 9.2: Striktně konzervativní Petriho síť

obsahuje všechna místa sítě. Pokusme se tedy nyní odvodit, co musí platit pro množinu míst Π v obecné Petriho síti. Má-li zůstat součet značek míst z $\Pi \subseteq P$ nezměněný při provedení libovolného přechodu $t \in T$, pak

$$\sum_{p \in {}^\bullet t \cap \Pi} W(p,t) = \sum_{p \in t^\bullet \cap \Pi} W(t,p)$$

Podle definice 7.12 je tato podmínka ekvivalentní podmínce

$$\sum_{p \in {}^\bullet t \cap \Pi} \underline{t}(p) = - \sum_{p \in t^\bullet \cap \Pi} \underline{t}(p)$$

tedy

$$\sum_{p \in {}^\bullet t \cap \Pi} \underline{t}(p) + \sum_{p \in t^\bullet \cap \Pi} \underline{t}(p) = 0$$

kterou lze dále upravit do tvaru:

$$\sum_{p \in ({}^\bullet t \cup t^\bullet) \cap \Pi} \underline{t}(p) = 0$$

a dokonce do tvaru

$$\sum_{p \in \Pi} \underline{t}(p) = 0$$

Nahradíme-li nyní množinu Π jejím charakteristickým vektorem c_Π, pak lze podmínku na množinu míst Π zapsat ve tvaru

$$\sum_{p \in \Pi} \underline{t}(p).c_\Pi(p) = 0$$

nebo vektorově

$$\underline{t}.c_\Pi = \mathbf{0}$$

Pokud se součet značek v místech z Π nemá změnit při provedení libovolného přechodu, pak podmínka $\underline{t}.c_\Pi = \mathbf{0}$ musí platit pro všechny přechody $t \in T$ a tudíž musí platit

$$\underline{N}^\mathsf{T}.c_\Pi = \mathbf{0}$$

kde $\underline{N}$ je matice Petriho sítě N.

Naopak, každé řešení rovnice $\underline{N}^\mathsf{T}.x = \mathbf{0}$, které obsahuje pouze složky z $\{0, 1\}$ je charakteristickým vektorem množiny Π, která zachovává součet značek.

Budeme se zabývat vztahem i ostatních řešení rovnice $\underline{N}^\mathsf{T}.x = \mathbf{0}$ k vlastnostem určitých podmnožin míst a zavedeme třídu obecných invariantů.

DEF

■ **Definice 9.1** *P-invariant, binární P-invariant.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Vektor $i : P \to \mathbb{Z}$ nazýváme *P-invariantem* Petriho sítě N, jestliže platí $\underline{N}^\mathsf{T}.i = \mathbf{0}$. Jestliže $i(p) \in \{0, 1\}$ pro všechna $p \in P$, pak i nazýváme *binárním P-invariantem* sítě N. □

Lemma

■ **Lemma 9.1** *Součet a násobek P-invariantu.*

Nechť i_1 a i_2 jsou P-invarianty sítě N a nechť $z \in \mathbb{Z}$. Pak $i_1 + i_2$ a $z.i_1$ jsou také P-invarianty sítě N.

Důkaz. Je jednoduchým cvičením z lineární algebry. □ □

x+y

■ **Příklad 9.1** *P-invarianty.*

Na obrázku 9.3 jsou uvedeny všechny P-invarianty Petriho sítě z obrázku 9.1. Invarianty i_1 a i_2 jsou binární invarianty odpovídající množinám $\{p_1, p_3, p_4\}$ a $\{p_1, p_2, p_4, p_5\}$. □

	t_1	t_2	t_3	t_4	t_5	i_1	i_2	i_3	i_4
p_1	−1	0	−1	0	1	1	1	2	0
p_2	1	−1	0	0	0	0	1	1	1
p_3	1	0	0	1	−1	1	0	1	−1
p_4	0	0	1	−1	0	1	1	2	0
p_5	0	1	0	1	−1	0	1	1	1

Obrázek 9.3: Matice a P-invarianty sítě z obrázku 9.1

Zamysleme se nyní nad tím, jakou interpretaci mají nebinární invarianty; v předchozím příkladě invarianty i_3 a i_4. Jestliže pečlivě prohlédneme síť na obrázku 9.1, zjistíme, že jedna značka v místě p_1 odpovídá dvěma značkám v místech p_2 a p_3 dohromady a podobně značka v místě p_4 odpovídá dvěma značkám rozloženým v místech p_3 a p_5. Tedy značky v místech p_1 a p_4 mají váhu dvojnásobnou než značky v místech p_2, p_3 a p_5. Uvažujme proto takto vážený součet značek. Pak pro každá dvě značení M_1 a M_2, pro která $M_1\ [t\rangle\ M_2$, kde $t \in \{t_1, t_2, t_3, t_4, t_5\}$, platí

$$2M_1(p_1) + 2M_1(p_4) + M_1(p_2) + M_1(p_3) + M_1(p_5) = \\ 2M_2(p_1) + 2M_2(p_4) + M_2(p_2) + M_2(p_3) + M_2(p_5)$$

což vyjadřuje právě invariant i_3:

$$M_1.i_3 = M_2.i_3$$

Podobně lze z obrázku 9.1 zjistit, že pro každé dostupné značení $M \in\ [M_0\rangle$ zůstává v platnosti vztah

$$M(p_3) = M(p_2) + M(p_5)$$

což je podmínka odpovídající poslednímu invariantu i_4:

$$M_0.i_4 = M.i_4$$

Koncept P-invariantů souvisí do značné míry s již zavedeným pojmem Petriho sítě konzervativní vzhledem k váhovému vektoru. Každý P-invariant může být považován za obecný váhový vektor, v němž jsou povoleny i záporné složky. Na rozdíl od pojmu konzervativnosti, který akceptoval požadavek neměnícího se počtu značek celé sítě či podsítě, je pojem P-invariantu obecnější a odráží určité podmínky systému, které by měly být zachovány.

Věnujme dále pozornost některým vlastnostem P-invariantů, které usnadní analýzu Petriho sítě.

Věta

■ **Věta 9.1** *Značení sítě vzhledem k P-invariantu.*

Nechť N je Petriho síť s počátečním značením M_0. Pak pro každý P-invariant i sítě N a pro každé dosažitelné značení $M \in\ [M_0\rangle$ platí

$$M.i = M_0.i$$

Důkaz. Nechť $M_1, M_2 \in [M_0\rangle$ a nechť $t \in T$ takové, že $M_1\ [t\rangle\ M_2$. Pak podle věty 7.1 platí $M_2 = M_1 + \underline{t}$. Dále platí, že $\underline{t}.i = \mathbf{0}$, neboť i je invariant. Proto

$$M_2.i = (M_1 + \underline{t}).i = M_1.i + \underline{t}.i = M_1.i$$

□ □

Opačná implikace platí pouze tehdy, může-li být každý přechod sítě N proveden alespoň jednou, tzn. speciálně platí pro živé sítě.

■ **Věta 9.2** *P-invariant v živé síti.* Věta

Nechť N je živá Petriho síť a nechť $i : P \to \mathbb{Z}$ je vektor míst, pro který platí

$$\forall M \in [M_0\rangle : M.i = M_0.i$$

Pak i je P-invariant.

Důkaz. Stačí dokázat, že pro každý přechod $t \in T$ platí $\underline{t}.i = \mathbf{0}$. Nechť tedy $t \in T$ a $M \in [M_0\rangle$ a nechť t je M-proveditelný. Pak $M\ [t\rangle\ M'$ a platí

$$M.i = M'.i = (M + \underline{t}).i = M.i + \underline{t}.i$$

Tudíž $\underline{t}.i = \mathbf{0}$. □ □

Je zřejmé, že místo, které může získat neomezený počet značek, nemůže patřit žádnému kladnému P-invariantu. Této vlastnosti lze využít pro analýzu omezenosti Petriho sítě.

■ **Definice 9.2** *Pokrytí sítě P-invarianty.* DEF

Petriho síť N je *pokryta P-invarianty*, jestliže pro každé místo $p \in P$ existuje nezáporný P-invariant i sítě N takový, že jeho složka $i(p) > 0$. □

■ **Věta 9.3** *Kladný P-invariant.* Věta

Je-li Petriho síť N pokryta P-invarianty, pak existuje P-invariant i sítě N, pro který platí $\forall p \in P : i(p) > 0$.

Důkaz. Podle předpokladu, pro každé $p \in P$ existuje invariant i_p sítě N, pro který $i_p(p) > 0$. Podle lemmatu 9.1 je invariantem také

$$i = \sum_{p \in P} i_p$$

Každá složka vektoru i je proto vypočtena podle

$$i(p') = \sum_{p \in P} i_p(p')$$

je tedy součtem $|p|$ nezáporných hodnot, z nichž nejméně jedna je větší než nula – konkrétně je to ta složka $i_p(p')$, kde $p = p'$. Proto i součet je větší než nula, tedy $\forall p \in P : i(p) > 0$ □ □

Věta

■ **Věta 9.4** *Pokrytí P-invarianty a omezenost sítě.*

Nechť N je Petriho síť s konečným počátečním značením M_0. Je-li N pokryta P-invarianty, pak je omezená.

Důkaz. Nechť $q \in P$ je libovolné místo sítě N a i je P-invariant, pro který $i(q) > 0$ a nechť $M \in [M_0\rangle$. Pomocí aplikace věty 9.1

$$M(q).i(q) \leq \sum_{p \in P} M(p).i(p) = M.i = M_0.i$$

Odtud dostaneme

$$M(q) \leq M_0 \frac{i}{i(q)}$$

□ □ Opačné tvrzení k větě 9.4, tj. je-li síť omezená, pak je pokryta P-invarianty, obecně neplatí.

Nyní si ukážeme, na příkladě nepříliš rozsáhlého systému, jak mohou být použity P-invarianty k ověření určitých strukturálních vlastností modelu.

x+y

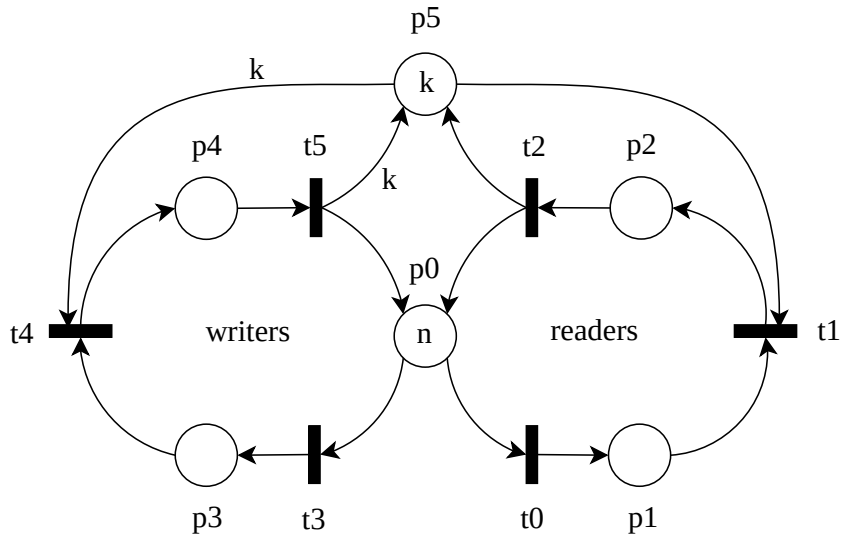
■ **Příklad 9.2** *Čtenáři a písaři.*

Uvažujme model, který je v operačních systémech označován termínem *čtenáři a písaři* (angl. *readers and writers*). Každý z n procesů operačního systému může používat společnou vyrovnávací paměť (buffer), aby do ní určitá data zapsal nebo z ní data přečetl. Má-li být zajištěna spolehlivost operačního systému, pak je nutné přístup procesů k vyrovnávací paměti určitým způsobem řídit. Předpokládejme, že se procesy budou chovat podle těchto pravidel:

- Jestliže žádný proces nezapisuje data do vyrovnávací paměti, pak nejvýše k procesů, $k \leq n$ může simultánně číst z vyrovnávací paměti.

- Přístup libovolného procesu, který chce zapisovat, je povolen pouze tehdy, jestliže žádný z procesů ani nečte, ani nezapisuje z/do vyrovnávací paměti.

Petriho síť tohoto modelu je na obrázku 9.4. Procesy se mohou nacházet v jednom z pěti stavů, odpovídajících místům p_0, p_1, p_2, p_3, p_4. Na počátku jsou všechny procesy pasivní a synchronizační místo p_5 obsahuje k značek.



Obrázek 9.4: Čtenáři a písaři

Ukážeme, že na základě invariantů této Petriho sítě, které jsou uvedeny v tabulce na obrázku 9.5, můžeme dokázat korektnost návrhu daného systému. Z jednotlivých invariantů i_1 a i_2 vyplývá pro každé

	t_0	t_1	t_2	t_3	t_4	t_5	i_1	i_2	M_0
p_0	-1	0	1	-1	0	1	1	0	n
p_1	1	-1	0	0	0	0	1	0	0
p_2	0	1	-1	0	0	0	1	1	0
p_3	0	0	0	1	-1	0	1	0	0
p_4	0	0	0	0	1	-1	1	k	0
p_5	0	-1	1	0	$-k$	k	0	1	k

Obrázek 9.5: Matice, P-invarianty a počáteční značení

značení $M \in [M_0\rangle$ následující:

- Invariant i_1:

$$\sum_{i=0}^{4} M(p_i) = \sum_{i=0}^{4} M_0(p_i) = n$$

To znamená, že počet procesů je konstantní, roven n (žádné procesy se neztrácejí, ani nepřibývají), a že každý proces je v jednom ze stavů p_0, p_1, p_2, p_3, p_4.

- Invariant i_2:

$$M(p_2) + k.M(p_4) + M(p_5) = M_0(p_2) + k.M_0(p_4) + M_0(p_5)$$

Tedy, p_4 obsahuje nanejvýš jednu značku, tj. vždy existuje nejvýše jeden zapisující proces. Obsahuje-li místo p_4 značku, pak $M(p_2) = M(p_5) = 0$, tj. jakmile některý z procesů zapisuje, žádný proces nečte. Místo p_2 může obsahovat maximálně k značek, tj. maximálně k procesů čte simultánně z vyrovnávací paměti. Jestliže žádný z procesů nečte, $M(p_4) = 0$, pak p_2 může obsahovat k značek a pak je synchronizační místo p_5 prázdné.

□

Na základě těchto faktů můžeme dokázat následující tvrzení.

Tvrzení.

Petriho síť na obrázku 9.4 s uvedeným počátečním značením a s kapacitami míst

$$K(p_i) = \begin{cases} 1, & \text{pro } i = 4 \\ k, & \text{pro } i \in \{2, 5\} \\ n, & \text{pro } i \in \{0, 1, 3\} \end{cases}$$

je živá.

Důkaz. Specifikované kapacity $K(p_i), i \in \{0, 1, 2, 3, 4, 5\}$, jak jsme již ukázali, neovlivní živost sítě. Prověříme dále, zda pro každé značení $M \in [M_0\rangle$ je alespoň jeden přechod M-proveditelný. V případě $M(p_0) + M(p_2) + M(p_4) > 0$ vidíme z grafu sítě, že může být proveden alespoň jeden z přechodů t_0, t_2, t_3 nebo t_5. Jestliže je $M(p_0) + M(p_2) + M(p_4) = 0$, pak z i_1 plyne $M(p_1) + M(p_3) = n$ a y i_2 plyne $M(p_5) = k$. Tedy t_1 nebo t_3 jsou proveditelné. Nyní, jestliže p_0 je prázdné pro nějaké $M \in [M_0\rangle$, pak může získat značku v následujících krocích. Z toho plyne živost přechodů t_0 a t_3. Živost ostatních přechodů pak plyne zcela zjevně. □ □

9.2 T-invarianty

Nyní se budeme zabývat řešením soustavy rovnic tvaru:

$$\underline{N}.u = \mathbf{0}$$

Předpokládejme, že vektor $u : T \to \mathbb{N}$ je takovým řešením. Jestliže je možné, počínaje určitým značením M, provést každý přechod t přesně $u(t)$-krát, pak opět získáme značení M.

Skutečně, nechť c_t je charakteristický vektor množiny $\{t\}$, $t \in T$, pak $\underline{t} = \underline{N}.c_t$ Jestliže $M_0\ [t\rangle\ M_1$, pak $M_0 + \underline{t} = M_1$ (věta 7.1).

Podobně, jestliže $M_0\ [t_1\rangle\ M_1\ [t_2\rangle\ M_2$, pak $M_0 + \underline{t_1} + \underline{t_2} = M_2$ a tedy $M_0 + \underline{N}.c_{t_1} + \underline{N}.c_{t_2} = M_0 + (c_{t_1} + c_{t_2}) = M_2$. Obecně, tudíž, pro $M_0\ [t_1\rangle\ M_1\ [t_2\rangle\ \ldots\ [t_k\rangle\ M_k$ dostáváme:

$$M_n = M_0 + \sum_{i=1}^{k} t_i = M_0 + \sum_{i=1}^{k} \underline{N}.c_{t_i} = M_0 + \underline{N} \sum_{i=1}^{k} c_{t_i} = M_0 + \underline{N}.u$$

Tento důležitý fakt vyjádříme následující větou.

■ **Věta 9.5** Věta

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť a nechť $M_0, M_1, \ldots, M_k \in [M_0\rangle$ a $t_1, t_2, \ldots, t_k \in T$, přičemž

$$M_0\ [t_1\rangle\ M_1\ [t_2\rangle\ \ldots\ [t_k\rangle\ M_k$$

Nechť vektor $u : T \to \mathbb{N}$ je definován takto:

$$u(t) = |\{i : t_i = t \wedge 1 \leq i \leq k\}|$$

Pak

$$M_0 + \underline{N}.u = M_k$$

□

Opak věty 9.5 obecně neplatí, protože pro provedení výpočetní posloupnosti odpovídající vektoru u je třeba dostatečného počtu značek a dostatečně volné kapacity míst.

■ **Věta 9.6** Věta

Nechť N je Petriho síť $N = (P, T, F, W, K, M_0)$, pro kterou $K(p) = \omega$ pro všechna $p \in P$. Nechť $M, M' : P \to \mathbb{N}$ jsou dvě značení a nechť $u : T \to \mathbb{N}$ je vektor. Pak $M + \underline{N}.u = M'$, jestliže existuje $M'' : P \to \mathbb{N}$ a přechody $t_1, t_2, \ldots, t_k \in T$ takové,že $(M+M'')\ [t_1\rangle\ \ldots\ [t_k\rangle\ (M'+M'')$ a pro všechna $t \in T$ je $u(t) = |i : t_i = t \wedge 1 \leq i \leq k\}|$.

Důkaz.

1. „⇐“: Je tvrzením věty 9.5.

2. „⇒“: Provedeme indukcí pro $j = \sum_{i=1}^{k} u(t_i)$.

   Nechť $j = 0$, pak $M = M'$. Dokazované tvrzení platí pro libovolné značení M'', protože $M + M'' \ [\emptyset\rangle \ M' + M''$.

   Nyní předpokládejme, že tvrzení platí pro $j-1$ a nechť $t \in T$ je přechod, pro který $u = u' + c_t$. Pak $\sum_{i=1}^{k} u'(t_i) = j - 1$.

   Máme $M + \underline{N}.u = M'$. Dále nechť $M''' = M' - \underline{t}$. Pak

   $$M+\underline{N}.u' = M+\underline{N}.(u-c_t) = M+\underline{N}.u-\underline{N}.c_t = M+\underline{N}.u-\underline{t} = M'-t = M'''$$

   Podle induktivní hypotézy existuje $k \in \mathbb{N}$, určité značení M'' a výpočetní posloupnost $t_1, t_2, \ldots t_k$ tak, že $(M+M'') \ [t_1\rangle \ \ldots \ [t_k\rangle \ (M'''+M'')$, kde $u'(t) = |\{i : t_i = t \wedge 1 \leq i \leq k\}|$.

   Nyní nechť značení $\widehat{M} : P \to \mathbb{N}$ je definováno takto:

   $$\widehat{M}(p) = \begin{cases} W(p,t), & \text{pro } p \in {}^\bullet t \\ 0, & \text{jinak} \end{cases}$$

   Pak $\widehat{M} \ [t\rangle \ \widehat{M} + \underline{t}$ a $(M + M'' + \widehat{M}) \ [t_1\rangle \ \ldots \ [t_k\rangle \ (M''' + M'' + \widehat{M}) \ [t_{k+1}\rangle \ (M'''+\underline{t}+M''+\widehat{M})$, $t_{k+1} = t$, protože $\forall p \in P : K(p) = \omega$. Tedy,

   $$M''' + \underline{t} = M'' \text{ a } \forall t \in T : u(t) = |\{i : t_i = t \wedge 1 \leq i \leq k+1\}|$$

□ □

Dále se budeme zabývat vztahem mezi řešením soustavy $\underline{N}.x = \mathbf{0}$ a vlastností nazývanou reprodukovatelnost značení.

DEF

▪ **Definice 9.3** *Reprodukovatelnost značení.*

Značení M Petriho sítě N se nazývá *reprodukovatelné*, jestliže existuje $M' \neq M$ tak, že $M' \in \ [M\rangle$ a zároveň $M \in \ [M'\rangle$. □

Lemma

▪ **Lemma 9.2**

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť, pro kterou $\forall p \in P : K(p) = \omega$. Je-li značení M sítě N reprodukovatelné, pak je rovněž reprodukovatelné značení $M + M'$ pro libovolné značení M' sítě N.

Důkaz. Je-li M reprodukovatelné, pak $M\ [\alpha\rangle\ M$ pro určitou výpočetní posloupnost $\alpha \in T^+$. Poněvadž $\forall p \in P : K(p) = \omega$, platí $M + M'\ [\alpha\rangle\ M + M'$ pro libovolné značení $M' : P \to \mathbb{N} \cup \{\omega\}$. □ □

DEF

■ **Definice 9.4** *T-invariant.*

Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť. Vektor $i : T \to \mathbb{Z}$ se nazývá *T-invariant* sítě N, jestliže $\underline{N}.i = \mathbf{0}$. □

Lemma

■ **Lemma 9.3** *Součet a násobek T-invariantu.*

Nechť i_1 a i_2 jsou T-invarianty sítě N a nechť $z \in \mathbb{Z}$. Pak $i_1 + i_2$ a $z.i_1$ jsou také T-invarianty sítě N.

Důkaz. Je analogický k důkazu lemmatu 9.1. □ □

Věta

■ **Věta 9.7**

Nechť N je Petriho síť s neomezenými kapacitami všech míst. Síti N přísluší nenulový T-invariant i právě tehdy, má-li reprodukovatelné značení.

Důkaz. $\underline{N}.i = \mathbf{0} \Leftrightarrow \mathbf{0} + \underline{N}.i = \mathbf{0} \Leftrightarrow \exists t_1, t_2, \ldots t_k \in T$ a M'' takové, že $(\mathbf{0}+M'')\ [t_1\rangle\ \ldots\ [t_k\rangle\ (\mathbf{0}+M'')$ a $\forall t \in T : i(t) = |\{i : t_i = t \wedge 1 \leq i \leq k\}|$ (věta 9.6). □ □

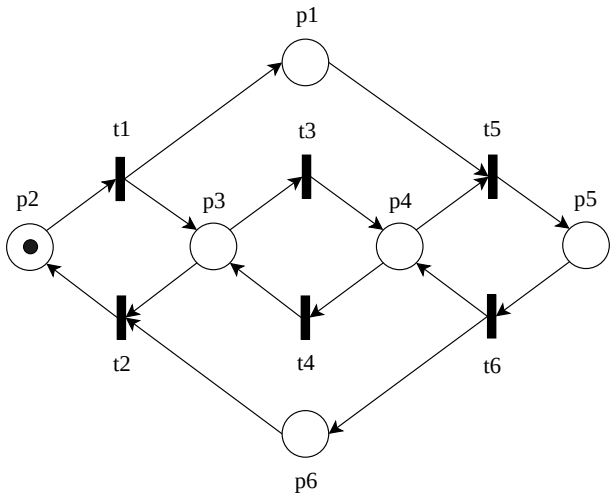
DEF

■ **Definice 9.5** *Realizovatelnost T-invariantu.*

T-invariant i Petriho sítě N se nazývá *realizovatelný*, jestliže existuje $M \in [M_0\rangle$ a výpočetní posloupnost $M\ [t_1\rangle\ \ldots\ [t_k\rangle\ M_n$ taková, že $\forall t \in T : i(t) = |\{i : t_i = t \wedge 1 \leq i \leq k\}|$. □

Ne každý kladný T-invariant i je realizovatelný. Dokonce ani nepostačuje, aby N byla živá a omezená a každé značení bylo reprodukovatelné a invariant i nebyl součtem jiných kladných T-invariantů. Na obrázku 9.6 je příklad takové sítě. T-invariant i definovaný zobrazením $i(t_1) = i(t_2) = i(t_5) = i(t_6) = 1$ a $i(t_3) = i(t_4) = 0$ není realizovatelný.

Na závěr ukážeme, že živá a omezená Petriho síť je pokryta T-invarianty.



Obrázek 9.6: Petriho síť s nerealizovatelným T-invariantem

DEF

■ **Definice 9.6** *Pokrytí sítě T-invarianty.*

Petriho síť N je pokryta T-invarianty, jestliže pro každý přechod t sítě N existuje nezáporný T-invariant i sítě N takový, že $i(t) > 0$. □

Věta

■ **Věta 9.8** *Kladný T-invariant.*

Je-li Petriho síť N pokryta T-invarianty, pak existuje invariant i sítě N takový, že $i(t) > 0$ pro všechny přechody $t \in T$.

Důkaz. Podle definice 9.6 pro každý přechod $t \in T$ existuje T-invariant i_t sítě N takový, že $i_t(t) > 0$. Invariant $i = \sum_{t \in T} i_t$, získaný aplikací lemmatu 9.3, je hledaným T-invariantem. □ □

Věta

■ **Věta 9.9** *T-invarianty v živých omezených sítích.*

Každá živá a omezená Petriho síť je pokryta T-invarianty.

Důkaz. Je-li $N = (P, T, F, W, K, M_0)$ živá a omezená Petriho síť, pak existuje značení $M \in [M_0\rangle$ a výpočetní posloupnost $M\ [t'_1\rangle\ M_1\ [t'_2\rangle\ \ldots\ [t'_k\rangle\ M_k$, ve které byl každý přechod $t \in T$ proveden alespoň jedenkrát a kde $M_k = M$. Definujme vektor $u : T \to \mathbb{N}$ takto:

$$u(t) = |\{i : t'_i = t \wedge 1 \leq i \leq k\}|$$

Podle věty 9.5 platí $M + \underline{N}.u = M_k$, a protože $M = M_k$, je $\underline{N}.u = \mathbf{0}$. Vektor u je tedy T-invariant, a poněvadž navíc $\forall t \in T : u(t) > 0$, u je T-invariant, který pokrývá Petriho síť N. □ □

Věta 9.9 představuje pouze nutnou podmínku pro živost omezené Petriho sítě. Její význam pro analýzu je zřejmý: není-li daná Petriho síť pokryta T-invarianty, pak není živá nebo není omezená.

Pojmy k zapamatování

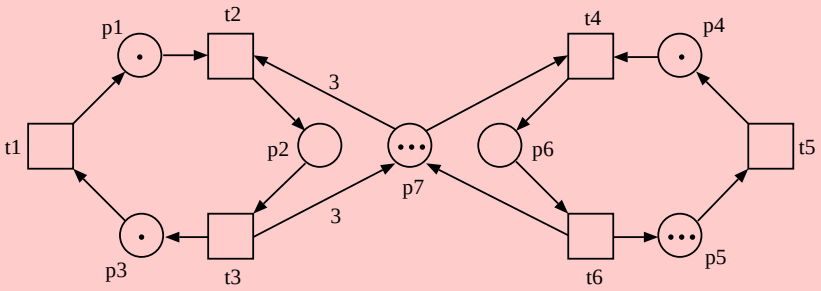
- P-invariant, pokrytí sítě P-invarianty
- T-invariant, pokrytí sítě T-invarianty
- reprodukovatelnost značení, realizovatelnost T-invariantu

Shrnutí

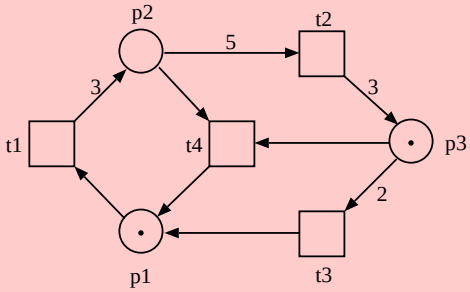
Tato kapitola si kladla za cíl uvést do metod analýzy založené na lineární algebraické reprezentaci Petriho sítě. Nejdříve jsme uvažovali třídu P-invariantů, které zachovávají celkový součet značek po libovolném provedení přechodu. Koncept P-invariantů souvisí do značné míry s již zavedeným pojmem Petriho sítě konzervativní vzhledem k váhovému vektoru. Vedle toho T-invarianty udávají kolikrát je třeba, počínaje určitým značením, provést každý přechod sítě, abychom získali nazpět toto značení.

Příklady k procvičení

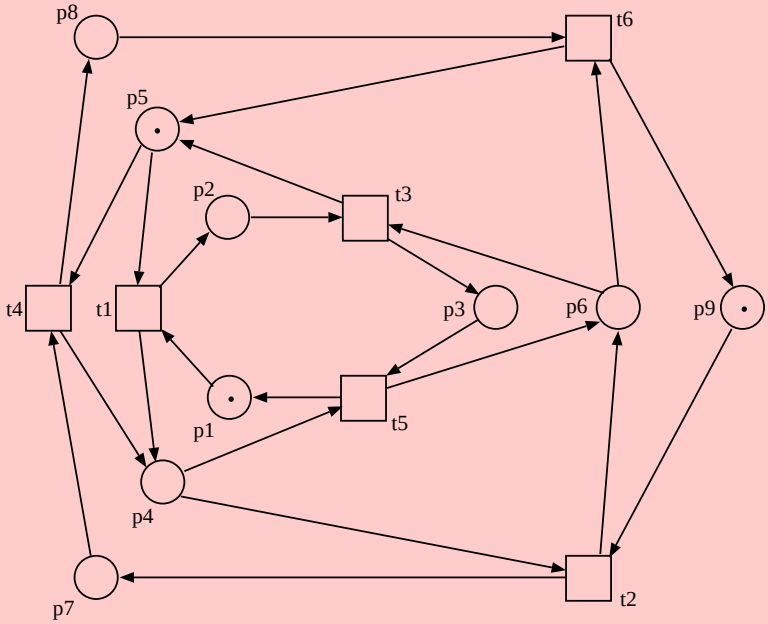
1. Vypočtěte P-invarianty Petriho sítě z obrázku. Je tato síť pokryta P-invarianty?



2. Ukažte, že Petriho síť z obrázku není pokryta P-invarianty.



3. Ukažte, že Petriho síť z obrázku má T-invarianty, které nejsou realizovatelné.



Kapitola 10

Jazyky Petriho sítí

Čas potřebný ke studiu: 13 hodin

Cíle kapitoly

Jak jsme se dozvěděli v předchozích kapitolách, výpočetní posloupnost Petriho sítě je posloupnost *přechodů*, jejichž postupným provedením se z počátečního značení dostáváme do dalších a dalších značení Petriho sítě. Aby však mohl být daný přechod proveden, musí být splněny jisté *podmínky* kladené na okolí tohoto přechodu, tedy na váhy hran vstupujících a vystupujících, značení míst v presetu a postsetu přechodu a kapacity postsetu přechodu.
Přiřadíme-li jednotlivým přechodům symboly, pak posloupnosti symbolů odpovídající daným výpočetním posloupnostem Petriho sítě tvoří jejich jazyk, neboli množinu vět (řetězců) tohoto jazyka.
Tato kapitola definuje způsoby, jakými lze k jednotlivým přechodům přiřadit symboly jazyka, a u jednotlivých způsobů studuje dopad na mohutnost třídy jazyků generované Petriho sítěmi.
Obdobně jsou zde specifikovány rozličné způsoby určení, kterou výpočetní posloupnost zahrneme do množiny vět jazyka a kterou již ne (pomocí definice *koncových stavů*) a opět u těchto způsobů studujeme dopad na mohutnost třídy jazyků.
Na příkladě jedné třídy jazyků Petriho sítí jsou ukázány jejich uzávěrové vlastnosti a zařazení do Chomského hierarchie jazyků.

Průvodce studiem

Ke studiu této kapitoly budete potřebovat nejen znalosti z kapitol 7 a 8, ale rovněž základní znalosti z teorie formálních jazyků a porozumění Chomského hierarchii jazyků.

Obsah

V kapitole 7 jsme uvedli koncept jazyka generovaného Petriho sítí jako množiny všech výpočetních posloupností, které mohou být v dané Petriho síti provedeny. Tato množina představuje jednu z nejdůležitějších charakteristik modelovaného systému. Proto také tvoří často formální základ pro stanovení ekvivalence Petriho sítí; dvě Petriho sítě jsou *ekvivalentní*, jestliže jsou totožné jejich jazyky. Z této ekvivalence pak vycházejí mnohé optimalizace nebo transformace Petriho sítí.

Důležité je rovněž využití jazyků Petriho sítí při syntéze systémů, jejichž modelem je Petriho síť s požadovaným chováním. Existují postupy, jak lze automaticky získat kompozicí elementárních Petriho sítí výslednou Petriho síť, jejíž chování bylo specifikováno určitým formálním jazykem. Tyto postupy jsou podobné automatické syntéze konečného automatu na základě specifikovaného regulárního výrazu.

Ne poslední z motivací studia jazyků Petriho sítí je jejich začlenění do Chomského hierarchie formálních jazyků. Jejich „pozice“ v této hierarchii nám ukáže, jakou modelovací a rozhodovací sílu příslušných modelů můžeme v různých problémech spojených s teorií formálních jazyků očekávat.

10.1 Definice jazyků Petriho sítí

Jak již jsme několikrát zdůraznili, Petriho sítě vznikly zobecněním konečných automatů. Tato skutečnost se projevuje i v definici jazyků Petriho sítí. V analogii s jazyky přijímanými konečnými automaty je účelné zavést i pro Petriho síť pojmy vstupní abeceda, počáteční stav a množina koncových stavů. Připomeňme, že přechodová funkce Petriho sítě již byla definována v kapitole 7.

10.1.1 Abeceda a ohodnocení Petriho sítě

Dosud jsme předpokládali, že symboly řetězců reprezentujících výpočetní posloupnosti jsou shodné s označeními přechodů Petriho sítě (s prvky množiny T). To nám sice umožňuje jednoznačně popisovat posloupnosti provádění přechodů sítě, na druhé straně však nemůžeme, vzhledem k sémantice modelovaných operací, vyjádřit takovou situaci, kdy dva různé přechody sítě popisují stejnou operaci nebo situaci, kdy přechod modeluje určitou vnitřní (pomocnou) operaci, kterou nechceme ve větě jazyka zobrazovat. Z tohoto důvodu zavedeme, vedle množiny přechodů T, *abecedu* Σ *Petriho sítě* a zobrazení $\lambda : T \to \Sigma \cup \{\epsilon\}$, které každému přechodu sítě přiřadí symbol abecedy Σ nebo prázdný symbol ϵ. Zobrazení λ budeme nazývat *ohodnocením přechodů* (angl. *labeling*) a příslušnou Petriho síť *ohodnocenou Petriho*

sítí.

Podle ohodnocení přechodů (podle tvaru zobrazení λ) se v literatuře rozlišují tři typy ohodnocených Petriho sítí:

1. Nejomezenějším typem jsou sítě s injektivním ohodnocením $\lambda : T \to \Sigma$, t.j.
$$\lambda(t) = \lambda(t') \Rightarrow t = t'$$
označované jako *free-labeled Petri nets.*

2. Obecnější typ ohodnocení $\lambda : T \to \Sigma$ připouští, aby dva různé přechody byly ohodnoceny stejným symbolem abecedy, avšak nepřipouští ohodnocení přechodu prázdným symbolem.

3. Třetí typ je libovolné ohodnocení tvaru $\lambda : T \to \Sigma \cup \{\epsilon\}$.

10.1.2 Počáteční stav a počáteční místo

Již víme, že množinu stavů Petriho sítě tvoří množina $[M_0\rangle$; počáteční stav zřejmě odpovídá počátečnímu značení M_0. Pro určité operace s Petriho sítěmi je někdy vhodné, aby počáteční stav sítě byl spojen se značením jediného místa – *počátečního (startovacího) místa* p_s tak, že
$$M_0(p_\mathsf{s}) = 1 \wedge \forall p \in P, p \neq p_\mathsf{s} : M_0(p) = 0$$
Ukážeme, že zavedení počátečního místa nevylučuje, aby fakticky libovolné značení Petriho sítě mohlo být počátečním značením. Uvažujme libovolnou Petriho síť $N = (P, T, F, W, K, M_0)$. Ekvivalentní síť $N' = (P', T', F', W', K', M'_0)$ s počátečním místem p_s vytvoříme takto:

1. $P' = P \cup \{p_\mathsf{s}\}$, kde $p_\mathsf{s} \notin P$ je nově přidané počáteční místo

2. $T' = T \cup \{t_\mathsf{s}\}$, kde $t_\mathsf{s} \notin T$ je nově přidaný přechod

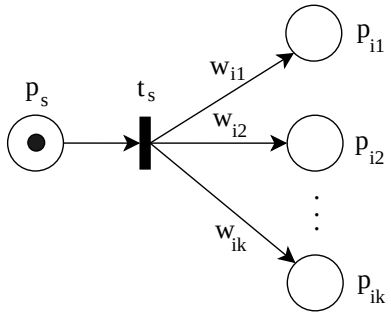
3. $F' = F \cup F_{t_\mathsf{s}}$, kde $F_{t_\mathsf{s}} = \{\langle p_\mathsf{s}, t_\mathsf{s}\rangle\} \cup \{\langle t_\mathsf{s}, p\rangle | M_0(p) \neq 0\}$

4. W' je rozšíření váhové funkce W: $W'(p_\mathsf{s}, t_\mathsf{s}) = 1 \wedge W'(t_\mathsf{s}, p) = k \Leftrightarrow M_0(p) = k, k \in \mathbb{N} \setminus \{0\}$

5. K' je rozšíření K: $K'(p_\mathsf{s}) = 1$

6. $M'_0 : P \cup \{p_\mathsf{s}\} \to \mathbb{N}, M'_0(p_\mathsf{s}) = 1, \forall p \in P : M'_0(p) = 0$

Situaci ilustruje obrázek 10.1. Na počátku je proveditelný pouze přechod t_s.Jeho provedením získáme značení M'_0, kde $M'_0(p) = M_0(p)$ pro všechna $p \in P$. Množiny výpočetních posloupností sítí N a N' se liší pouze tím, že každá výpočetní posloupnost sítě N' začíná symbolem t_s; pokud definujeme ohodnocení $\lambda(t_\mathsf{s}) = \epsilon$ a $\forall t \in T : \lambda(t) = t$, pak jsou tyto množiny shodné.



Obrázek 10.1: Počáteční místo Petriho sítě

10.1.3 Koncové stavy Petriho sítě

Definice koncových stavů má největší vliv na jazyky specifikované Petriho sítěmi. Existují čtyři různé definice, které vymezují třídy jazyků označované jako jazyky typu **L**, **G**, **T** a **P**. První třída jazyků využívá k definici množinu koncových stavů tak, jak ji známe z konečných automatů.

DEF

■ **Definice 10.1** *Jazyky typu* **L**.

Nechť N je Petriho síť s počátečním značením M_0, s ohodnocením přechodů $\lambda : T \rightarrow \Sigma \cup \{\epsilon\}$, se (zobecněnou) přechodovou funkcí δ (definice 7.11)

$$\delta : [M_0\rangle \times T^* \rightarrow [M_0\rangle$$

a s množinou koncových stavů $Q_\mathsf{F} \subseteq [M_0\rangle$. Jazyk $L(N) \subseteq \Sigma^*$ Petriho sítě definovaný jako

$$L(N) = \{\lambda(\alpha) | \alpha \in T^* \wedge \delta(M_0), \alpha) \in Q_\mathsf{F}\}$$

se nazývá *jazykem typu* **L**. □

Třída jazyků typu **L** je bohatá a mocná. Mechanismus generování vět dosažením přesně určeného značení (koncového stavu) však není zcela v souladu se základní filozofií Petriho sítě, speciálně s pravidly provádění přechodů sítě. Skutečně, je-li přechodová funkce $\delta(M, t)$ definována pro značení M, pak je definována rovněž $\delta(M', t)$, pro každé $M' \geq M$. Tento nesoulad odstraňuje definice jazyků typu **G**.

DEF

■ **Definice 10.2** *Jazyky typu* **G**.

Nechť N je Petriho síť s počátečním značením M_0, s ohodnocením přechodů λ, přechodovou funkcí δ a s množinou koncových stavů Q_F.

Jazyk $L(N)$ Petriho sítě N definovaný jako

$$L(N) = \{\lambda(\alpha) | \alpha \in T^* \wedge \exists M_{\mathsf{F}} \in Q_{\mathsf{F}} : \delta(M_0, \alpha) \geq M_{\mathsf{F}}\}$$

se nazývá *jazykem typu* **G**. □

Třetí typ jazyků Petriho sítí je definován na základě absolutizovaného pojetí koncového stavu. Značení M_{F} je koncovým stavem právě tehdy, jestliže funkce $\delta(M_{\mathsf{F}}, t)$ je nedefinována pro každý přechod t. Naproti tomu poslední, čtvrtý typ jazyků Petriho sítí je odvozen z předpokladu, že každé dosažitelné značení může reprezentovat koncový stav. Tyto odlišné přístupy vymezuje následující definice.

DEF

■ **Definice 10.3** *Jazyky typu* **T** *a* **P**.

Nechť N je Petriho síť s počátečním značením M_0, s ohodnocením přechodů λ a přechodovou funkcí δ.

1. Jazyk $L(N)$ Petriho sítě N definovaný jako
   $L(N) = \{\lambda(\alpha) | \alpha \in T^* \wedge \delta(M_0, \alpha) \in [M_0\rangle \wedge \delta(\delta(M_0, \alpha), t)$
   je nedefinováno $\forall t \in T\}$
   se nazývá *jazykem typu* **T**.

2. Jazyk $L(N)$ Petriho sítě N definovaný jako

   $$L(N) = \{\lambda(\alpha) | \alpha \in T^* \wedge \delta(M, \alpha) \in [M_0\rangle\}$$

   se nazývá *jazykem typu* **P**.

□

Jazyk typu **P** je tzv. *prefixovým jazykem*. Je-li $x \in L(N)$ a $x = yz$ pak také $y \in L(N)$.

Uvážíme-li nyní, že pro každou z tříd **L**, **G**, **T** a **P** mohou být vymezeny tři třídy jazyků Petriho sítí podle typu ohodnocení λ, dostáváme dvanáct specifických tříd podle schématu na obrázku 10.2. Mezi tě-

	$\lambda : T \to \Sigma$ λ injektivní (free labeled)	$\lambda : T \to \Sigma$ bez ϵ-přechodů	$\lambda : T \to \Sigma \cup \{\epsilon\}$ s ϵ-přechody
L-typ	$\mathbf{L}^{\mathsf{f}}$	**L**	$\mathbf{L}^{\epsilon}$
G-typ	$\mathbf{G}^{\mathsf{f}}$	**G**	$\mathbf{G}^{\epsilon}$
T-typ	$\mathbf{T}^{\mathsf{f}}$	**T**	$\mathbf{T}^{\epsilon}$
P-typ	$\mathbf{P}^{\mathsf{f}}$	**P**	$\mathbf{P}^{\epsilon}$

Obrázek 10.2: Třídy jazyků

mito třídami jazyků existují evidentně vzájemné vztahy vyjádřitelné množinovou inkluzí. Z klasifikace jazyků podle typu ohodnocení λ je např. zřejmé, že platí :

$$\mathbf{L}^{\mathsf{f}} \subseteq \mathbf{L} \subseteq \mathbf{L}^{\epsilon}$$

$$\mathbf{G}^{\mathsf{f}} \subseteq \mathbf{G} \subseteq \mathbf{G}^{\epsilon}$$

$$\mathbf{T}^{\mathsf{f}} \subseteq \mathbf{T} \subseteq \mathbf{T}^{\epsilon}$$

$$\mathbf{P}^{\mathsf{f}} \subseteq \mathbf{P} \subseteq \mathbf{P}^{\epsilon}$$

Na obrázku 10.3 jsou znázorněny vzájemné vztahy těchto tříd jazyků tak, jak jsou uvedeny a diskutovány v [9]. Orientovaná hrana z A do B vyjadřuje inkluzi $A \supseteq B$.

$$\begin{array}{ccccc}
\mathbf{T}^{\mathsf{f}} & \leftarrow & \mathbf{T} & \leftarrow & \mathbf{T}^{\epsilon} \\
 & & \uparrow & & \updownarrow \\
\mathbf{L}^{\mathsf{f}} & \leftarrow & \mathbf{L} & \leftarrow & \mathbf{L}^{\epsilon} \\
 & & \downarrow & & \downarrow \\
\mathbf{G}^{\mathsf{f}} & \leftarrow & \mathbf{G} & \leftarrow & \mathbf{G}^{\epsilon} \\
\downarrow & & \downarrow & & \downarrow \\
\mathbf{P}^{\mathsf{f}} & \leftarrow & \mathbf{P} & \leftarrow & \mathbf{P}^{\epsilon}
\end{array}$$

Obrázek 10.3: Vztahy tříd jazyků

10.2 Vlastnosti jazyků Petriho sítí typu L

V této sekci se omezíme pouze na jazyky typu **L**. Tato třída, jak ukazuje obrázek 10.3, je dostatečně obecná a byla z hlediska vlastností nejvíce studována a analyzována. Zaměříme se na dva okruhy problémů; na uzávěrové vlastnosti vzhledem k řadě operací nad jazyky a dále na vztah této třídy k Chomského hierarchii formálních jazyků.

10.2.1 Standardní tvar Petriho sítě

Abychom zjednodušili některé další postupy, zavedeme nejdříve určitou standardizovanou formu ohodnocené Petriho sítě, na kterou může být každá jiná Petriho síť převedena. Ve snaze zredukovat počet složek definice ohodnocení sítě nebudeme dále uvádět kapacitu míst K a budeme předpokládat, že všechna místa mají neomezenou kapacitu (viz. sekce 7.2).

DEF

■ **Definice 10.4** *Petriho síť ve standardním tvaru.*

Petriho síť $N = (P, T, F, W, M_0, p_\mathsf{s}, \Sigma, \lambda, P_\mathsf{F}, Q_\mathsf{F})$ nazveme *ohodnocenou Petriho sítí ve standardním tvaru*, jestliže

1. Složky P, T, F, W, M_0 a Q_F mají dosud užívaný význam
2. $p_\mathsf{s} \in P$ je počáteční místo takové, že
   (a) $M_0(p_\mathsf{s}) = 1 \wedge \forall p \in P \setminus \{p_\mathsf{s}\} : M_0(p) = 0$
   (b) $\forall t \in T : \langle t, p_\mathsf{s}\rangle \notin F$
3. $\lambda : T \to \Sigma$ je ohodnocení přechodů sítě
4. P_F je množina koncových míst, pro níž platí
   (a) $P_\mathsf{F} \subseteq P$
   (b) $p_\mathsf{s} \in P_\mathsf{F} \Leftrightarrow \epsilon \in L(N)$
   (c) $\forall t \in T, \forall p_\mathsf{F} \in P_\mathsf{F} \setminus \{p_\mathsf{s}\} : \langle p_\mathsf{F}, t\rangle \notin F$
   (d) Je-li $M(p_\mathsf{F}) > 0$ pro nějaké $M \in [M_0\rangle$, pak $\delta(M, t)$ je nedefinována pro všechna $t \in T$.

□

Petriho síť ve standardním tvaru zjednodušuje provádění sítě a rozhodování, zda generovaný řetězec je prvkem jazyka $L(N)$. Skutečně, provádění sítě končí, jakmile koncové místo získá značku (v tom případě již není proveditelný žádný přechod). Pokud nyní dosažené značení je značením koncovým, pak odpovídající ohodnocená výpočetní posloupnost je prvkem jazyka $L(N)$. Tento test na koncové značení je samozřejmě nutný, protože po ukončení evoluce sítě, může být dosaženo značení M, které pokrývá koncové značení, avšak koncovým značením není. V tom případě odpovídající posloupnost nepatří do $L(N)$. Velmi často se jako koncové značení (viz důkaz následující věty a příklady tohoto odstavce) užívá jediné koncové značení, které má, s výjimkou složky $M(p_\mathsf{F})$, všechny složky nulové. Pokud $\epsilon \in L(N)$, pak je koncovým značením také značení počáteční.

Věta

■ **Věta 10.1** *Existence sítě ve standardním tvaru.*

Ke každé ohodnocené Petriho síti N (typu **L**) existuje ekvivalentní ohodnocená Petriho síť N' ve standardním tvaru tak, že $L(N) = L(N')$.

Důkaz. Ukážeme konstrukci sítě N'. Nechť N je Petriho síť typu **L**, $N = (P, T, F, W, M_0, p_\mathsf{s}, \Sigma, \lambda, Q_\mathsf{F})$, kde $Q_\mathsf{F} \subseteq [M_0\rangle$.

Pro síť $N' = (P', T', F', W', M_0', p', \Sigma', \lambda', P_\mathsf{F})$ definujeme její složky takto:

1. $P' = P \cup \{p_\mathsf{s}, p_\mathsf{F}, p_r\}$, kde p_s, p_F a p_r jsou nová místa reprezentující počáteční, koncové a pomocné místo (run place) sítě N'.

2. Počáteční značení M_0' má složky: $M_0'(p_\mathsf{s}) = 1$ a $M_0'(p) = 0$ pro všechna $p \in P' \setminus \{p_\mathsf{s}\}$

3. Množina koncových míst $P_\mathsf{F} = \begin{cases} \{p_\mathsf{F}\}, & \text{jestliže } M_0 \notin Q_\mathsf{F} \\ \{p_\mathsf{s}, p_\mathsf{F}\}, & \text{jestliže } M_0 \in Q_\mathsf{F} \end{cases}$

4. Nejdříve vytvoříme nové přechody pro generování řetězců délky 1. Tyto řetězce lze snadno nalézt:

$$x \in L(N) \wedge |x| = 1 \Leftrightarrow \exists t \in T : \lambda(t) = x \wedge \delta(M_0, t) \in Q_\mathsf{F}$$

   (a) Pro všechna $x \in L(N)$, $|x| = 1$ (tedy $x \in \Sigma$) položíme

      i. $t'_x \in T'$ je nový přechod
      ii. $\lambda'(t'_x) = x$
      iii. $\{\langle p_\mathsf{s}, t'_x\rangle, \langle t'_x, p_\mathsf{F}\rangle\} \subseteq F'$
      iv. $W'(p_\mathsf{s}, t'_x) = W'(t'_x, p_\mathsf{F}) = 1$

   Dále budeme uvažovat věty délky větší než 1.

   (b) Nechť $x \in L(N)$ a $|x| > 1$ a nechť tento řetězec je generován výpočetní posloupností $t_{i_1} t_{i_2} \ldots t_{i_k} \in T^+$ (tj. $x = \lambda(t_{i_1} t_{i_2} \ldots t_{i_k})$. Tuto výpočetní posloupnost sítě N bude emulovat výpočetní posloupnost $a t'_{i_1} t'_{i_2} \ldots t'_{i_k} b$ sítě N', jejíž přechody jsou definovány následovně:

      i. Položíme: $\langle p_\mathsf{s}, a\rangle \in F'$, $W'(p_\mathsf{s}, a) = 1$. Pro všechna $p \in P$, pro která $M_0(p) > 0$ bude $\langle a, p\rangle \in F'$ a $W'(a, p) = M_0(p)$. Dále $\langle a, p_r\rangle \in F'$ a $W'(a, p_r) = 1$.
      ii. Nechť $\delta(M_0, t_{i_1} t_{i_2} \ldots t_{i_k}) = M_\mathsf{F}$, kde $M_\mathsf{F} \in Q_\mathsf{F}$. Pro všechna $p \in P$, pro která $M_\mathsf{F}(p) > 0$ položíme: $\langle p, b\rangle \in F'$ a $W'(p, b) = M_\mathsf{F}(p)$. Dále $\langle p_r, b\rangle \in F'$, $W'(p_r, b) = 1$ a $\langle b, p_\mathsf{F}\rangle \in F'$, $W'(b, p_\mathsf{F}) = 1$. Přechody a a b (stejně jako všechny ostatní přechody z T') jsou spojeny s místem p_r vlastní smyčkou, což zaručuje, že po odstranění značky z místa p_r se stanou všechny přechody sítě N' neproveditelné.

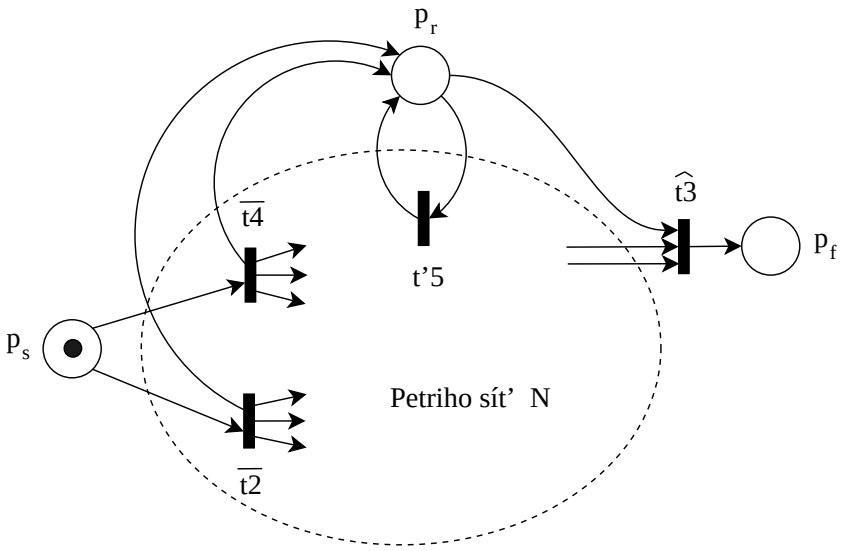
iii. Pro všechna $t \in T$ definujeme $t' \in T'$ takto:

$$^\bullet t' = {}^\bullet t \cup \{p_r\}$$

$$t'^\bullet = t^\bullet \cup \{p_r\}$$

$$\lambda'(t') = \lambda(t)$$

Pokud bychom nyní položili $\lambda'(a) = \lambda'(b) = \epsilon$, pak již získáváme síť N' takovou, že $L(N) = L(N')$. Protože N je však sítí typu $\mathbf{L}$ a ne $\mathbf{L}^\epsilon$, nelze přechody a a b ohodnotit prázdným symbolem. Pokusíme se proto o jiné řešení, spočívající ve sloučení dvojic přechodů (a, t'_{i_1}), resp. (t'_{i_k}, b) do jediného přechodu $\overline{t_{i_1}}$, resp. $\widehat{t_{i_k}}$ tak, jak to ilustruje obrázek 10.4.



Obrázek 10.4: Konstrukce Petriho sítě ve standardním tvaru

(c) Popíšeme konstrukci přechodů typu $\overline{t}$ slučujících dvojice přechodů (a, t'_{i_1}). Pro všechny přechody $t \in T$ postupujeme takto:

Jestliže platí $\forall p \in {}^\bullet t : M_0(p) \geq W(p,t)$ (tj. $\delta(M_0, t)$ je definováno a t je prvním přechodem nějaké výpočetní posloupnosti sítě N), pak k přechodu t definujeme přechod $\overline{t} \in T'$:

i. ${}^\bullet\overline{t} = \{p_s\}$ a $W(p_s, \overline{t}) = 1$

ii. Označme ${}^\bullet\tau = \{p | (M_0(p) - W(p,t) > 0\}$. Zřejmě ${}^\bullet\tau \subseteq {}^\bullet t$. Pak položíme $\overline{t}^\bullet = t^\bullet \cup {}^\bullet\tau \cup \{p_r\}$, přičemž

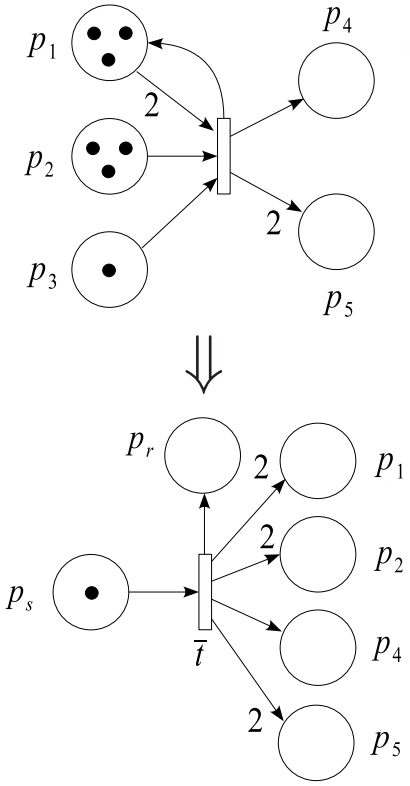
$$W'(\overline{t}, p_r) = 1$$

$W'(\overline{t}, p) = M_0(p) - w(p,t) + w(t,p)$, kde

$$w(x,y) = \begin{cases} W(x,y), & \text{je-li } \langle x,y\rangle \in F \\ 0, & \text{v opačném případě} \end{cases}$$

iii. $\lambda'(\overline{t}) = \lambda(t)$

Tuto konstrukci ilustruje obrázek 10.5.



Obrázek 10.5: Konstrukce Petriho sítě ve standardním tvaru - pokračování

(d) Analogicky vytvoříme přechody typu $\widehat{t}$ slučující dvojice (b, t'_{i_k}). Pro všechny přechody $t \in T$ a koncová značení $M_{\mathsf{F}} \in Q_{\mathsf{F}}$ postupujeme takto:

Jestliže platí $\forall p \in t^\bullet : M_{\mathsf{F}}(p) \geq W(t,p)$ pro nějaké $M_{\mathsf{F}} \in Q_{\mathsf{F}}$ (tj. t může být posledním přechodem jisté výpočetní posloupnosti), pak definujeme přechod $\widehat{t} \in T'$:

i. Položíme ${}^\bullet\widehat{t} = {}^\bullet t \cup \{p | M_{\mathsf{F}}(p) > 0\} \cup \{p_r\}$, přičemž:

$W'(p_r, \widehat{t}) = 1$

$W'(p, \widehat{t}) = M_{\mathsf{F}}(p) + w(p,t) - w(t,p)$, kde $w(x,y)$ je definováno stejně jako v předešlém bodě

ii. $\widehat{t}^{\bullet} = \{p_{\mathsf{F}}\}$ a $W(\widehat{t}, p_{\mathsf{F}}) = 1$
iii. $\lambda'(\widehat{t}) = \lambda(t)$

5. Množina koncových stavů je nyní definována takto:

$$Q'_{\mathsf{F}} = \begin{cases} \{M'_{\mathsf{F}}\}, & \text{jestliže } \epsilon \notin L(N) \\ \{M'_{\mathsf{F}}, M'_0\}, & \text{jestliže } \epsilon \in L(N) \end{cases}$$

kde $M'_{\mathsf{F}}(p) = 0$ pro $p \in P' \setminus \{p_{\mathsf{F}}\}$ a $M'_{\mathsf{F}}(p_{\mathsf{F}}) = 1$.

Tím jsme dokončili konstrukci sítě N'. Jestliže nyní řetězec $x \in L(N)$ je generován výpočetní posloupností $t_{i_1} t_{i_2} \ldots t_{i_k}$, $x = \lambda(t_{i_1} t_{i_2} \ldots t_{i_k})$, pak této výpočetní posloupnosti odpovídá právě jedna výpočetní posloupnost $\overline{t}_{i_1} t_{i_2} \ldots \widehat{t}_{i_k}$, která generuje řetězec $\lambda'(\overline{t}_{i_1} t_{i_2} \ldots \widehat{t}_{i_k}) = x$. Tedy $x \in L(N) \Leftrightarrow x \in L(N')$ a tudíž $L(N) = L(N')$ a sítě N a N' jsou ekvivalentní. □ □

10.2.2 Uzávěrové vlastnosti

Existence standardního tvaru Petriho sítě typu **L** účelně využijeme při důkazech a konstrukcích souvisejících s uzávěrovými vlastnostmi této třídy jazyků.

Věta

■ **Věta 10.2** *Uzavřenost vzhledem ke konkatenaci jazyků.*

Nechť L_1 a L_2 jsou dva jazyky generované Petriho sítěmi. Pak jazyk $L = L_1.L_2$ je také jazykem generovaným Petriho sítí.

Důkaz. Připomeňme, že konkatenace jazyků $L_1.L_2$ je jazyk L

$$L = \{xy | x \in L_1 \wedge y \in L_2\}$$

Předpokládejme, že jazyk L_1, resp. jazyk L_2 je generován Petriho sítí N_1, resp. N_2 ve standardním tvaru. Konstrukce sítě N generující jazyk L je jednoduchá, zvláště v případě, kdy $\epsilon \notin L(N_1)$. Pak zřejmě stačí ztotožnit místo $p_{\mathsf{F}1}$ s místem $p_{\mathsf{S}2}$. Množina koncových míst sítě N je rovna $P_{\mathsf{F}1}$. Tuto konstrukci ilustruje obrázek 10.6 v příkladu 10.1.

V případě, že $\epsilon \in L(N_1)$, pak stačí rozšířit spojení obou sítí o určité nové přechody takto:

1. Pro každý přechod $t \in T_2$ takový, že $(p_{\mathsf{S}2}, t) \in F_2$ sestrojíme nový přechod $t' \in T$, pro který:

   (a) $(p_{\mathsf{S}1}, t') \in F$, $W(p_{\mathsf{S}1}, t') = W_2(p_{\mathsf{S}1}, t)$
   (b) $t'^{\bullet} = t^{\bullet} \wedge \forall p \in t^{\bullet} : W'(t', p) = W_2(t, p)$

(c) $\lambda(t') = \lambda_2(t)$

2. $M_0(p_{S1}) = 1, \forall p \in P_1 \cup P_2 \setminus \{p_{S1}\} : M_0(p) = 0$

3. Množina koncových míst výsledné sítě: $P_F = \{p_{S1}\} \cup P_{F2}$

□ □

x+y

■ **Příklad 10.1** *Konkatenace Petriho sítí.*

Na obrázku 10.6 je ohodnocená Petriho síť N_1 generujícíjazyk L_1 popsaný regulárním výrazem $(a + b)^+$, Petriho síť N_2 generující jazyk $L_2 = \{a^n b^n | n \geq 1\}$ a Petriho síť N generující jazyk $L = L_1.L_2$. □

■ **Věta 10.3** *Uzavřenost vzhledem ke sjednocení jazyků.* Věta

Nechť L_1 a L_2 jsou dva jazyky generované Petriho sítěmi. Pak jazyk $L = L_1 \cup L_2$ je jazykem generovaným Petriho sítí.

Důkaz. Nechť jazyk L_1, resp. L_2 je generován Petriho sítí N_1, resp. N_2 ve standardním tvaru. Vytvoříme síť N generující jazyk $L(N_1) \cup L(N_2)$ obdobným postupem jako v předchozím případě. Tentokrát však sloučíme počáteční místa p_{S1} a p_{S2} obou sítí do počátečního místa p_S sítě N. Formálně:

1. Pro každý přechod $t \in T_1$ takový, že ${}^\bullet t = \{p_{S1}\}$ vytvoříme přechod $t' \in T$, pro který:

   (a) ${}^\bullet t' = \{p_S\}$

   (b) $t'^\bullet = t^\bullet \wedge \forall p \in t'^\bullet : W(t', p) = W_1(t, p)$

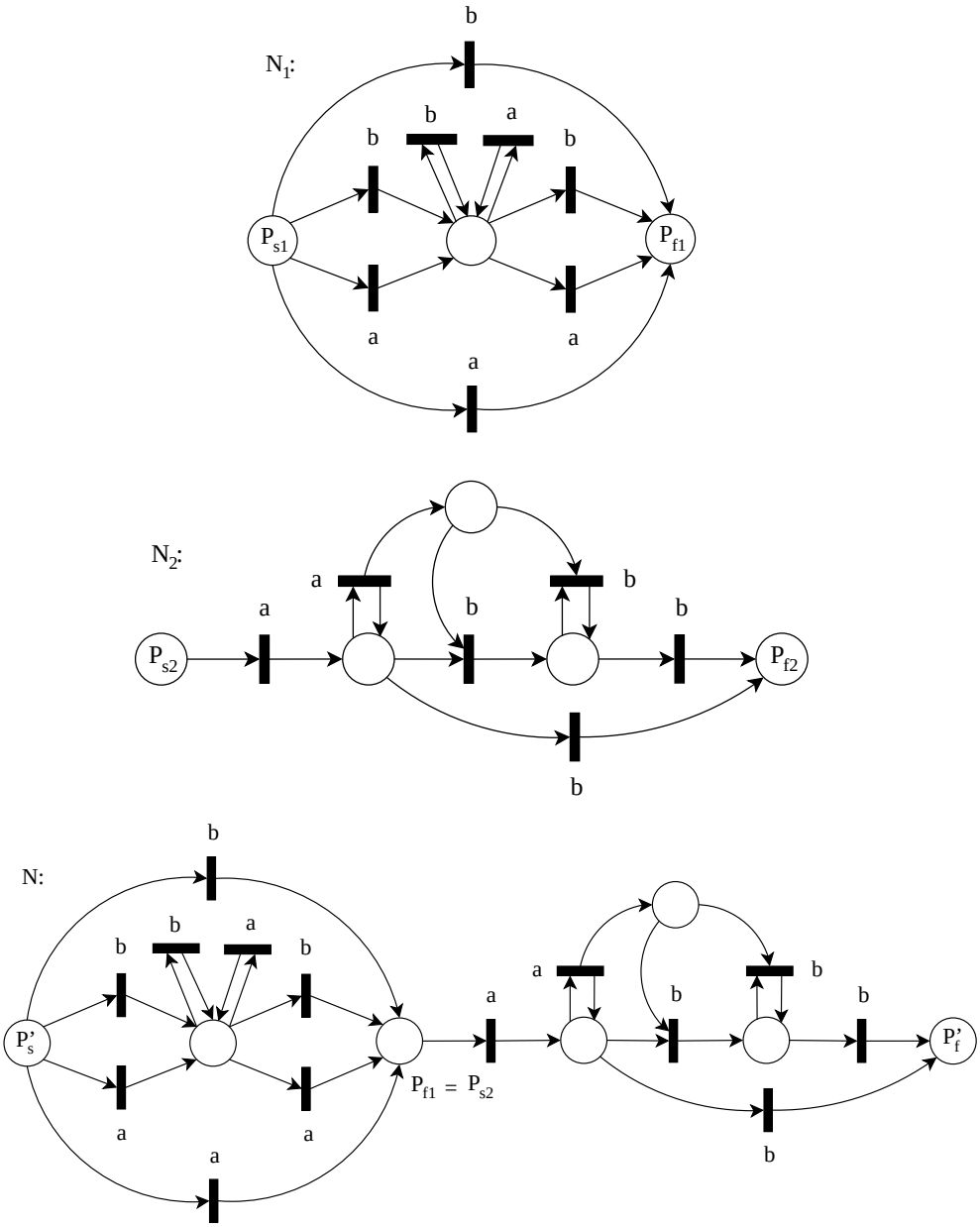
   (c) $\lambda(t') = \lambda_1(t)$

2. Analogicky s předchozím bodem postupujeme pro $t \in T_2$, ${}^\bullet t = \{p_{S2}\}$

3. $M_0(p_S) = 1, \forall p \in P_1 \cup P_2 \cup \{p_{S1}, p_{S2}\} : M_0(p) = 0$

4. $P_F = \begin{cases} \{p_S, p_{F1}, p_{F2}\}, & \text{jestliže } p_{S1} \in P_{F1} \text{ nebo } p_{S2} \in P_{F2} \\ \{p_{F1}, p_{F2}\}, & \text{v opačném případě} \end{cases}$

□ □



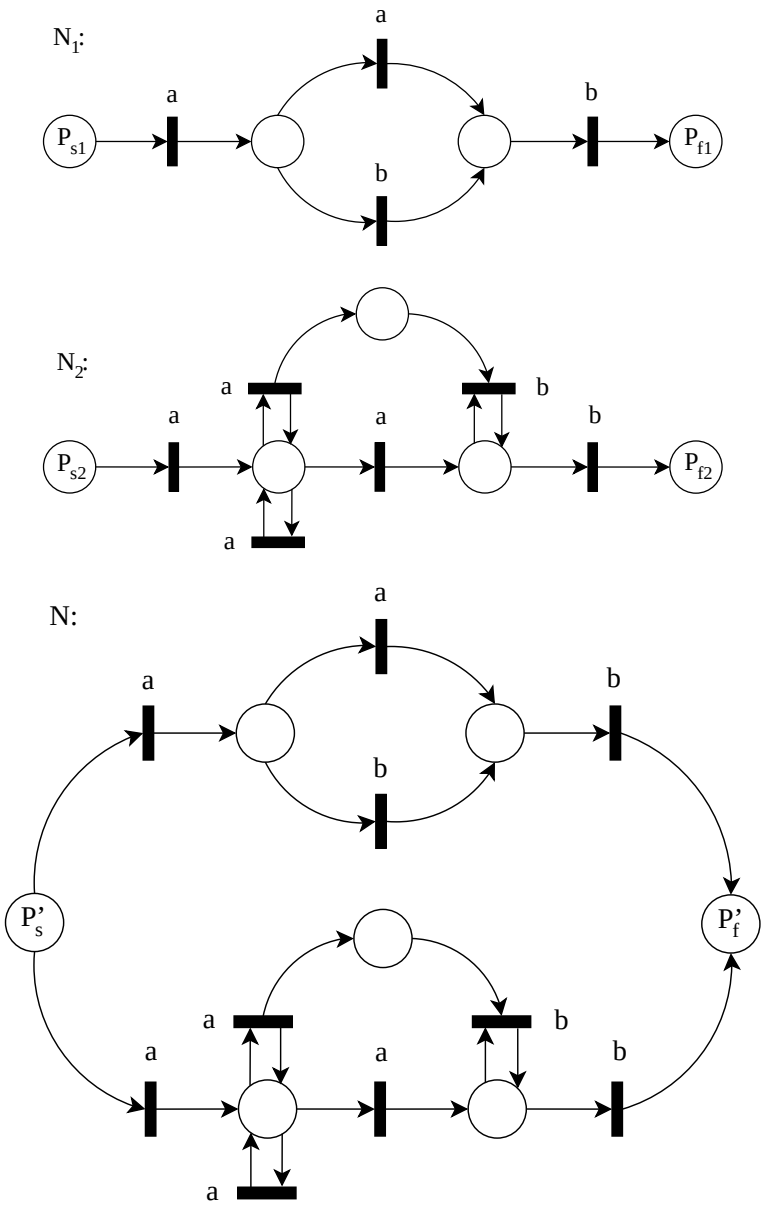
Obrázek 10.6: Ilustrace konkatenace Petriho sítí

x+y

■ **Příklad 10.2** *Sjednocení Petriho sítí.*

Na obrázku 10.7 jsou postupně uvedeny grafy ohodnocených Petriho sítí generujících jazyky $L_1 \equiv a(a+b)b$, $L_2 = \{a^m b^n | m > n > 1\}$ a $L = L_1 \cup L_2$. □

Pro modelování paralelní činnosti dvou Petriho sítí zavedeme speciální operátor *paralelní kompozice* řetězců a jazyků (angl. *concurrency*



Obrázek 10.7: Ilustrace sjednocení Petriho sítí

operator), který se označuje symbolem $\|$.

■ **Definice 10.5** *Paralelní spojení.*

DEF

Nechť $x_1, x_2 \in \Sigma^*$ jsou dva řetězce nad abecedou Σ a nechť $a, b \in \Sigma$ jsou symboly. *Paralelní spojení (kompozici) dvou řetězců* definujeme

rekurentně s využitím popisu ve tvaru regulárního výrazu:

$$ax_1 \parallel bx_2 = a(x_1 \parallel bx_2) + b(ax_1 \parallel x_2)$$

$$a \parallel \epsilon = \epsilon \parallel a = a$$

kde ϵ je prázdný řetězec. *Paralelní kompozice* $L_1 \parallel L_2$ jazyků L_1 a L_2 je definována :

$$L_1 \parallel L_2 = \{x \parallel y | x \in L_1 \wedge y \in L_2\}$$

□

x+y

■ **Příklad 10.3** *Paralelní spojení jazyků.*

Je-li $L_1 = \{a, b\}$ a $L_2 = \{c\}$, pak $L_1 \parallel L_2 = \{abc, acb, cab\}$. □

Dá se ukázat, že vzhledem k paralelní kompozici jsou uzavřeny jazyky regulární, kontextové a jazyky typu 0. Naproti tomu jazyky bezkontextové vzhledem k paralelní kompozici uzavřeny nejsou.

Věta

■ **Věta 10.4** *Uzavřenost vzhledem k paralelní kompozici.*

Nechť L_1 a L_2 jsou jazyky generované Petriho sítěmi. Pak jazyk $L = L_1 \parallel L_2$ je rovněž jazyk generovaný Petriho sítí.

Důkaz. Opět předpokládejme, že jazyky L_1 a L_2 jsou generovány sítěmi N_1 a N_2 ve standardním tvaru. Nejprve vytvoříme síť N sloučením sítí N_1 a N_2, ve které

1. $M_0(p_{S1}) = M_0(p_{S2}) = 1 \wedge \forall p \in P_1 \cup P_2 \setminus \{p_{S1}, p_{S2}\} : M_0(p) = 0$
2. $Q_F = \{M_F | M_F \in [M_0\rangle \wedge M_F(p_{F1}) = M_F(p_{F2}) = 1\}$

Podle věty 10.1 převedeme síť N na standardní tvar. □ □

x+y

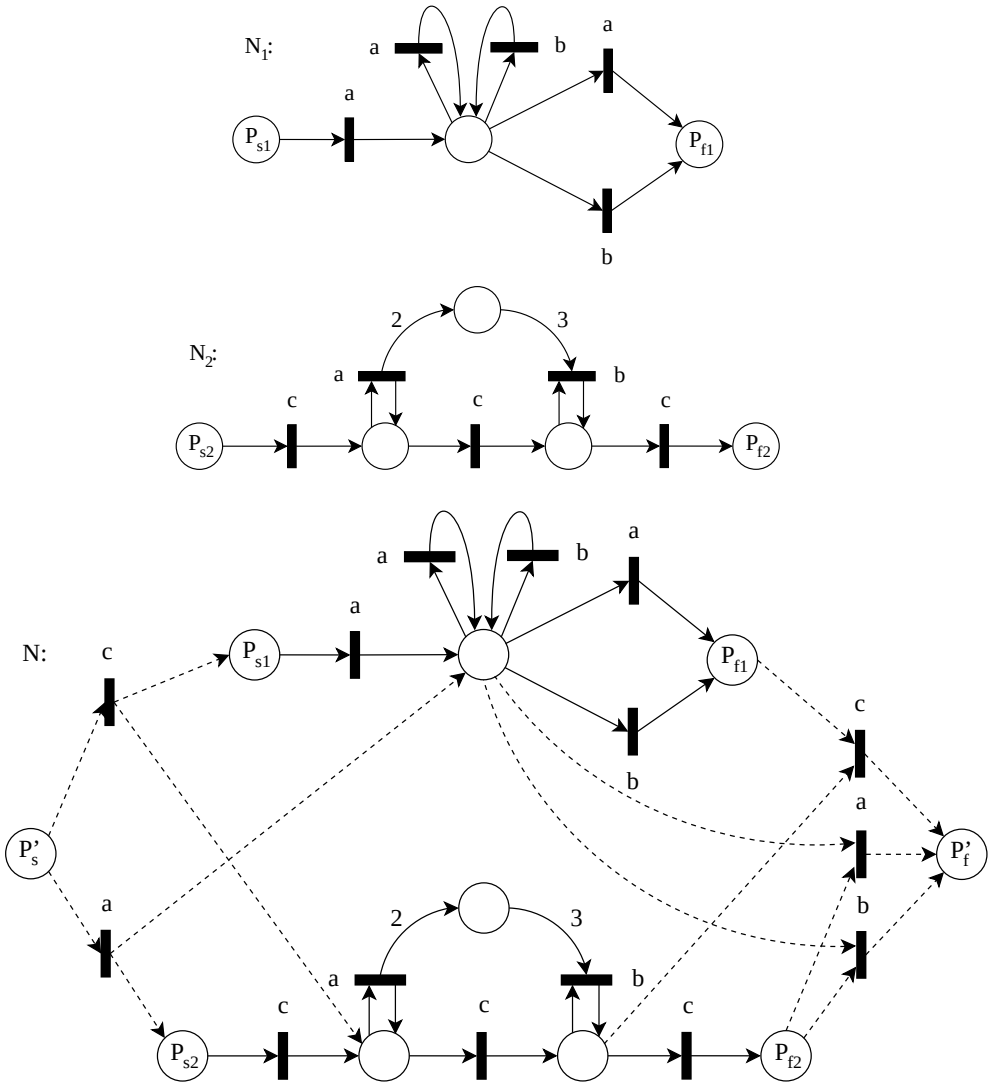
■ **Příklad 10.4** *Paralelní kompozice sítí.*

Výsledek konstrukce sítě generující paralelní kompozici jazyků $L_1 \equiv a(a+b)^+$ a $L_2 = \{ca^{3n}cb^{2n}c | n \geq 0\}$ ilustruje obrázek 10.8. Hrany, které jsou vyznačeny čárkovaně, vznikly převodem na standardní tvar sítě. □

Věta

■ **Věta 10.5** *Uzavřenost vzhledem k průniku jazyků.*

Nechť L_1 a L_1 jsou jazyky generované Petriho sítěmi. Pak $L = L_1 \cap L_2$ je rovněž jazyk generovaný Petriho sítí.



Obrázek 10.8: Ilustrace paralelní kompozice Petriho sítí

Důkaz. Nechť $L_1 = L(N_1)$, $L_2 = L(N_2)$ a $L = L(N)$. Musíme zajistit, aby výsledná síť N generovala právě ty řetězce, které generuje síť N_1 i N_2. K tomu účelu hledáme všechny dvojice $(t, t') \in T_1 \times T_2$, pro které $\lambda_1(t) = \lambda_2(t')$. Každou z takových dvojic nahradíme jediným přechodem $\widehat{t}$ sítě N podle tohoto postupu:

1. $${}^\bullet\widehat{t} = {}^\bullet t \cup {}^\bullet t' \wedge \forall p \in {}^\bullet\widehat{t} : W(p,\widehat{t}) = \begin{cases} W_1(p,t), & \text{jestliže } p \in P_1 \\ W_2(p,t'), & \text{jestliže } p \in P_2 \end{cases}$$

2. $$\widehat{t}^\bullet = t^\bullet \cup t'^\bullet \wedge \forall p \in \widehat{t}^\bullet : W(\widehat{t},p) = \begin{cases} W_1(t,p), & \text{jestliže } p \in t^\bullet \setminus t'^\bullet \\ W_2(t',p), & \text{jestliže } p \in t'^\bullet \setminus t^\bullet \\ W_1(t,p) + W_2(t',p), & \text{jestliže } p \in t^\bullet \cap t'^\bullet \end{cases}$$

3. $\lambda(\widehat{t}) = \lambda_1(t)$
4. Zavedeme nové počáteční místo p_{s} sítě N. Jestliže pro přechod $\widehat{t} \in T$, jenž byl vytvořen podle (1) a (2), platí ${}^\bullet\widehat{t} = \{p_{\mathsf{S1}}, p_{\mathsf{S2}}\}$, pak přechod $\widehat{t}$ nahradíme přechodem $\widetilde{t}$, pro který
   (a) ${}^\bullet\widetilde{t} = \{p_{\mathsf{s}}\}$, $W(p_{\mathsf{s}}, \widetilde{t}) = 1$
   (b) $\widetilde{t}^\bullet = \widehat{t}^\bullet$, $\forall p \in \widetilde{t}^\bullet : W(\widetilde{t}, p) = W(\widehat{t}, p)$
   (c) $\lambda(\widetilde{t}) = \lambda(\widehat{t})$
5. Analogicky s (4) vytvoříme nové koncové místo p_{F} a jeho „spojení“ s přechodem $\widehat{t}$, pro který ${}^\bullet\widehat{t} = \{p_{\mathsf{F1}}, p_{\mathsf{F2}}\}$.

□ □

x+y

■ **Příklad 10.5** *Průnik Petriho sítí.*

Příklad konstrukce sítě N, pro kterou $L(N) = L(N_1) \cap L(N_2)$, kde $L(N_1) = \{ca^{3n}cb^{2n}c | n \geq 0\}$ a $L(N_2) = \{ca^{2n}cb^{3n}c | n \geq 0\}$, je uveden na obrázku 10.9. □

Věta

■ **Věta 10.6** *Uzavřenost vzhledem k reverzi.*

Jazyky Petriho sítí jsou uzavřeny vzhledem k reverzi, tj. je-li $L = L(N)$ jazyk generovaný Petriho sítí N, pak existuje Petriho síť N' taková, že $L(N') = L^{\mathsf{R}}$.

Důkaz. Připomeňme, že jazyk $L^{\mathsf{R}} = \{x^{\mathsf{R}} | x \in L\}$, kde

1. $\epsilon^{\mathsf{R}} = \epsilon$
2. je-li $x = a_1 a_2 \ldots a_n$, pak $x^{\mathsf{R}} = a_n a_{n-1} \ldots a_1$.

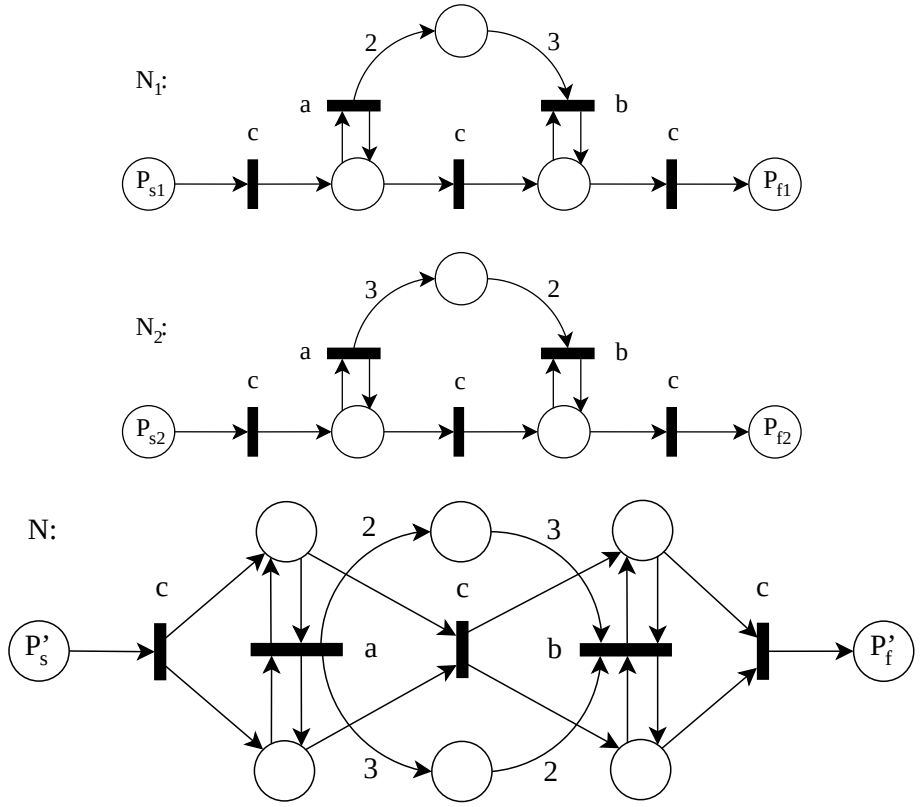
Konstrukce sítě N', za předpokladu, že N je ve standardním tvaru, je zřejmě triviální. Stačí zaměnit počáteční a koncové místo a orientaci hran sítě N a dostaneme již síť N'. □ □

Dříve, než začneme diskutovat operace nad jazyky, vzhledem ke kterým jazyky typu **L** uzavřeny nejsou, shrneme dosavadní výsledky o uzavřenosti **L**-jazyků s doplněním zřejmé možnosti kombinace uzávěrových vlastností.

Věta

■ **Věta 10.7** *Uzavřenost jazyků Petriho sítí.*

Jazyky Petriho sítí jsou uzavřeny vzhledem ke konečnému počtu aplikací operací konkatenace, sjednocení, průniku, paralelní kompozice a reverze, a to v libovolném pořadí aplikace.



Obrázek 10.9: Ilustrace průniku Petriho sítí

Důkaz. Je přímým důsledkem definice uzavřenosti jazyků vzhledem k operaci nad jazyky a vět 10.2 až 10.6. □ □

V souvislosti s uzavřeností **L**-jazyků vzhledem ke sjednocení a průniku vzniká důležitá otázka, zda třída **L**-jazyků tvoří Boolovu algebru, tj. zda je uzavřena i vzhledem ke komplementu. Podle [9] je pro jazyky typu **L** tato otázka otevřeným problémem; bylo ukázáno, že mimo tuto třídu existují jazyky Petriho sítí, jejichž komplement nelze generovat žádnou Petriho sítí.

Stejně důležitou operací jako je komplement, pro kterou je známa bohužel negativní odpověď, je operace iterace jazyků, která má úzký vztah k modelování obecných cyklů.

■ **Věta 10.8** *Neuzavřenost vzhledem k iteraci.* Věta

Jazyky Petriho sítí nejsou uzavřeny vzhledem k operaci $*$ iterace jazyka.

Důkaz. Nejprve si uvědomíme, že věta 10.8 není ve sporu s větou 10.7, protože konstrukce iterace jazyka $L^* = L^0 \cup L^1 \cup \ldots \cup L^n \cup \ldots$ předpokládá uzavřenost k libovolnému (nekonečnému) počtu aplikací operace konkatenace a sjednocení.

Elegantní důkaz pak vychází z teorie abstaktních tříd jazyků (Abstract Families of Languages AFL) [11]. Bylo dokázáno, že nejmenší úplná třída AFL, která je uzavřena vzhledem k průniku a nekonečné konkatenaci jazyků, a která obsahuje jazyk $\{a^n b^n | n \geq 0\}$, obsahuje také každou rekurzivně vyčíslitelnou množinu (každý jazyk typu 0). Vzhledem k tomu, že jazyky Petriho sítí jsou uzavřeny vzhledem k průniku, obsahují jazyk $\{a^n b^n | n \geq 0\}$ (viz příklad 10.1) a jsou vlastní podmnožinou jazyků typu 0, nemohou být uzavřeny vzhledem k nekonečné konkatenaci a tudíž ani k iteraci.

Tento fakt může být překvapující, protože je dobře známo, že všechny třídy 3, 2, 1, 0 Chomského hierarchie formálních jazyků jsou uzavřeny vzhledem k iteraci. K lepšímu pochopení nám pomůže následující definice a věta. □

□

DEF

■ **Definice 10.6** *Řádně ukončující Petriho síť.*

Petriho síť se nazývá *řádně ukončující* (angl. *properly terminating*), jestliže splňuje následující. Kdykoli provádění Petriho sítě skončí, pak jsou splněny obě podmínky:

1. jediná značka je umístěna v koncovém místě sítě,

2. počet značek vytvořených v průběhu provádění je konečný.

□

Pro naši diskusi je podstatná podmínka (1), která vylučuje situaci, kdy pro určité značení M není přechodová funkce $\delta(M, t)$ definována pro žádný přechod t a přitom $M \notin Q_{\mathsf{F}}$ (tj. M není koncovým stavem). Pro takové sítě pak lze „spojením“ počátečního a koncového místa zajistit nekonečnou konkatenaci generovaných řetězců a proto platí věta 10.9.

Věta

■ **Věta 10.9** *Uzavřenost řádně ukončujících sítí vzhledem k iteraci.*

Třída jazyků generovaných řádně ukončujícími Petriho sítěmi je uzavřena vzhledem k iteraci.

Důkaz. Bez důkazu. □ □

Dá se však ukázat, že tato třída je pouze třídou regulárních jazyků, tj. každá řádně ukončující Petriho síť je ekvivalentní určitému konečnému automatu.

Poslední operací, na kterou obrátíme pozornost vzhledem k vlastnosti uzavřenosti, je operace substituce jazyků (do jazyka). Tato operace má rovněž značný význam pro modelování; úzce souvisí s principem abstrakce. Lze ji použít k aplikaci hierarchického modelování, kdy jistý přechod je nahrazen podsítí, tj. komplexní operace, je vyjádřena posloupnostmi dílčích operací.

Vzhledem k charakteru substituovaných řetězců můžeme rozlišit tři typy substituce. Nechť L je jazyk do něhož provádíme substituci, tj. nahrazujeme každý symbol $a \in \Sigma$ každé věty $x \in L$. Mluvíme o:

1. *obecné substituci*, jestliže symbol a může být nahrazen libovolným řetězcem substituujícího jazyka L_a, přičemž L_a je libovolný (i nekonečný) formální jazyk,
2. *konečné substituci*, jestliže L_a je konečný jazyk,
3. *homomorfismu*, je-li L_a tvořen jediným řetězcem.

Opět, bohužel, jazyky Petriho sítí nejsou uzavřeny vzhledem k obecné substituci, tj. obecně neplatí, že substitucí jazyka L_a generovaného jistou Petriho sítí do jazyka L generovaného jinou Petriho sítí získáme jazyk generovatelný nějakou Petriho sítí. Jako příklad lze uvést substituci jazyka

$$L_c = \{a^n b^n | n \geq 1\}$$

do jazyka

$$L = \{c^i | i \geq 1\}$$

L_c i L jsou zřejmě jazyky Petriho sítí. Výsledek substituce, jazyk

$$L' = \{a^{m_1} b^{m_1} a^{m_2} b^{m_2} \ldots a^{m_k} b^{m_k} | m_i \geq 1 \wedge k \geq 1\}$$

je však jazykem, který splňuje větu 10.8 ($L' = L_c^+$) a není jazykem generovatelným Petriho sítí (L_c je typický bezkontextový jazyk).

Na druhé straně, omezíme-li se na substituci typu (2) a (3) dostáváme, jak uvidíme, vzhledem k uzavřenosti substituce, pozitivní odpověď.

■ **Věta 10.10** *Uzavřenost vzhledem ke konečné substituci.* Věta

Nechť L_1 je jazyk generovaný Petriho sítí a L_2 je regulární jazyk. Pak jazyk L, který vznikne konečnou substitucí jazyka L_2 do jazyka L_1, je jazyk generovatelný Petriho sítí.

Důkaz. Schéma důkazu vychází z věty 10.9. Poněvadž L_2 je regulární jazyk, může být generován řádně ukončující Petriho sítí. Nahradíme-li každý přechod s ohodnocením a sítě generující jazyk L_1 takovou Petriho sítí, pak výsledná síť generuje jazyk $L_{a \to L_2}$ (jazyk získaný substitucí vět jazyka L_2 do vět jazyka L_1 na místo symbolu a). □

□

Důsledek.

Třída jazyků Petriho sítí je uzavřena vzhledem ke konečné substituci a vzhledem k homomorfismu. □

10.2.3 Vztah jazyků Petriho sítí k Chomského hierarchii jazyků

V této podsekci se opět zaměříme pouze na jazyky Petriho sítí typu **L**. Naším cílem bude nalezení vztahu třídy těchto jazyků k třídám jazyků Chomského hierarchie, t.j. k jazykům regulárním, bezkontextovým, kontextovým a rekurzivně vyčíslitelným.

Věta

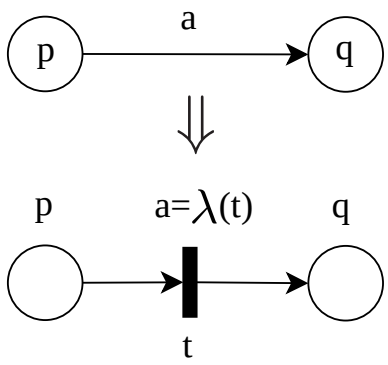
■ **Věta 10.11** *Pokrytí třídy regulárních jazyků.*

Každý regulární jazyk je jazykem generovaným Petriho sítí.

Důkaz. Ukážeme, že ke každému konečnému automatu $M = (Q, \Sigma, \delta, q_0, F)$, kde $\delta : Q \times \Sigma \to 2^Q$, $q_0 \in Q$ a $F \subseteq Q$, sestrojíme ohodnocenou Petriho síť $N = (P, T, F, W, \Sigma, p_\mathsf{s}, \lambda, M_0, Q_\mathsf{F})$ takovou, že $L(M) = L(N)$. Základní myšlenku konstrukce ilustruje obrázek 10.10.

Zvolíme tedy jednotlivé složky sítě N tak, aby platilo:

1. $P = Q$
2. $T = \{t_{paq} | p, q \in Q, a \in \Sigma, q \in \delta(p, a)\}$
3. $\forall t_{paq} \in T : {}^\bullet t_{paq} = \{p\} \wedge t_{paq}^\bullet = \{q\} \wedge W(p, t_{paq}) = W(t_{paq}, q) = 1$
4. $p_\mathsf{s} = q_0$
5. $\lambda : T \to \Sigma, \forall t_{paq} \in T : \lambda(t_{paq}) = a$
6. $M_0 : P \to \mathbb{N}, M_0(p_\mathsf{s}) = 1 \wedge \forall p \in P \setminus \{p_\mathsf{s}\} : M_0(p) = 0$
7. $Q_\mathsf{F} = \{M_\mathsf{F} | M_\mathsf{F} \in [M_0\rangle \wedge (\exists p : M_\mathsf{F}(p) \neq 0 \wedge p \in F)\}$



Obrázek 10.10: Převod konečného automatu na ekvivalentní Petriho síť

Nyní lze formálně dokázat ekvivalenci:

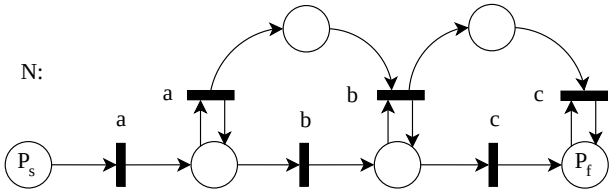
$$w \in L(M) \Leftrightarrow w \in L(N)$$

tj. $L(M) = L(N)$. □ □

Opačné tvrzení k větě 10.11, jak jistě očekáváme, neplatí.

Důkazem, že ne každý jazyk generovaný Petriho sítěmi je jazykem regulárním, je např. jazyk $L = \{a^n b^n | n \geq 1\}$ generovaný Petriho sítí na obrázku 10.6, o němž víme, že je jazykem bezkontextovým. Jazyky Petriho sítí jsou tedy vlastní nadtřídou regulárních jazyků.

Nyní se budeme zabývat vztahem jazyků generovaných Petriho sítěmi a jazyků bezkontextových. Na obrázku 10.11 je zobrazena Petriho síť generující jazyk $L = \{a^n b^n c^n | n \geq 1\}$.



Obrázek 10.11: Petriho síť generující kontextový jazyk

Tento jazyk není jazykem bezkontextovým (viz např. [2]), ale typickým jazykem kontextovým. Vzniká tak otázka, zda pro jazyky Petriho sítí platí, vzhledem k bezkontextovým jazykům, stejný vztah jako k jazykům regulárním, tj. zda jsou vlastní nadtřídou jazyků bezkontextových. Negativní odpověď dává lemma 10.1.

Lemma

■ **Lemma 10.1**

Jazyk $L = \{ww^{\mathsf{R}} | w \in \Sigma^*\}$ není jazykem Petriho sítí.

Důkaz. Nejprve odvodíme nutnou podmínku pro stavový prostor Petriho sítě generující jazyk L a pak ukážeme, že tato podmínka nemůže být splněna.

Předpokládejme tedy, že existuje Petriho síť N, $L(N) = L$. Nechť $|\Sigma| = k$, $k > 1$ a uvažujme řetězec $xx^{\mathsf{R}} \in L$, $|x| = r$. Poněvadž existuje k^r různých možných řetězců x, musí stavový prostor sítě N obsahovat alespoň k^r různých stavů (dostupných značení) tak, aby provedení r přechodů generujících řetězec x vedlo k jednoznačnému „zapamatování“ struktury řetězce x. Skutečně, pokud bychom disponovali menším počtem stavů, pak by pro jisté řetězce x_1 a x_2 platilo $\delta(M_0, x1) = \delta(M_0, x_2)$ pro $x_1 \neq x_2$. Pak ale

$$\delta(M_0, x_1 x_2^{\mathsf{R}}) = \delta(\delta(M_0, x_1), x_2^{\mathsf{R}}) = \delta(\delta(M_0, x_2), x_2^{\mathsf{R}}) = \delta(M_0, x_2 x_2^{\mathsf{R}}) \in Q_{\mathsf{F}}$$

a tedy by platilo $x_1 x_2^{\mathsf{R}} \in L$, $x_1 \neq x_2$, což je ve sporu s definicí jazyka L.

Nyní však ukážeme, že podmínka, aby provedení výpočetní posloupnosti délky r aktualizovalo libovolný ze stavů množiny o mohutnosti k^r je nesplnitelná. Uvažujme takovou výpočetní posloupnost:

$$M_0 \ [t_1\rangle \ M_1 \ [t_2\rangle \ \ldots \ [t_r\rangle \ M_r$$

a předpokládejme, že množina přechodů T sítě N má mohutnost m.

Podle věty 9.5 můžeme stav (značení) M_r vyjádřit ve tvaru

$$M_r = M_0 + \underline{N}.u$$

kde $\underline{\mathrm{N}}$ je matice Petriho sítě a u je vektor $u : T \to \mathbb{N}$

$$u(t) = |\{i : t_i = t \wedge 1 \leq i \leq r\}|$$

Zřejmě platí $\sum_{t \in T} u(t) = r$. V nejlepším případě, každý z vektorů u splňující tuto podmínku generuje různý stav M_r. K vyčíslení počtu různých vektorů u použijeme vztah pro počet rozkladů čísla $r \in \mathbb{N}$ na m nezáporných celočíselných členů (dávající v součtu r), který je roven kombinačnímu číslu

$$\begin{pmatrix} r+m-1 \\ m-1 \end{pmatrix}$$

Protože

$$\begin{pmatrix} r+m-1 \\ m-1 \end{pmatrix} = \frac{(r+m-1)\ldots(r+1)}{(m+1)!} < (r+m)^m$$

je počet dosažitelných stavů po provedení r přechodů ostře menší než $(r+m)^m$. Pro dostatečně velké r pak platí $(r+m)^m < k^r$ a nutná podmínka pro generování řetězce xx^{R} není splněna. Jazyk L tedy není jazykem Petriho sítí. □ □

■ **Věta 10.12** *Nesrovnatelnost jazyků Petriho sítí a bezkontextových.* Věta

Třída jazyků Petriho sítí a třída bezkontextových jazyků je nesrovnatelná vzhledem k inkluzi.

Důkaz. Plyne z existence jazyků popsaných na obrázku 10.11 a v lemmatu 10.1. Poznamenejme, že jazyk L z lemmatu 10.1 je skutečně jazykem bezkontextovým; může být generován např. gramatikou s množinou pravidel $\{S \to aSa | a \in \Sigma\} \cup \{S \to \epsilon\}$. □ □

Lemma 10.1 a jeho důkaz ukazují důležitou omezující vlastnost Petriho sítí. Petriho sítě, stejně jako konečné automaty, nejsou schopny „si zapamatovat“ libovolně dlouhou posloupnost libovolných symbolů, ale pouze posloupnost omezené délky. Nemohou tak simulovat činnost zásobníkových automatů, které, jak víme, jsou alternativním specifikačním prostředkem pro vymezení celé třídy bezkontextových jazyků. Důvodem je dokázaná nekompatibilita potřebných stavových prostorů; Petriho síť může dát k dispozici, v závislosti na délce vstupního řetězce, pouze kombinatoricky rostoucí počet dostupných stavů, kdežto zásobníkový automat disponuje exponenciálně rostoucím počtem.

Na druhé straně Petriho síť generuje jazyky (obrázek 10.11), které zásobníkový automat není schopen přijímat. To je důsledkem odlišnosti v řízení obou automatů. Přechod zásobníkového automatu je odvozován z vrcholu zásobníku, kdežto Petriho síť zpřístupňuje v každém okamžiku libovolný čítač (místo).

Důležitá otázka, které bezkontextové jazyky jsou současně jazyky Petriho sítí, zůstává dosud otevřeným problémem. Můžeme však vymezit podtřídu bezkontextových jazyků, které tuto vlastnost mají. Jsou to tzv. *omezené bezkontextové jazyky* (angl. *bounded context-free languages*).

■ **Definice 10.7** *Omezený bezkontextový jazyk.* DEF

Bezkontextový jazyk L se nazývá *omezeným bezkontextovým jazykem* nad abecedou Σ, jestliže existují řetězce $w_1, w_2, \ldots, w_n \in \Sigma^*$ takové, že $L \subseteq w_1^* w_2^* \ldots w_n^*$. □

Omezené bezkontextové jazyky jsou charakterizovány následující větou, jejíž důkaz lze nalézt v [11].

Věta

■ **Věta 10.13** *Charakteristika omezených bezkontextových jazyků.*

Třída omezených bezkontextových jazyků je nejmenší třída jazyků splňující tyto podmínky:

1. Je-li B konečná podmnožina množiny Σ^*, pak B je omezený bezkontextový jazyk.
2. Jsou-li B_1 a B_2 omezené bezkontextové jazyky, pak $B_1 \cup B_2$ a $B_1.B_2$ jsou omezené bezkontextové jazyky.
3. Je-li B omezený bezkontextový jazyk a $x, y \in \Sigma^*$, pak jazyk $\{x^i B y^i | i \geq 0\}$ je omezený bezkontextový jazyk. Poznámka: $\{x^i B y^i | i \geq 0\} = \{x^i z y^i | z \in B \wedge i \geq 0\}$.

□

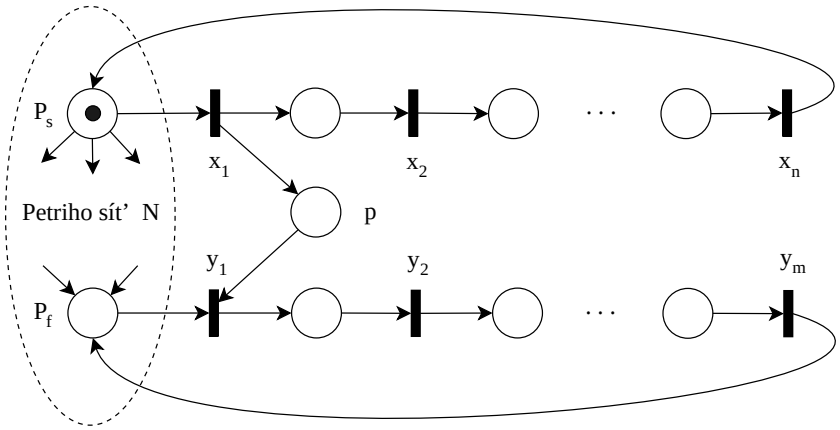
Věta

■ **Věta 10.14** *Pokrytí omezených bezkontextových jazyků Petriho sítěmi.*

Každý omezený bezkontextový jazyk je jazykem generovaným Petriho sítí.

Důkaz. K důkazu využijeme tvrzení věty 10.13. Ukážeme, že každá z podmínek (1)–(3) platí v určité podtřídě jazyků Petriho sítí a tudíž existuje podtřída jazyků Petriho sítí, které generují právě omezené bezkontextové jazyky.

1. Jazyk splňující tvrzení (1) věty 10.13 je regulární a tedy (podle věty 10.11) je jazykem Petriho sítě.
2. Podmínka (2) věty 10.13 je splněna pro všechny jazyky Petriho sítí (věty 10.2 a 10.3 uzavřenost vzhledem ke konkatenaci a sjednocení jazyků).
3. Abychom ukázali, že je splněna i podmínka (3) věty 10.13, popíšeme konstrukci Petriho sítě generující jazyk $\{x^i B y^i | i \geq 0\}$. Předpokládejme, že jazyk B je generován Petriho sítí N ve standardním tvaru a nechť $x = x_1 x_2 \ldots x_n$ a $y = y_1 y_2 \ldots y_m$. Na obrázky 10.12 je schematicky znázorněna konstrukce výsledné sítě. Vidíme, že těžištěm konstrukce je pomocné místo p, které má funkci čítače; každé generování řetězce $x = x_1 x_2 \ldots x_n$ zvýší počet značek místa p o jedničku. Protože koncový stav sítě vyžaduje značku pouze v místě p_F, je tedy řetězec $y = y_1 y_2 \ldots y_m$ generován právě tolikrát, kolikrát byl generován řetězec x. Formální popis konstrukce ponecháme jako cvičení.



Obrázek 10.12: Petriho síť generující omezený bezkontextový jazyk

Dokázali jsme tedy, že každý omezený bezkontextový jazyk lze generovat jistou Petriho sítí. □

□

Omezené bezkontextové jazyky jsou vlastní podtřídou bezkontextových jazyků Petriho sítí. Existují regulární jazyky, které nejsou omezenými bezkontextovými jazyky, např. jazyk popsaný regulárním výrazem $(a + b)^+$, jak je ukázáno v [11]. Jiným příkladem bezkontextového (neregulárního) jazyka, který je generovatelný Petriho sítí a není omezeným bezkontextovým jazykem, je jazyk $\{a^n b^n | n \geq 0\}$.

Nyní se zaměříme na vztah jazyků Petriho sítí ke kontextovým jazykům. Především nás zajímá, zda existují jazyky Petriho sítí, které nejsou kontextové, tj. zda existují Petriho sítě, které mají modelovací schopnost Turingových strojů. Odpověď na tuto otázku je negativní.

■ **Věta 10.15** *Pokrytí Petriho sítí kontextovými jazyky.* Věta

Všechny jazyky Petriho sítí jsou kontextové jazyky.

Důkaz. Precizní důkaz tohoto tvrzení není jednoduchý, protože vyžaduje nalezení konstrukce ekvivalentní kontextové gramatiky (nebo lineárně omezeného automatu) k libovolné Petriho síti. Tato konstrukce byla nalezena a je popsána v [12]. □ □

Pravdivost věty 10.15 můžeme podepřít i následující argumentací. Předpokládáme-li, že Petriho sítě nepopírají Turingovu tezi, pak lze k dané Petriho síti N sestrojit Turingův stroj T, který simuluje síť N a přijímá jazyk $L(T) = L(N)$. Nechť páska stroje T uchovává momentální značení každého místa sítě N; po přečtení vstupního symbolu

je simulováno provedení příslušného přechodu, tj. změna značení některých míst. Ukažme, že takto navržený stroj T je lineárně omezený automat a tedy jazyk $L(N)$ je jazykem kontextovým.

Nejprve připomeňme, že lineárně omezený automat je Turingův stroj, který pracuje tak, že využívanou část pásky lze ohraničit jistou lineární funkcí závisející na délce vstupního řetězce. Nyní kvantifikujme tuto využívanou část pásky stroje T celkovým součtem S všech značek všech míst a zkoumejme, jak se tento součet mění v závislosti na délce vstupního řetězce.

Nechť vstupnímu řetězci délky $k \geq 1$ odpovídá výpočetní posloupnost $t_1 t_2 \ldots t_k$ provedených přechodů sítě N. Označme celočíselnou proměnnou d_t počet značek, kterým přispívá přechod t (jeho provedení) k celkovému počtu značek sítě. Zřejmě

$$d_t = \sum_{p \in t^\bullet} W(t,p) - \sum_{p \in {}^\bullet t} W(p,t)$$

Pak počet značek S sítě N, po provedení výpočetní posloupnosti $t_1 t_2 \ldots t_k$ lze vyjádřit ve tvaru:

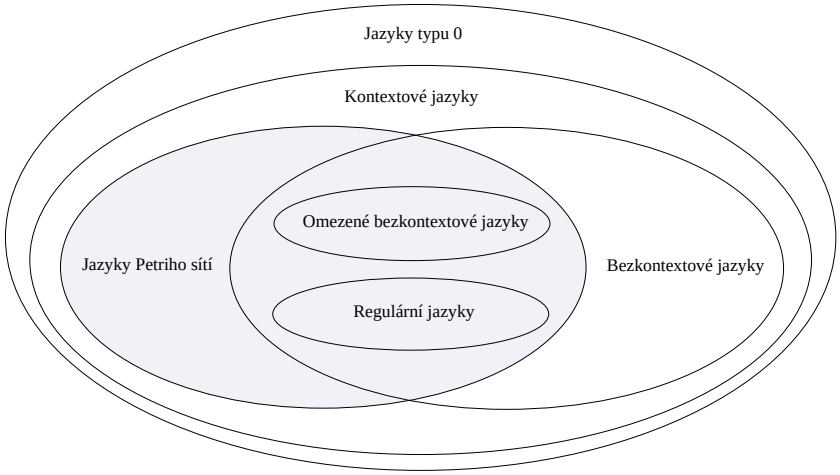
$$S = 1 + \sum_{i=1}^{k} d_{t_i}$$

Z definice Petriho sítě plyne existence maxima $m = \max_{t \in T} d_t$, s jehož využitím lze hodnoty S ohraničit v závislosti na délce výpočetní posloupnosti a tudíž i vstupního řetězce:

$$S(k) \leq 1 + k.m$$

Protože výraz na pravé straně představuje lineární funkci nezávisle proměnné k, je Turingův stroj T lineárně omezeným automatem a jazyk $L(T)$ jazykem kontextovým.

Inkluze jazyků Petriho sítí a kontextových jazyků je vlastní, protože jsme ukázali, že existují i bezkontextové jazyky, které nelze generovat žádnou Petriho sítí. Na obrázku 10.13 jsou shrnuty získané poznatky v grafické podobě.



Obrázek 10.13: Vztah jazyků Petriho sítí k Chomského hierarchii

Pojmy k zapamatování

- jazyk typu L, G, T a P
- Petriho síť ve standardním tvaru
- uzavřenost jazyků vzhledem ke konkatenaci, sjednocení, paralelní kompozici, průniku, reverzi, iteraci a konečné substituci

Shrnutí

V této kapitole jsme se soustředili na jazyky Petriho sítí. Existují čtyři různé definice, které vymezují třídy jazyků označované jako jazyky typu L, G, T a P. Pro zjednodušení provádění sítě a rozhodování, zda generovaný řetězec je prvkem jazyka zavádíme Petriho síť ve standardním tvaru. Existence standardního tvaru Petriho sítě typu L jsme účelně využili při důkazech a konstrukcích souvisejících s uzávěrovými vlastnostmi této třídy jazyků. Další z motivací studia jazyků Petriho sítí je jejich začlenění do Chomského hierarchie formálních jazyků.

Příklady k procvičení

1. Dokažte nebo vyvraťte uzavřenost jazyků Petriho sítí vzhledem ke konkatenaci, sjednocení, průniku, paralelní kompozici a reverzi.
2. Vymezte třídu jazyků Petriho sítí typu L.
3. Uveďte vztah třídy jazyků Petriho sítí typu L k ostatním třídám Chomského hierarchie jazyků.
4. Sestrojte grafy ohodnocených Petriho sítí generujících následující jazyky:
   (a) $L_1 = a(a+b)^+$
   (b) $L_2 = \{a^m b^n \mid n > m > 1\}$
   (c) $L_3 = L_1.L_2$

Kapitola 11

Podtřídy a rozšíření Petriho sítí

Čas potřebný ke studiu: 5 hodin

Cíle kapitoly

Tato kapitola uvádí vybrané podtřídy P/T Petriho sítí, stavové stroje, značené grafy a sítě s volným výběrem. Tyto podtřídy mají nižší *modelovací mocnost* (viz sekce 11.1) než třída P/T Petriho sítí, jejich *rozhodovací mocnost* (také viz sekce 11.1) je však vyšší.
Na závěr kapitoly uvádíme naopak rozšíření třídy Petriho sítí, zejména o inhibiční hrany (inhibotory), které zvyšují modelovací mocnost až na úroveň Turingových strojů, výrazně však snižují rozhodnutelnost problému analýzy. Další rozšíření, jako časové a časované Petriho sítě, stochastické Petriho sítě, barvené Petriho sítě a Petriho sítě vyšší úrovně jsou studovány pouze stručně.

Průvodce studiem

Pro porozumění této kapitole je zapotřebí znát definice P/T sítí uvedené v kapitole 7) a rozumět základním problémům analýzy, viz kapitola 8.

Obsah

11.1 Modelovací a rozhodovací mocnost

V této kapitole se seznámíme s některými modifikacemi, které představují zúžení a rozšíření základní třídy Petriho sítí (P/T Petriho sítí). Tyto modifikace jsou ve většině případů motivovány dvěma aspekty matematických nebo formálních modelů, které se nazývají *modelovací a rozhodovací mocnost modelu* (algl. *modeling power and decision power*).

Modelovací mocnost představuje schopnost modelu vyjadřovat fundamentální rysy modelovaného systému a vede k úspěšné reprezentaci systému modelem. *Rozhodovací mocnost*modelu se vztahuje k existenci a znalostem postupů, jež umožňují analyzovat specifické verze modelu a určovat vlastnosti modelovaného systému.

Modelovací a rozhodovací mocnost modelu jsou, bohužel, v jistém smyslu protichůdné vlastnosti. Zvýšením modelovací mocnosti obvykle klesá rozhodovací mocnost třídy modelů a naopak. Výstižným příkladem je hierarchie automatů, která přísluší Chomského hierarchii jazyků.

11.2 Podtřídy Petriho sítí

S nejjednodušší podtřídou Petriho sítí jsme se setkali v odstavci 10.2.3. Tvoří ji sítě nazývané stavové stroje, které jsou ekvivalentní s konečnými automaty.

11.2.1 Stavové stroje

DEF

■ **Definice 11.1** *Stavový stroj.*

Petriho síť $N = (P, T, F, W, K, M_0)$ nazýváme *stavovým strojem* (angl. *state machine*), jestliže splňuje podmínku:

$$\forall t \in T : |{}^\bullet t| = |t^\bullet| = 1 \wedge W({}^\bullet t, t) = W(t, t^\bullet) = 1$$

□

Stavové stroje mají výbornou rozhodovací mocnost. Jsou striktně konzervativní a tudíž jejich stavový prostor (a strom dosažitelných značení) je konečný a umožňuje tak rozhodnout všechny otázky analýzy. Jejich modelovací mocnost je však stejná jako mocnost konečných automatů a proto je jejich samostatné používání omezené. Důležitou úlohu mají v metodě analýzy živosti obecných Petriho sítí, která je založena na vyhledávání podsítí izomorfních se stavovými stroji (tzv. SCSM-komponent (Strongly Connected State Machines)) a jejich pokrytí dané Petriho sítě [13].

11.2.2 Značené grafy

Další podtřídou Petriho sítí jsou značené grafy (synchronizační grafy). Tato třída může být z hlediska grafů chápána jako duální ke třídě stavových strojů, protože značený graf se vyznačuje právě jedním vstupním a jedním výstupním přechodem každého místa.

DEF

■ **Definice 11.2** *Značený graf.*

Petriho síť $N = (P, T, F, W, K, M_0)$ se nazývá *značený graf* (angl. *marked graph*), jestliže splňuje podmínky:

1. $\forall t_1, t_2 \in T : t_1 F^* t_2$, tedy graf sítě N je silně souvislý
2. $\forall p \in P : |{}^\bullet p| = |p^\bullet| = 1 \wedge W({}^\bullet p, p) = W(p, p^\bullet) = 1$

□

Na obrázku 11.1 je příklad značeného grafu a jeho ekvivalentní, často používané grafové reprezentace, která dala název této třídě.

Modelovací mocnost značených grafů a stavových strojů se liší. Značené grafy, na rozdíl od stavových strojů, mohou při provádění vytvářet nové značky nebo rušit (absorbovat) již vytvořené značky, což je podmínka pro modelování paralelismu nebo synchronizace (čekání na vytvořenou značku). Nemohou však modelovat konflikty (viz sekce 7.4) nebo rozhodnutí závislá na datech (nelze „větvit“ místa).

Pro analýzu živosti, bezpečnosti a dosažitelnosti značených grafů má zásadní význam pojem cyklu.

DEF

■ **Definice 11.3** *Cyklus značeného grafu.*

Nechť $N = (P, T, F, W, K, M_0)$ je značený graf. *Cyklem značeného grafu* nazýváme posloupnost C přechodů a míst

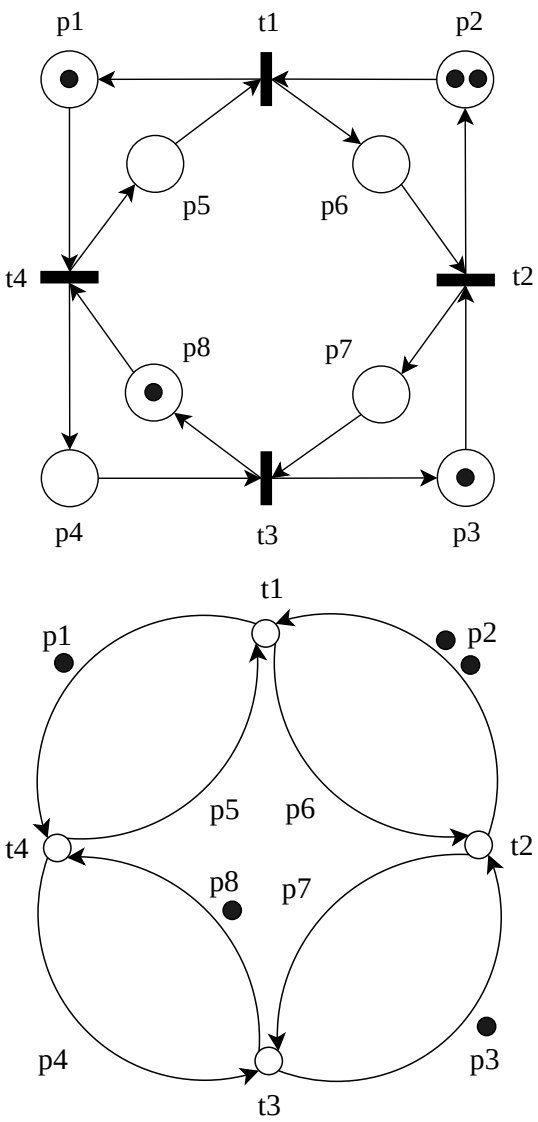
$$C = t_1 p_1 t_2 p_2 \ldots p_{n-1} t_n$$

pro kterou platí:

1. $\langle t_i, p_i \rangle \in F \wedge \langle p_i, t_{i+1} \rangle \in F$ pro $i = 1, 2, \ldots, n-1$
2. $t_1 = t_n$
3. $p_i^\bullet \neq p_j^\bullet \wedge {}^\bullet p_i \neq {}^\bullet p_j$ pro $1 \leq i \neq j \leq n-1$

□

V alternativní grafové reprezentaci značeného grafu odpovídá cyklus značeného grafu cyklu orientovaného grafu určeného vrcholy $t_1 t_2 \ldots t_n$.



Obrázek 11.1: Značený graf a jeho alternativní reprezentace

Na obrázku 11.1 můžeme snadno identifikovat např. cykly $t_1t_2t_1$ nebo $t_1t_2t_3t_4$ nebo $t_1t_4t_3t_2t_1$.

Rozhodnutelnost zmíněných problémů analýzy značených grafů je založena na následujících větách, které uvedeme bez důkazů.

■ **Věta 11.1** *Značení cyklu značeného grafu v průběhu evoluce sítě.* Věta

Počet značek obsažených v cyklu značeného grafu se během provádění přechodů nemění, tj. místa, která jsou součástí cyklu, tvoří P-invariant značeného grafu. □

Věta

■ **Věta 11.2** *Vlastnosti značeného grafu.*

Nechť $N = (P, T, F, W, K, M_0)$ je značený graf.

1. N je *živý* právě tehdy, když každý cyklus grafu N obsahuje alespoň jednu značku při značení M_0.
2. N je *bezpečný* právě tehdy, když N je živý a každé místo $p \in P$ patří do cyklu, který při počátečním značení obsahuje právě jednu značku.
3. Značení M je dostupné ze značení $M' \in [M_0\rangle$ právě tehdy, je-li počet značek v každém cyklu grafu N stejný v obou značeních M i M'.

□

Pro analýzu značených grafů existuje metoda, která snižuje časovou náročnost operacemi redukcí na značeném grafu [14], jež zachovávají analyzované vlastnosti.

11.2.3 Petriho sítě s volným výběrem

Perspektivní podtřídou Petriho sítí, které umožňují modelovat konflikty i paralelismus, i když v omezeném rozsahu, jsou tzv. Petriho sítě s volným výběrem.

DEF

■ **Definice 11.4** *Petriho síť s volným výběrem.*

Petriho síť $N = (P, T, F, W, K, M_0)$ se nazývá *Petriho sítí s volným výběrem* (angl. *free-choice Petri net*), jestliže splňuje podmínku:

1. $\forall \langle p, t\rangle \in F \cap (P \times T) : p^\bullet = \{t\} \vee {}^\bullet t = \{p\}$
2. $\forall z \in F : W(z) = 1$

□

Podmínku (1) lze vyjádřit i jiným způsobem, jak je uvedeno v následující větě.

Věta

■ **Věta 11.3** *Vlastnosti sítě s volným výběrem.*

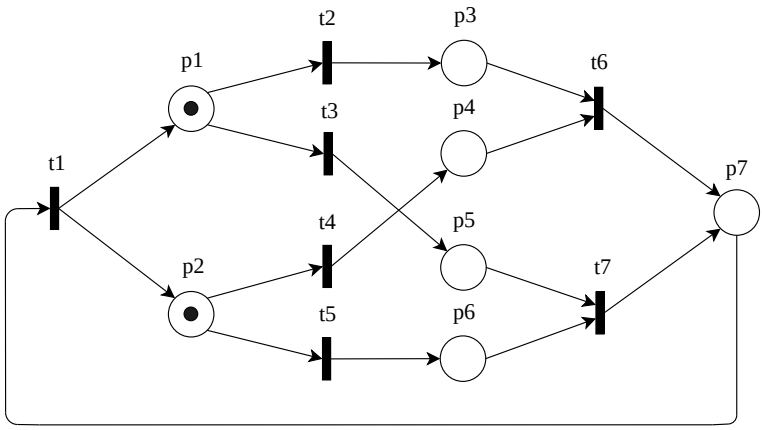
Nechť $N = (P, T, F, W, K, M_0)$ je Petriho síť s volným výběrem. Pak následující vlastnosti jsou ekvivalentní:

1. $p \in P \wedge |p^\bullet| > 1 \Rightarrow \forall t \in {}^\bullet p : {}^\bullet t = \{p\}$
2. $p_1, p_2 \in P \wedge p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow \exists t \in T : p_1^\bullet = p_2^\bullet = \{t\}$

3. $p \in P \wedge |p^\bullet| > 1 \Rightarrow {}^\bullet(p^\bullet) = \{p\}$

Důkaz této věty ponecháme jako cvičení. □

Příklad Petriho sítě s volným výběrem je uveden na obrázku 11.2.

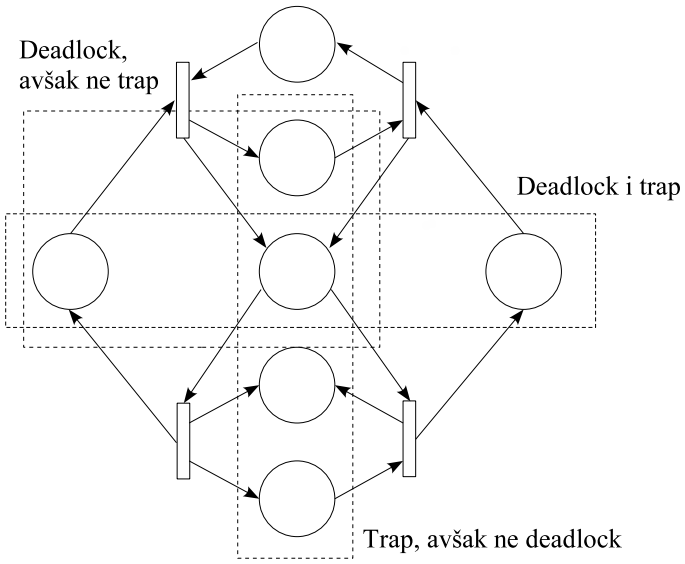


Obrázek 11.2: Petriho síť s volným výběrem

Význam uvedených podmínek je v tom, že v Petriho síti, ve které jsou splněny, můžeme řešení konfliktů jistým způsobem řídit. Skutečně, jestliže určité místo je vstupním místem pro několik přechodů (je možným zdrojem konfliktu), pak všechny tyto přechody mají pouze jeden výstup (jediné výstupní místo). Tudíž, buď jsou proveditelné, při značení M, všechny konfliktní přechody, nebo žádný z nich. To pak umožňuje svobodný výběr přechodu z množiny konfliktních přechodů; přítomnost jiných značek v jiných místech nemusí být v tomto výběru brána v úvahu. Na obrázku 11.2 je to případ místa p_1 a přechodů t_2, t_3 nebo místa p_2 s přechody t_4, t_5.

Pro analýzu živosti Petriho sítí s volným výběrem mají důležitou úlohu dva typy podmnožin množiny míst. Podmnožina Π, kde $\Pi \subseteq P$, nazývaná *deadlock*, je definovaná inkluzí ${}^\bullet\Pi \subseteq \Pi^\bullet$ a je tvořena místy, která, jakmile ztratí značky, nemohou je už nikdy získat. Podobně, podmnožina Π, $\Pi \subseteq P$, nazývaná *trap (past)*, je definována inkluzí $\Pi^\bullet \subseteq {}^\bullet\Pi$ a je tvořena místy, která, jakmile získají značky, nemohou je již nikdy ztratit. Příklady těchto podmnožin jsou uvedeny na obrázku 11.3.

Systematické vyhledání uvedených podmnožin míst je algoritmizovatelné. Vlastní analýza živosti je pak založena na následujícím poznatku, který uvedeme bez důkazu.



Obrázek 11.3: Význačné podmnožiny míst - deadlock, trap

Věta

■ **Věta 11.4** *Živost sítě s volným výběrem.*

Petriho síť N s volným výběrem je živá právě tehdy, jestliže každý neprázdný deadlock sítě N obsahuje trap, který je označen v počátečním značení sítě N.

Důkaz. Důkaz lze nalézt např. v [9]. □ □

11.3 Rozšíření Petriho sítí

Existuje celá řada rozšíření Petriho sítí, která zvyšuje jejich modelovací mocnost až na úroveň Turingových strojů. Tato rozšíření úzce souvisejí s možností „testovat nulu“, tj. s možností podmínit provedení přechodu sítě absencí značky v některém místě. Tato rozšíření jsou v důsledku Turingovy téze ekvivalentní a proto se seznámíme pouze s jedním rozšířením, které je jednoduché a velmi rozšířené, s Petriho sítěmi s inhibitory.

11.3.1 Petriho sítě s inhibitory

DEF

■ **Definice 11.5** *Petriho síť s inhibitory.*

Petriho síť $N = (P, T, F, W, K, M_0)$ se označuje jako *Petriho síť s inhi-*

bitory (s inhibičními hranami), pokud množina F obsahuje neprázdnou podmnožinu F^{I} hran, pro které

$$F^{\mathsf{I}} \subseteq F \cap (P \times T), \forall f \in F^{\mathsf{I}} : W(f) = 1$$

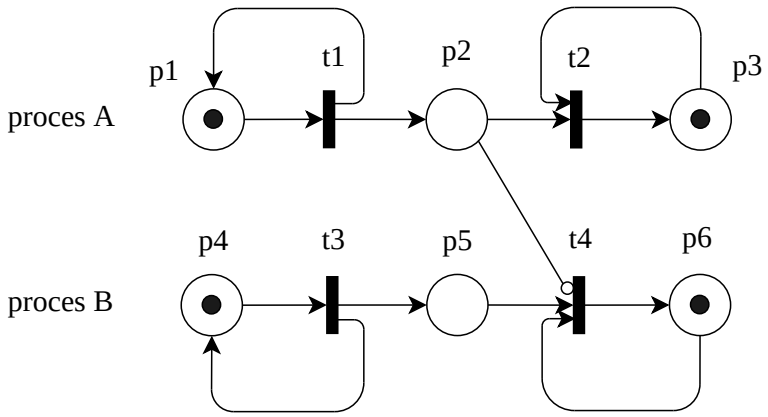
a které modifikují pravidlo pro provedení přechodu takto:

Přechod $t \in T$ je proveditelný při značení $M : p \to \mathbb{N}$, jestliže

$$\forall p \in {}^\bullet t : \begin{cases} M(p) \geq W(p,t), & \text{jestliže } \langle p,t\rangle \in F \setminus F^{\mathsf{I}} \\ M(p) = 0, & \text{jestliže } \langle p,t\rangle \in F^{\mathsf{I}} \end{cases}$$

Množina F^{I} se nazývá *množinou inhibičních hran* nebo krátce *množinou inhibitorů*. □

Inhibitor ovlivňuje pouze proveditelnost přechodu, ne však změnu následného značení. Je-li $\langle p,t\rangle \in F^{\mathsf{I}}$ a $M\ [t\rangle\ M'$, pak $M'(p) = 0$, pokud $p \notin t^\bullet$. Graficky je inhibitor vyznačován jako hrana, která není ukončena šipkou, ale malým kroužkem (konvence pocházející z logických systémů). Na obrázku 11.4 je příklad použití Petriho sítě s inhibitorem při modelování priority. Operace procesu A modelovaná přechodem t_2 má přednost před operací t_4 procesu B.



Obrázek 11.4: Petriho síť s inhibitorem

Nyní ukážeme, že výpočetní síla Petriho sítí s inhibitorem je skutečně ekvivalentní se silou Turingových strojů. K tomuto účelu využijeme ekvivalence Turingových strojů s tzv. registrovými stroji [15]. Registrový stroj je počítač, jehož paměť je tvořena řadou registrů, které slouží k uchování neomezeně velkých čísel. Program je posloupnost adresovatelných instrukcí typu „inkrementuj registr“, „přejdi k instrukci s, je-li obsah registru n nulový“ apod. V [15] je dokázáno, že registrový stroj s instrukcemi:

- $I(n)$: Zvětši obsah registru n o jedničku
- $D(n)$: Zmenši obsah registru n o jedničku (obsah n je nenulový)
- $J(n)[s]$: Přejdi k příkazu s, je-li obsah n roven nule

je ekvivalentní Turingovu stroji. Dokážeme-li tedy, že registrový stroj s těmito instrukcemi může být převeden na Petriho síť s inhibitory, pak jsme dokázali také ekvivalenci Petriho sítí s inhibitory s Turingovými stroji.

Abychom reprezentovali registrový stroj Petriho sítí, reprezentujeme m použitých registrů místy $p'_1, p'_2, \ldots, p'_m$. Dalších $r + 1$ míst $p_0, p_1, \ldots, p_r$, kde $M(p_i) \in \{0, 1\}$, použijeme pro implementaci čítače instrukcí programu tvořeného r instrukcemi. Místo p_0 (označené v počátečním značení) reprezentuje hodnotu čítače před provedením první instrukce programu.

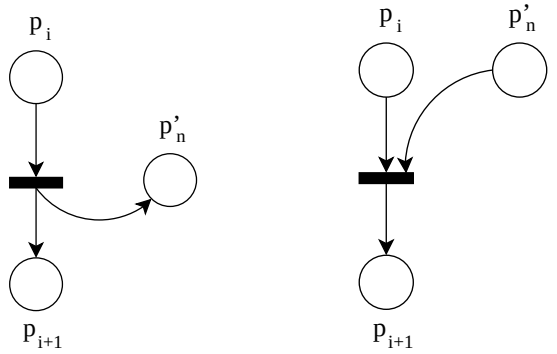
Na obrázku 11.5 je uvedena reprezentace všechny tří postačujících příkazů registrového stroje pomocí přechodů Petriho sítě s inhibitory.

Zavedení inhibičních hran tedy zvyšuje modelovací mocnost Petriho sítí na nejvyšší možnou mez (Turingova teze). Ztrácíme však, bohužel, možnost analyzovat tyto sítě, protože většina problémů analýzy vede ke známým nerozhodnutelným problémům Turingových strojů.

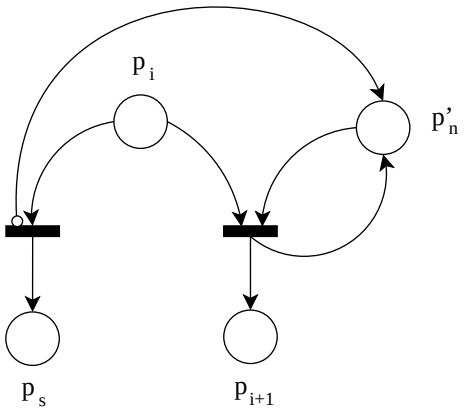
Na závěr naší diskuse různých tříd Petriho sítí se zmíníme o dalších modifikacích, které dokumentují nejenom bohatost a rozmanitost této třídy modelů, ale které mají rovněž důležité místo v praktických aplikacích.

11.3.2 Časové a časované Petriho sítě

První z nich jsou tzv. *časové Petriho sítě* (angl. *time Petri nets*) [16] a jejich speciální případ *časované Petriho sítě* (angl. *timed Petri nets*) [17]. Tato třída Petriho sítí je rozšířením klasických Petriho sítí o možnost popisu časových vztahů mezi operacemi v modelovaném systému. Každý přechod t časové Petriho sítě je atributován dvěma nezápornými reálnými čísly a a b, $a \leq b$ reprezentujícími relativní časové hodnoty vztažené k okamžiku, kdy se stává přechod t proveditelným. Hodnota a značí minimální čas, který musí uplynout, aby přechod t mohl být proveden, a hodnota b je maximální čas, po který je přechod t proveditelný, aniž byl proveden. Tedy, stal-li se přechod t proveditelným v čase τ, pak může být proveden v časovém intervalu $\langle \tau+a, \tau+b \rangle$, pokud ovšem v tomto intervalu nedošlo ke změně značení, která vylučuje provedení přechodu t. V případě časovaných Petriho sítí se neuvažuje „počáteční zpoždění“, takže přechod je ohodnocen pouze délkou intervalu, po který je přechod proveditelný (tj. $a = 0$).



instrukce $I(n)$ instrukce $D(n)$



instrukce $J(n)[s]$

Obrázek 11.5: Reprezentace instrukcí registrového stroje

11.3.3 Stochastické Petriho sítě

Příbuzným modelem k časovým sítím jsou *stochastické Petriho sítě* [18], které ohodnocují přechody sítě stochastickým atributem, reprezentujícím délku trvání operace. Na rozdíl od časových Petriho sítí, jejichž analýza je založena na pojmech diskutovaných v kapitole 8, analýza stochastických Petriho sítí se opírá o teorii stochastických procesů. Tyto sítě se uplatňují zejména při modelování a analýze výkonnosti systémů.

11.3.4 Petriho sítě vyšší úrovně

K velmi významným a pozoruhodným výsledkům vývoje síťových modelů patří tzv. *Petriho sítě vyšší úrovně* (angl. *higher-level Petri nets*) [19]. Na rozdíl od P/T Petriho sítí umožňují Petriho sítě vyšší úrovně kategorizovat a individualizovat jednotlivé značky značení míst a sta-

novit podmínky provádění přechodů respektující atributy těchto značek. K nejvíce rozšířeným třídám Petriho sítí vyšší úrovně patří *Pr/T Petriho sítě*(Predicate/Transitions Petri nets), jejich podtřída, *barvené Petriho sítě* (angl. *coloured Petri nets*) a zvláště pak *hierarchické barvené Petriho sítě*, které navíc umožňují explicitně strukturovat graf Petriho sítí. Charakterizujme alespoň základní složky této třídy modelů.

Model systému ve tvaru hierarchické barvené Petriho sítě se skládá ze dvou složek: *grafické síťové struktury* a její *formální inskripce*. Tato inskripce obvykle zahrnuje:

1. *Množiny barev* specifikující typy značek v modelu. Značky představují jednoduché a strukturované datové objekty, např. n-tice, záznamy, seznamy.

2. *Počáteční značení* specifikující počáteční multimnožinu značek v každém místě.

3. *Hranové výrazy* specifikující multimnožiny značek, které jsou, v průběhu evoluce sítě, odebírány, resp. přidávány ze vstupních, resp. do vstupních míst. Tyto výrazy mohou obsahovat proměnné datových typů (barev), booleovské selektory a obvyklé algebraické operátory. Při provádění přechodu jsou proměnné výrazu „navázány“ na konkrétní hodnotu značení příslušných míst.

4. *Strážní podmínky* (angl. *guards*) omezující hodnoty, na které mohou být volné proměnné výrazu navázány.

5. *Procedury* (kódové segmenty) spojené s přechody sítě, jež mohou provádět složité výpočty s datovými objekty reprezentovanými značkami včetně definovaných vedlejších efektů (čtení, generování výstupů).

Hierarchická struktura sítě, podporující modulární výstavbu modelu, je dosažena možností definice podsítí (nazývaných *stránky*) a mechanismů jejich náhrady za uzly nadřazené sítě. Tyto mechanismy zahrnují *substituci* a *invokaci přechodu*, *substituci místa* a *sloučení míst*, které umožňuje sloučit vícenásobné výskyty téhož místa do jediného.

Pro Petriho sítě vyšší úrovně existují metody analýzy analogické P/T sítím (výpočet stromu dosažitelných značení a invariantů sítě). Jejich velkou výhodou je značná redukce rozsahu sítě (ve srovnání s odpovídající P/T sítí). Výhody Petriho sítí vyšší úrovně proti klasickým P/T sítím jsou často porovnávány s výhodami a možnostmi vyšších programovacích jazyků ve srovnání s jazyky symbolických instrukcí.

Pojmy k zapamatování

- stavový stroj, značený graf
- Petriho síť s volným výběrem
- Petriho síť s inhibitory

Shrnutí

V této kapitole jsme se seznámily s některými modifikacemi, které představují zúžení a rozšíření základní třídy Petriho sítí. Nejjednodušší podtřídou Petriho sítí jsou stavové stroje, které jsou ekvivalentní s konečnými automaty. Další podtřídou Petriho sítí jsou značené grafy. Perspektivní podtřídou Petriho sítí, které umožňují modelovat konflikty i paralelismus, i když v omezeném rozsahu, jsou Petriho sítě s volným výběrem. Existuje celá řada rozšíření Petriho sítí, která zvyšuje jejich modelovací mocnost až na úroveň Turingových strojů. Mezi ně patří především Petriho sítě s inhibitory. Zavedení inhibičních hran zvyšuje modelovací mocnost Petriho sítí na nejvyšší možnou mez, ztrácíme však možnost analyzovat tyto sítě, protože většina problému analýzy vede ke známým nerozhodnutelným problémům Turingových strojů.

Příklady k procvičení

1. Definujte stavový stroj a porovnejte jeho modelovací mocnost s konečnými automaty.
2. Definujte cyklus značeného grafu a uveďte jeho vlastnosti.
3. Dokažte následující větu:

   Nechť N je Petriho síť s volným výběrem. Pak platí následující tvrzení: $p \in P \wedge \mid p\bullet \mid > 1 \Rightarrow \forall t \in p\bullet : \bullet t = \{p\}$.
4. Uveďte evoluční pravidla pro Petriho sítě s inhibitory.

Část III

Vysokoúrovňové Petriho sítě

Kapitola 12

Barvené Petriho sítě

Čas potřebný ke studiu: 7 hodin

Cíle kapitoly

Tato kapitola představí čtenářům základní koncepty barvených Petriho sítí. Tyto sítě umožňují úspornější modelování některých rysů běžných v reálných systémech, jež není snadné přímo popisovat pomocí klasických P/T Petriho sítí (jako např. práci se složitějšími datovými typy). V textu se zaměříme na základní úvod do syntaxe a sémantiky barvených Petriho sítí postačující pro využití těchto sítí při simulačním ověřování systémů – metody jejich formální analýzy jsou již nad rámec tohoto textu.

Průvodce studiem

Tato kapitola navazuje na předchozí kapitoly popisující koncepty P/T Petriho sítí, jejichž důkladné zvládnutí se v dalším textu předpokládá.

Obsah

12.1 Motivace barvených Petriho sítí

Zavedení *barvených Petriho sítí* (tj. *Coloured Petri Nets* – CPN) je motivováno snahou odstranit některé nevýhody klasických (P/T) Petriho sítí. P/T Petriho sítě poskytují efektivní primitiva pro popis synchronizace paralelních procesů (a tedy pro popis toku řízení v paralelních systémech), ovšem složitější datové manipulace se v nich popisují hůře. CPN proto rozšiřují P/T Petriho sítě

- o definici *množin barev* (jež jsou obdobou datových typů v klasických programovacích jazycích),
- o spojení jednotlivých značek s určitou *barvou* (tj. s určitou hodnotou příslušného datového typu) a
- o manipulace barev značek při průchodu přechody – pomocí tzv. *inskripčních jazyků*, blízkých běžným programovacím jazykům, lze vyjádřit *podmínky nad barvami* manipulovaných značek a tím omezit proveditelnost přechodů a současně je možné přechody spojit se *změnami barev* procházejících značek.

S ohledem na používané inskripční jazyky a na způsob jejich propojení s Petriho sítěmi můžeme rozlišit řadu různých *dialektů barvených Petriho sítí*. Pravděpodobně nejvíce rozšířený je ale zřejmě dialekt barvených Petriho sítí zavedený *Kurtem Jensenem* z university v Aarhusu v Dánsku v roce 1981. Tento dialekt využívá inskripční jazyk inspirovaný funkcionálním jazykem *SML* a je spojen s nástroji *DesignCPN* a *CPNtools*. Tyto nástroje zahrnují pokročilý editor barvených Petriho sítí, výkonné simulátory a také jistou podporu pro analýzu stavových prostorů (grafů dosažitelnosti) CPN. Tyto nástroje jsou spolu s řadou článků, manuálů a případových studií dostupné na stránkách `www.daimi.au.dk/CPnets/`. Navíc je k dispozici kvalitní třídílná monografie [32] zahrnující základní koncepty CPN, metody jejich formální analýzy i popis průmyslových studií, ve kterých byly CPN uplatněny. V následujícím textu budeme často na této monografii stavět a budeme-li dále mluvit o CPN, budeme jimi chápat CPN dle této monografie, nebude-li explicitně řečeno jinak.

Mezi ostatní dialekty CPN patří například barvené Petriho sítě spojené s komerčním nástrojem ExSpect `www.exspect.com`. Řadu dalších nástrojů lze pak nalézt v databázi dostupné na Internetu na adrese `www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html`. Různé dialekty CPN jsou ovšem principiálně dosti podobné a tudíž po seznámení se s jedním z nich, není obvykle problém přejít na používání jiného.

Zavedení barev umožňuje někdy velmi výrazně stručnější zápis modelu. Na druhou stranu je ovšem použití CPN nutné zvažovat rovněž z hlediska možné analýzy a verifikace systémů pomocí nich popsaných. Analýza nad CPN, a to zejména formální analýza, je komplikovanější než nad klasickými P/T Petriho sítěmi. Volba, zda užít P/T Petriho sítě či CPN, pak záleží na konkrétní situaci – rozhoduje komplikovanost modelu, existence překladače z nějakého vhodného vyššího modelovacího jazyka do P/T Petriho sítí (jež může nahradit použití mechanismů nabízených CPN), dostatečnost simulační analýzy nebo potřeba formální analýzy, dostupnost nástrojů pro formální analýzu apod.

Skutečnost, že rozhodnutí použít CPN může skutečně často převážit, lze dokumentovat řadou průmyslových studií, ve kterých byly CPN aplikovány (z nich mnohé jsou popsány na již zmíněných stránkách `www.daimi.au.dk/CPnets/` a v třetím díle monografie [32]). Patří mezi ně například:

- komunikační protokoly a sítě,
- software (části SW Nokia, bankovní transakce, distribuované algoritmy, ...),
- hardware,
- řídící systémy,
- vojenské systémy,
- ...

Podobně jako u P/T Petriho sítí existují také různá rozšíření CPN o *fyzický čas*. O ten (a o možnost stochastické simulace a sběr statistik o analyzovaných modelech jako jsou průměrná doba určitých transakcí, délka front apod.) jsou rozšířeny mj. i CPN spojené s nástroji Design-CPN a CPNtools. O fyzický čas a stochastickou simulaci je rozšířen také nástroj ExSpect. Formální analýza tzv. dobře formovaných (well-formed) barvených Petriho sítí se stochastickým časováním [35] je pak dostupná např. v nástroji GreatSPN `www.di.unito.it/~greatspn`. Tuto problematiku zde však nebudeme rozebírat do hloubky, zájemce odkážeme na příslušnou literaturu.

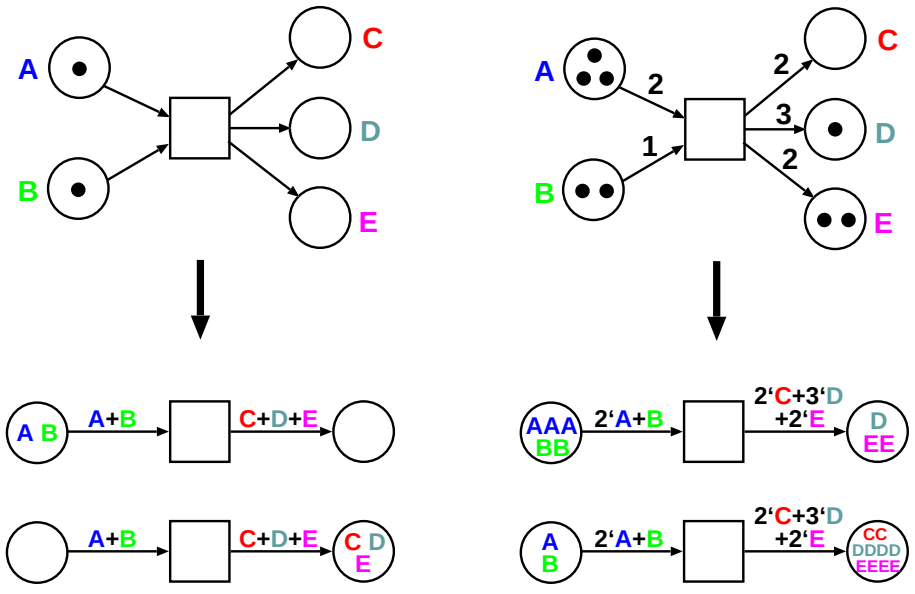
CPN jsou dále také základem pro další rozšíření, například různé způsoby *strukturování*. V sekci 13.1 si stručně přiblížíme tzv. *hierarchické CPN* a v sekci 13.2 dokonce i *objektově-orientované Petriho sítě*.

12.2 Zavedení barev do Petriho sítí

Nežli přejdeme k formální definici CPN, pokusíme se zavedení barev do Petriho sítí nejprve postupně ilustrovat neformálně na několika příkladech.

12.2.1 Základní idea

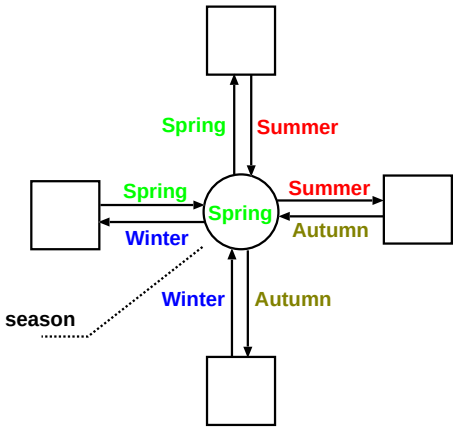
Na obrázku 12.1 máme příklad dvou P/T Petriho sítí a jejich možný převod do barvené Petriho sítě. Tento příklad odpovídá situaci, kdy pracujeme s výčtovým datovým typem s hodnotami A, B, C, D, E. Chceme-li si v P/T Petriho síti zapamatovat, kolik značek s příslušnou hodnotou máme, musíme pro každou hodnotu zavést patřičné místo, protože jednotlivé značky jsou v P/T Petriho síti nerozlišitelné. V CPN spojíme příslušnou hodnotu se značkou (tím značky odlišíme, individualizujeme) a můžeme je všechny uchovávat v jednom místě (v našem příkladu není úspora tak výrazná, ale představme si situaci, kdy pracujeme např. s intervalem 0..65536). Na hranách se pak musí objevit výrazy, které specifikují nejen *kolik značek* chceme odebrat či přidat z/do určitých míst, ale také o *jaké značky* se má jednat.



Obrázek 12.1: Individualizace (obarvení) značek v Petriho síti

Další podobný příklad je pak uveden na obrázku 12.2. Jedná se o velmi jednoduchý model systému změn ročních období. Všimněme si, že v jediném místě modelu bude vždy je jen jediná tečka, ale ta bude

obarvena podle příslušného ročního období. Přechody obklopující toto místo pak mohou roční období příslušným způsobem měnit.

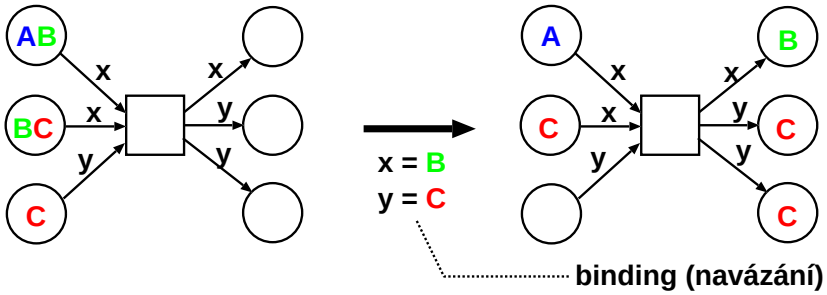


Obrázek 12.2: Jednoduchý model systému změn ročních období založený na CPN

Až doposud jsme na hranách uváděli pouze konstantní výrazy staticky specifikující kolik značek a jakých barev chceme přesunout z/do jednotlivých míst. Tento přístup je ale velmi často nepraktický. Velmi často chceme např. vyjádřit, že daný přechod se může provést pro libovolnou barvu ve vstupním místě, případně, že se může provést, pokud ve dvojici vstupních míst jsou značky libovolné, ale stejné barvy. Jsou-li množiny barev konečné, můžeme toto samozřejmě modelovat zavedením samostatného přechodu pro každou barvu. Je-li ale kardinalita množiny barev C, takových přechodů bude zapotřebí právě C, což může být značný počet. Budeme-li navíc chtít např. říci, že můžeme přesunout ze dvou vstupních míst libovolnou kombinaci dvou barev, budeme potřebovat již C^2 přechodů. Situace se podobá tomu, kdy bychom v běžných programovacích jazycích mohli v podmínkách pouze testovat proměnné na ekvivalenci s konstantami. To není příliš elegantní, a proto je přirozené, že v barvených Petriho sítích umožňujeme na hranách použití *proměnných*. Tato situace je ilustrována na obrázku 12.3.

navázání

Při provádění přechodu barvené Petriho sítě, na jehož hranách se vyskytují proměnné, se pak hledají tzv. *navázání* , tj. přiřazení barev jednotlivým proměnným, pro které je příslušný přechod proveditelný (tj. v příslušných vstupních místech je dostatečný počet značek zvolených barev). Vyskytuje-li se tatáž proměnná na více hranách v okolí stejného přechodu, musí být při provádění přechodu navázána tato proměnná na všech hranách na stejnou barvu.



Obrázek 12.3: Zavedení proměnných na hranách CPN

■ **Příklad 12.1**

x+y

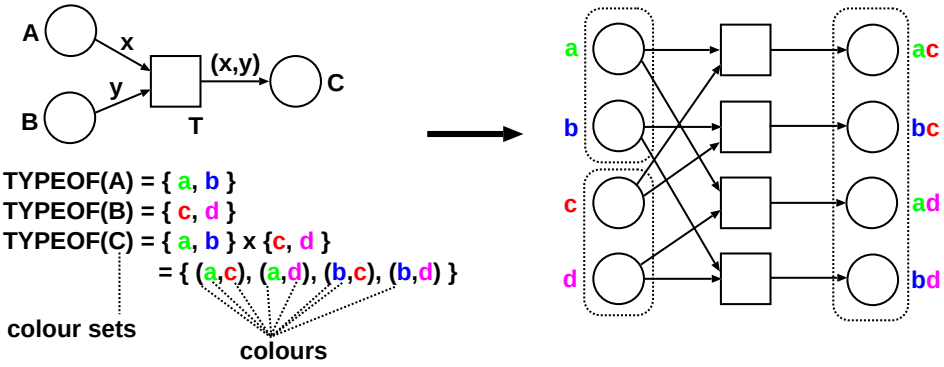
V síti na obrázku 12.3 je uvedený přechod proveditelný pro navázání $X = B, y = C$. Je proveditelný také pro nějaké jiné navázání? □

Zavedení konstant a proměnných na hranách barvených Petriho sítí představuje základ *inskripčního jazyka*, do jehož výrazů se přesouvá část modelu, jež by byl v P/T Petriho sítích kódován výhradně strukturou Petriho sítě. Možnosti inskripčního jazyka je pak možné dále zvyšovat zavedením *výrazů nad proměnnými*, zavedením *typování jednotlivých míst* (tj. přiřazením množin barev místům s tím, že se v nich pak mohou vyskytovat pouze značky příslušných barev) a přiřazením tzv. *stráží k přechodům* (tj. Booleovských podmínek nad proměnnými vyskytujícími se na hranách přechodu, jež omezují proveditelnost přechodu pouze na ta navázání, pro která se stráž vyhodnotí jako *true*).

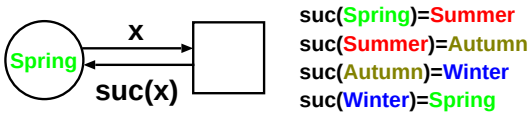
Ukázka použití typování míst a také jednoduchého výrazu konstruujícího dvojice z jednotlivých barev je v levé části obrázku 12.4. V pravé části tohoto obrázku je uvedena P/T Petriho síť odpovídající barvené síti z levé části obrázku. Na obrázku 12.5 je pak uveden jiný jednoduchý příklad použití hranových výrazů – změnu ročního období z obrázku 12.2 nyní modelujeme pomocí funkce *suc* (*successor*, následník) definované v pravé části obrázku.

12.2.2 Barvené Petriho sítě v nástroji DesignCPN

Pokusíme se nyní dále zpřesnit koncept barvených Petriho sítí, s nimž jsme se doposud seznámili. Zůstaneme stále ještě na neformální úrovni, ale přiblížíme se již blíže syntaxi a sémantice CPN tak, jak jsou používány v nástrojích DesignCPN a CPNtools. Použijeme přitom řadu příkladů převzatých z [32].



Obrázek 12.4: Zavedení typů míst a výrazů na hranách



Obrázek 12.5: Použití funkce v hranovém výrazu

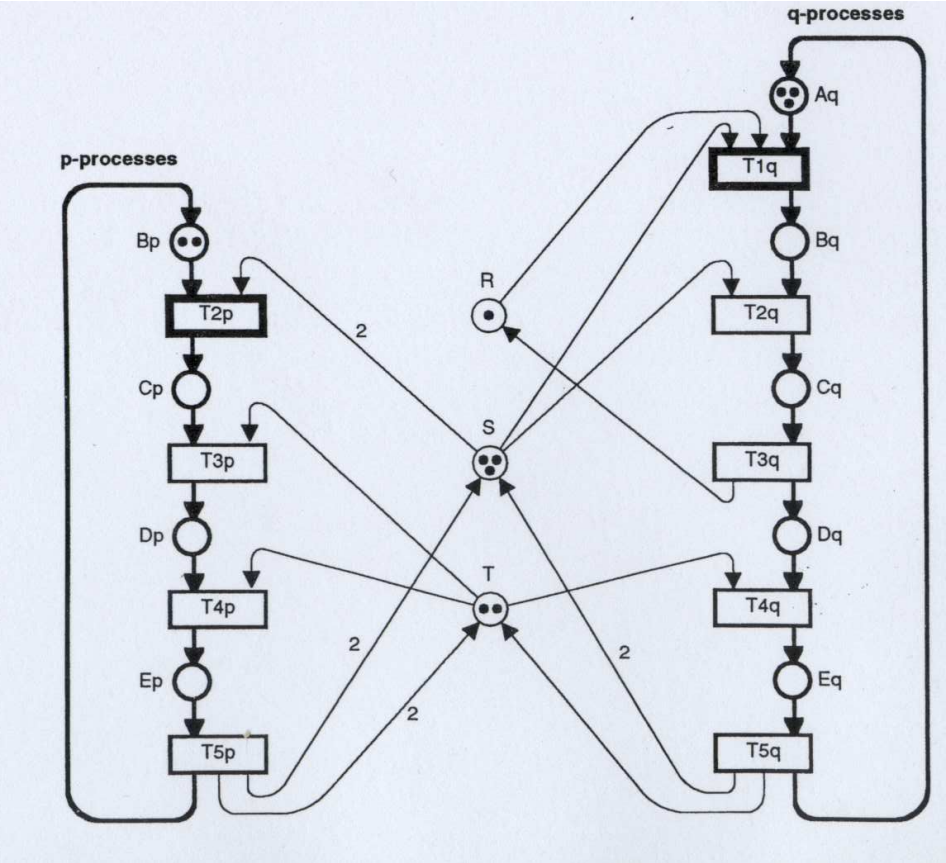
Uvažujme příklad popisu *systému přidělování prostředků* (zdrojů). Systém je tvořen:

- dvěmi typy procesů – procesy typu `p`, resp. `q`,
- třemi typy zdrojů – zdroje typu `R`, `S`, `T`,
- stavy procesů – `Bp`, `Cp`, ..., `Ep`, `Aq`, `Bq`, ..., `Eq`,
- počátečním stavem.

Vlastní činnost systému lze popsat *P/T Petriho sítí* tak, jak je ukázáno na obrázku 12.6. Vidíme, že zatímco výpočet procesů typu `p` prochází cyklicky čtyřmi fázemi modelovanými místy `Bp` až `Ep`, procesy typu `q` prochází cyklicky pěti fázemi modelovanými místy `Aq` až `Eq`. Změny stavu jsou modelovány přechody `T2p` až `T5p` a `T1q` až `T5q`, jejichž proveditelnost je vázána na dostupnost určitého počtu jednotek zdrojů typu `R`, `S` či `T`.

V CPN můžeme „sloučit“ popis chování značně podobných procesů `p` a `q`. Budeme registrovat, který průchod „alokačním cyklem“ daný proces provádí. Model ve tvaru *CPN*, jež je zachycen na obrázku 12.7, zahrnuje *dvě složky*:

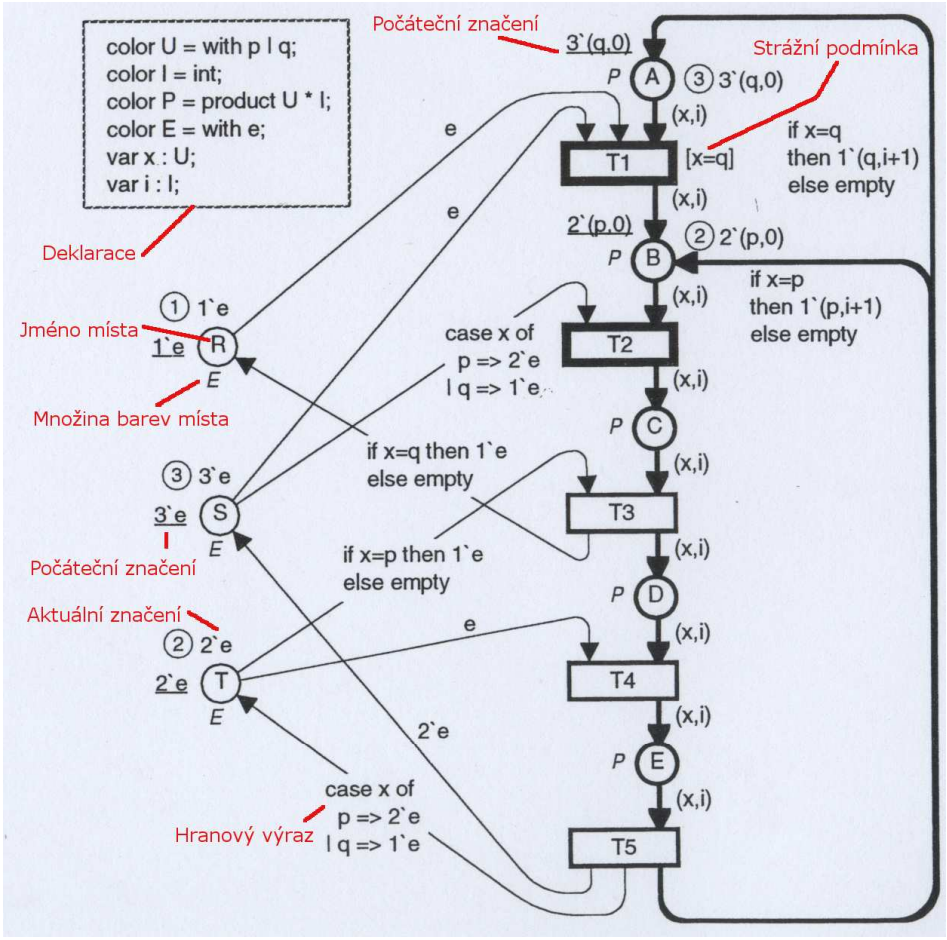
1. grafickou část – *graf Petriho sítě* a
2. popisy – *inskripci*.



Obrázek 12.6: P/T Petriho síť modelující jednoduchý systém sdílení prostředků [32]

Inskripce , vyjádřená inskripčním jazykem nástroje DesignCPN, jež je založen na SML, obsahuje následující elementy vyznačené na obrázku 12.7 červeně: inskripce

- deklaraci množin barev (colour sets), tj. datových typů, deklaraci typů jednotlivých proměnných hranových výrazů a případně deklaraci funkcí použitých v hranových výrazech či ve strážích přechodů (použito až v dalších příkladech),
- specifikaci množin barev míst,
- popis hran (hranové výrazy),
- strážní podmínky přechodů,
- počáteční značení,
- (jména míst a přechodů).



Obrázek 12.7: CPN modelující jednoduchý systém sdílení prostředků z obrázku 12.6 [32]

hranový výraz

Každý *hranový výraz* v CPN se vyhodnotí na *multimnožinu značek* (tj. soubor značek, ve kterém se jednotlivé značky mohou na rozdíl od množiny opakovat):

- konstruktor multimnožiny na hraně (nebo také v počátečním značení) je výraz $n_1`c_1 + n_2`c_2 + ...n_m`c_m$,
- $n_1, n_2, ..., n_m$ jsou konstanty, proměnné nebo funkce, které se vyhodnotí na přirozená čísla udávající počet značek,
- $c_1, c_2, ..., c_m$ jsou konstanty, proměnné nebo funkce, které se vyhodnotí na barvy značek.

x+y

■ **Příklad 12.2**

Vysvětleme si následující hranové výrazy:

- Hranový výraz `if x=C then 3‘D else 4‘E+5‘F` se vyhodnotí na multimnožinu obsahující tři značky s barvou D, je-li proměnná x navázána na barvu C, jinak se vyhodnotí jako multimnožina obsahující čtyři značky s barvou E a pět značek s barvou F.
- Hranový výraz `2‘(x+y)+3‘1` se vyhodnotí na multimnožinu obsahující dvě značky s hodnotou $x + y$ (operátor $+$ musí být samozřejmě definován nad množinou barev deklarovanou pro x a y – může se jednat např. o přirozená čísla o určitém rozsahu, kde $+$ se vyhodnocuje jako sčítání modulo rozsah dané množiny barev) a tři značky s barvou 1.

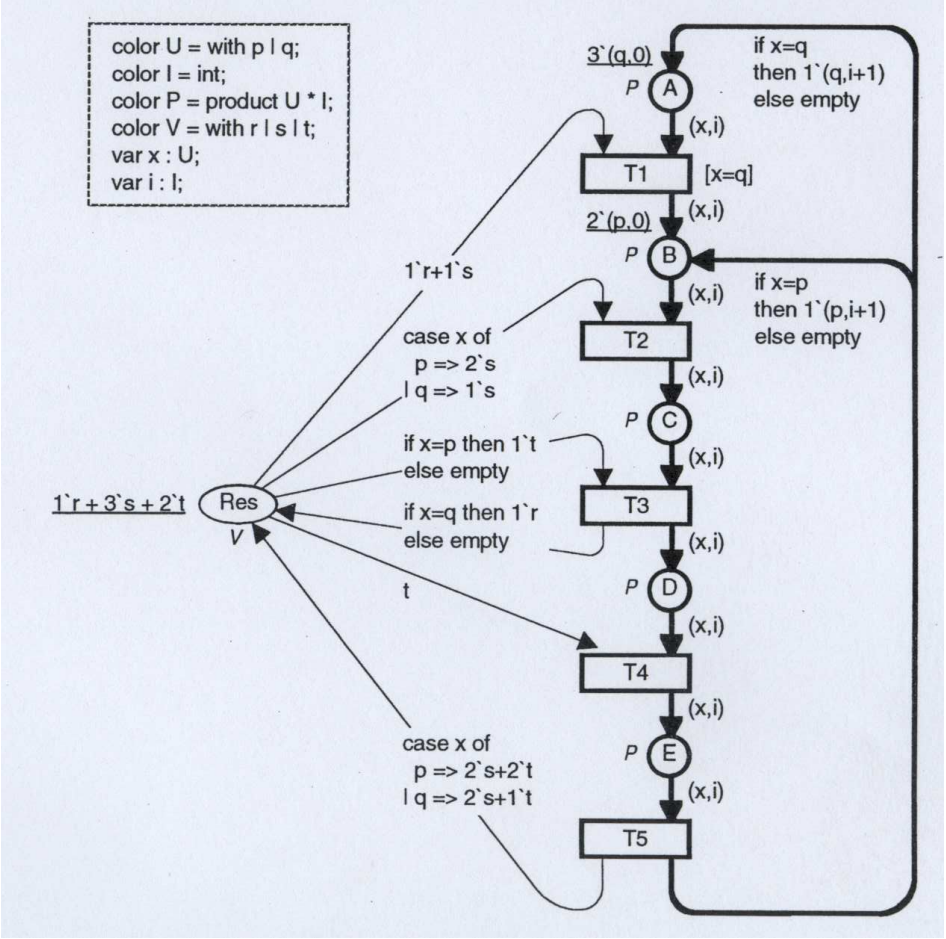
□

Na obrázku 12.7 jsou jednotlivé typy procesů rozlišeny barvou p nebo q z množiny barev $U = \{p, q\}$. Navíc jsou jednotlivé procesy očíslovány pomocí barvy $I = Int$ (podmnožina celých čísel s aritmetikou modulo rozsah této podmnožiny). Konkrétní procesy jsou pak tedy representovány barvami ve tvaru dvojice sestávající z p nebo q a čísla procesu, tj. patří do množiny barev $P = U \times I$ (na obrázku je použita syntaxe kartézského součinu dle DesignCPN). Množina barev E je zvláštní tím, že obsahuje jedinou barvu e, která se pak v modelu používá v roli „černé tečky“ známé z P/T Petriho sítí.

Na hranách v obrázku 12.7 si všimněme použití podmíněného výrazu `if` ...`then` ... `else` ... Ten se vždy vyhodnotí jako multimnožina, která ovšem může být i prázdná (klíčové slovo *empty*). Odebrání/přidání prázdné multimnožiny odpovídá neexistenci hrany v P/T Petriho síti až na to, že zde se užití prázdné množiny určí dynamicky, dle zvoleného navázání.

Po zavedení jiného systému barev a hranových výrazů můžeme náš systém sdílení zdrojů modelovat v CPN ještě úsporněji než na obrázku 12.7 – např. tak, jak je ukázáno na obrázku 12.8.

A konečně po zavedení ještě jiného systému barev a hranových výrazů můžeme náš systém sdílení zdrojů modelovat také tak, jak ukazuje obrázek 12.9. Tento příklad demonstruje skutečnost, že při použití CPN máme *volbu, které rysy systému popsat Petriho sítí a které výpočtem v použitém inskripčním jazyce*. Je na uživateli, aby mezi těmito dvěmi výrazovými prostředky nalezl vhodnou rovnováhu: přesune-li se převážná část modelu do inskripcí, může model ztratit přehlednost a použití CPN pro zápis takového modelu pak vlastně ztrácí smysl.



Obrázek 12.8: CPN modelující úspornějším způsobem než síť z obrázku 12.7 jednoduchý systém sdílení prostředků popsaný původně P/T Petriho sítí na obrázku 12.6 [32]

12.3 Formální definice syntaxe CPN

Zásadní roli pro definici sémantiky CPN hrají multimnožiny. Začneme proto formální definicí konceptu *multimnožiny* a operací nad ní.

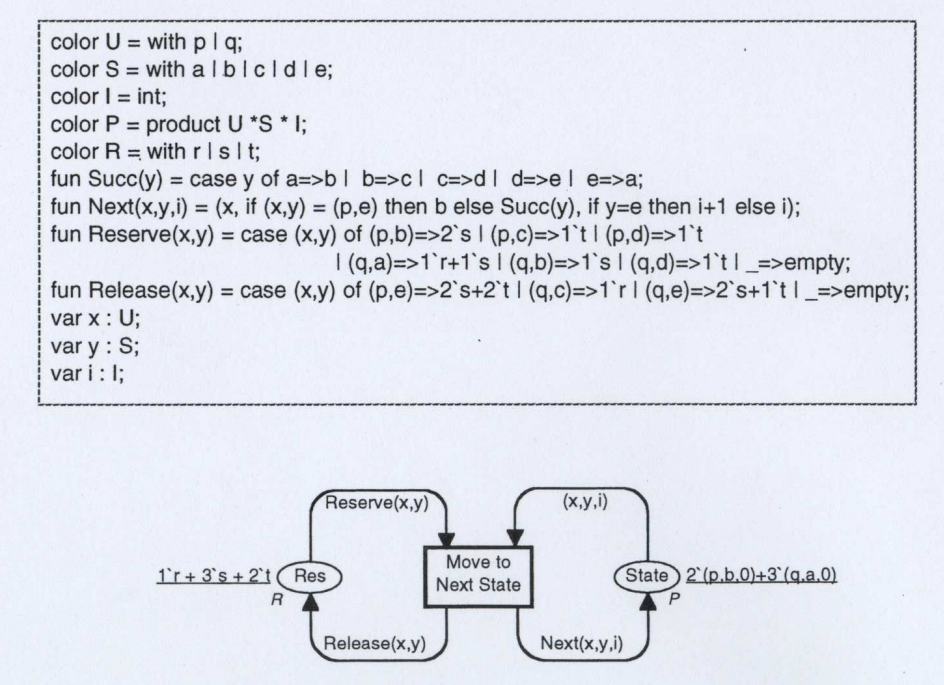
DEF

■ **Definice 12.1** *Multimnožina.*

Multimnožina m nad množinou S je funkce $m : S \to \mathbb{N}$, kde m(s) značí počet výskytů prvku s v multimnožině m. Multimnožinu m obvykle reprezentujeme formální sumou:

$$\sum_{s \in S} m(s)`s.$$

□



Obrázek 12.9: CPN modelující jednoduchý systém sdílení prostředků popsaný P/T Petriho sítí na obrázku 12.6 – většina modelu je nyní přesunuta z Petriho sítě do inskripcí [32]

Symbolem S_{MS} značíme *množinu všech multimnožin nad* S. Jestliže $m(s) \neq 0$, pak říkáme, že s *patří do* m a píšeme $s \in m$.

Pro multimnožiny jsou definovány následující operace a predikáty:

- operace *sjednocení* $m_1 + m_2 = \sum_{s \in S}(m_1(s) + m_2(s))‘s$,
- *skalární multiplikace* $c.m = \sum_{s \in S} c.m(s)‘s$ pro $c \in \mathbb{Z}$,
- predikáty $=$, $\neq$, $\leq$, $\geq$ (např. $m_1 \leq m_2 \Leftrightarrow \forall s \in S : m_1(s) \leq m_2(s)$),
- *kardinalita* $|m| = \sum_{s \in S} m(s)$ a
- je-li $m_1 \leq m_2$, pak také *rozdíl* $m_2 - m_1 = \sum_{s \in S}(m_2(s) - m_1(s))‘s$.

Při definici CPN bude vycházet z dané *konečné množiny* Σ *konečných množin barev* (Σ je tedy množina typů značek). *Typ proměnné* v budeme značit $Type(v)$. Je-li V množina proměnných, pak $Type(V) = \{Type(v) \mid v \in V\}$.

Typ výrazu $expr$ budeme značit $Type(expr)$. Množinu *proměnných výrazu* $expr$ budeme značit $Var(expr)$. *Uzavřeným výrazem* rozumíme

výraz *expr* bez proměnných, tj. $Var(expr) = \emptyset$. Označme $EXPR$ množinu všech výrazů přípustných ve zvoleném inskripčním jazyce. Označme dále $CEXPR \subseteq EXPR$ množinu všech uzavřených výrazů přípustných ve zvoleném inskripčním jazyce. Konečně označme $\mathbb{B} = \{\texttt{true}, \texttt{false}\}$.

DEF

■ **Definice 12.2** *Nehierarchická barvená Petriho síť.*

Nehierarchická barvená Petriho síť CPN je n-tice

$$CPN = (\Sigma, P, T, A, N, C, G, E, I),$$

kde:

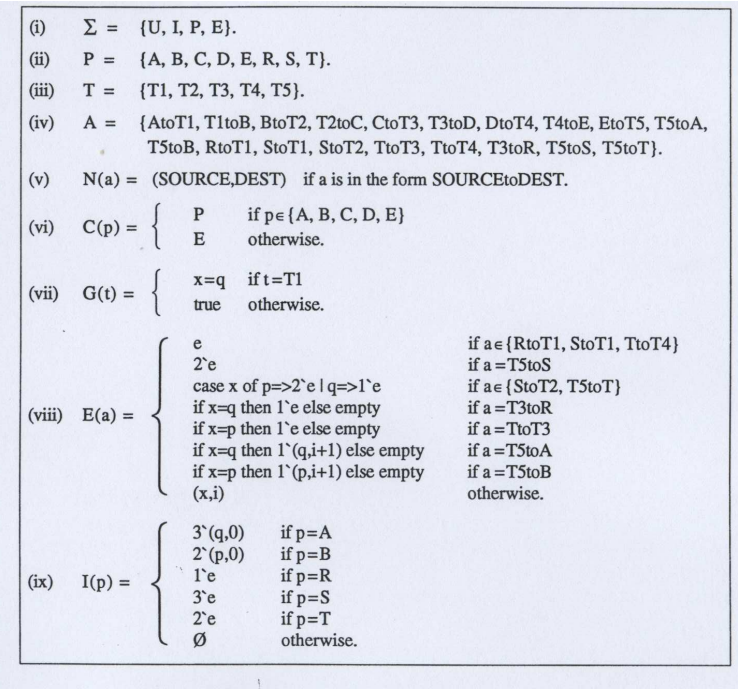
1. Σ je konečná množina konečných, neprázdných typů nazývaných *množinami barev*.
2. P je konečná množina *míst*.
3. T je konečná množina *přechodů* taková, že $P \cap T = \emptyset$.
4. A je konečná množina *hran* taková, že $A \cap P = A \cap T = \emptyset$.
5. N je *uzlová funkce* (node function) $N : A \to P \times T \cup T \times P$.
6. C je *funkce barev* (colour function) $C : P \to \Sigma$.
7. G je *funkce strážních podmínek* (guard function) $G : T \to EXPR$ taková, že:
$$\forall t \in T : Type(G(t)) = \mathbb{B} \wedge Type(Var(G(t))) \subseteq \Sigma.$$
8. E je *funkce hranových výrazů* (arc expression function) $E : A \to EXPR$ taková, že:
$$\forall a \in A : Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma,$$
přičemž $p(a)$ je místo v $N(a)$.
9. I je *inicializační funkce* $I : P \to CEXPR$ taková, že:
$$\forall p \in P : Type(I(p)) = C(p)_{MS}.$$

□

x+y

■ **Příklad 12.3**

Naši první CPN verzi systému alokace sdílených zdrojů z obrázku 12.7 můžeme dle definice zapsat tak, jak je ukázáno na obrázku 12.10. □



Obrázek 12.10: Množinový zápis CPN z obrázku 12.7 [32]

12.4 Formální definice sémantiky CPN

Základem pro definici sémantiky CPN je pojem navázání proměnných. Navázání přechodu pak můžeme definovat jako takové navázání proměnných, které respektuje jejich typování a které vede na splnění stráže přechodu (připomeňme si, že správné typování proměnných na hranách vůči typům míst, z/do kterých hrany vedou, je již součástí syntaxe CPN). Formálně můžeme tyto pojmy definovat takto.

DEF

■ **Definice 12.3** *Navázání.*

Navázání množiny proměnných z množiny V přiřazuje každému $v \in V$ prvek $b(v) \in Type(v)$.

Označme hodnotu získanou *vyhodnocením* výrazu *expr* při navázání b jako $expr\langle b \rangle$. Nechť $Var(t)$ je množina všech proměnných vyskytujících se ve výrazech na hranách přilehlých k přechodu t, respektive ve stráži t.

Navázání přechodu t je funkce b definovaná na $Var(t)$ tak, že:

1. $\forall v \in Var(t) : b(v) \in Type(v)$.
2. $G(t)\langle b \rangle$.

$B(t)$ pak značí množinu všech navázání přechodu t. □

Při definici značení vycházíme z pojmu element značení, což je dvojice sestávající z místa a barvy – sémantika je zřejmá: hovoříme o umístění určité značky v určitém místě. Později při definici provádění přechodu použijeme elementy značení také v tom smyslu, že do určitého místa se přidává, nebo z určitého místa se odebírá, určitá značka. Zavedení multimnožin nad elementy značení nám umožní elegantně pracovat s určitým počtem značek v určitém místě.

DEF

■ **Definice 12.4** *Značení.*

Element značení (token element) je dvojice (p, c), kde $p \in P$ a $c \in C(p)$. Množinu všech elementů značení značíme TE. *Značení* je pak multimnožina nad TE. Množinu všech značení značíme $\mathbb{M}$.

Počáteční značení M_0 je definováno vyhodnocením inicializační funkce: $\forall(p, c) \in TE : M_0((p, c)) = (I(p))(c)$. □

Pro definici provádění přechodů je klíčový pojem elementu navázání, což je dvojice přechod a jeho navázání. Zavedení elementu navázání nám umožní definovat sémantiku se skutečným paralelismem (*true concurrency* – odpovídá multiprocesingu v termínech běžně známých z operačních systémů), kdy se skutečně současně (nikoliv prokládaně – jako v multitaskingu) provádí obecně více přechodů pro více navázání (nebo též vícekrát tentýž přechod pro totéž navázání). Takovou multimnožinu elementů navázání nazýváme krokem CPN.

DEF

■ **Definice 12.5** *Krok.*

Element navázání (binding element) je dvojice (t, b), kde $t \in T$ a $b \in B(t)$. Množinu všech elementů navázání značíme BE.

Krokem pak rozumíme neprázdnou a konečnou multimnožinu nad BE. Množinu všech kroků značíme $\mathbb{Y}$. □

Nyní již můžeme završit naši definici sémantiky CPN zavedením toho, kdy je krok proveditelný v určitém značení a k jaké změně značení dojde při jeho provedení. (Připomeňme si, že požadavek na správné typování a splněnost stráže je zahrnut již do pojmu navázání přechodu.)

DEF

■ **Definice 12.6** *Proveditelnost kroku.*

Krok $Y \in \mathbb{Y}$ *je proveditelný (enabled) ve značení* $M \in \mathbb{M}$, jestliže platí následující:

$$\forall p \in P : \sum_{(t,b)\in Y} E(p, t)\langle b\rangle \leq M(p).$$

□

Je-li Y proveditelný, pak:

- Říkáme, že každý $(t, b) \in Y$ je proveditelný, a také, že t *je proveditelný (pro navázání b).*
- Pro $(t_1, b_1), (t_2, b_2) \in Y, (t_1, b_1) \neq (t_2, b_2)$, říkáme, že $(t_1, b_1), (t_2, b_2)$ jsou *současně proveditelné* (concurrently enabled).
- Je-li $Y((t, b)) \geq 2$, říkáme, že (t, b) *je současně proveditelný sám se sebou.*
- Podobně je-li $|Y(t)| \geq 2$, říkáme, že t *je současně proveditelný sám se sebou.*

DEF

■ **Definice 12.7** *Provedení kroku.*

Je-li $Y \in \mathbb{Y}$ proveditelný v $M_1 \in \mathbb{M}$, může *být proveden a změnit tak značení na* $M_2 \in \mathbb{M}$ takové, že:

$$\forall p \in P : M_2(p) = (M_1(p) - \sum_{(t,b)\in Y} E(p,t)\langle b\rangle) + \sum_{(t,b)\in Y} E(t,p)\langle b\rangle).$$

□

Pojmy k zapamatování

- inskripce, množina barev, popis hran, strážní podmínky přechodů
- navázání proměnných

Shrnutí

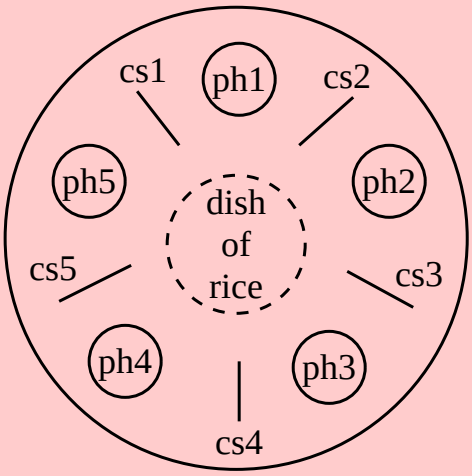
Tato kapitola si kladla za cíl definovat Barvené Petriho sítě. Model ve tvaru CPN zahrnuje graf Petriho sítě a inskripci. Inskripce, vyjádřená inskripčním jazykem, obsahuje deklaraci množin barev, specifikaci množin barev míst, popis hran, strážní podmínky přechodů a počáteční značení. Základem pro definici sémantiky CPN je pojem navázání proměnných.

Příklady k procvičení

1. Zkonstruujte barvenou Petriho síť modelující dva procesy procházející sdílenou kritickou sekcí chráněnou semaforem. Semafor odpovídá nezápornému celočíselnému čítači, jehož hodnotu mohou ovlivnit dvě operace: P a V. Operace V zvyšuje hodnotu semaforu, kdežto operace P ji (pokud je to možné) snižuje. Dále ukažte, že tentýž systém je možné bez problémů popsat i P/T Petriho sítí. Kterému z obou formalismů bychom v takovéto situaci měli dát přednost a proč?

2. Uvažte systém pěti čínských filozofů sedících kolem kulatého stolu. Uprostřed stolu je miska s rýží (dish of rice) a mezi každými dvěmi filozofy je jedna hůlka (chopstick). Každý filozof buďto přemýšlí, a nebo jí. Aby filozof mohl jíst, potřebuje dvě hůlky, přičemž může použít pouze nejbližší dvě (po své levé a pravé ruce). Chce-li filozof získat hůlky, nejprve uchopí levou hůlku a pak pravou. Po jídle filozof odkládá hůlky v opačném pořadí.

   Popište tento systém pomocí barvené Petriho sítě, která obsahuje dva datové typy (color sets) $PH = \{ph_1, ph_2, \ldots, ph_5\}$ a $CS = \{cs_1, cs_2, \ldots, cs_5\}$, které reprezentují filozofy a hůlky.



Kapitola 13

Strukturování barvených Petriho sítí

Čas potřebný ke studiu: 5 hodin

Cíle kapitoly

Tato kapitola představí čtenářům zejména základy možností hierarchického strukturování modelů založených na barvených Petriho sítích (podobné principy je ale samozřejmě možné užít i u klasických P/T Petriho sítí). Na závěr se velmi stručně zmíníme i o strukturování objektově orientovaném. Podobně jako u běžných programovacích jazyků je strukturování Petriho sítí nezbytné při práci s většími systémy.

Průvodce studiem

Tato kapitola navazuje na předchozí kapitoly popisující koncepty P/T Petriho sítí a CPN, jejichž zvládnutí se v dalším textu předpokládá.

Obsah

13.1 Hierarchické strukturování CPN

Modelování a návrh rozsáhlých systémů jsou samozřejmě nemyslitelné na úrovni jednoho, „plochého“ modelu. V CPN se zavádí čtyři klasické principy hierarchického strukturování modelů [33, 32]:

- *substituce přechodů*,
- *substituce míst*,
- *invokace přechodů* a
- *fúze míst*.

Tyto principy nejsou přitom omezené pouze na doménu CPN a dají se užít např. i v kontextu P/T Petriho sítí. Na druhou stranu se kromě těchto čtyřech konceptů používají i jiné principy strukturování – např. algebraické strukturování pomocí různých operátorů typu sekvenční a paralelní kompozice (viz např. [34]), či více dynamické objektově orientované strukturování, kterého si velmi stručně všimneme v sekci 13.2.

V následujícím textu si popíšeme příslušné strukturní mechanismy pouze neformálně. Jejich formální definice jde nad rámec tohoto textu a je jí možné nalézt např. v [33, 32].

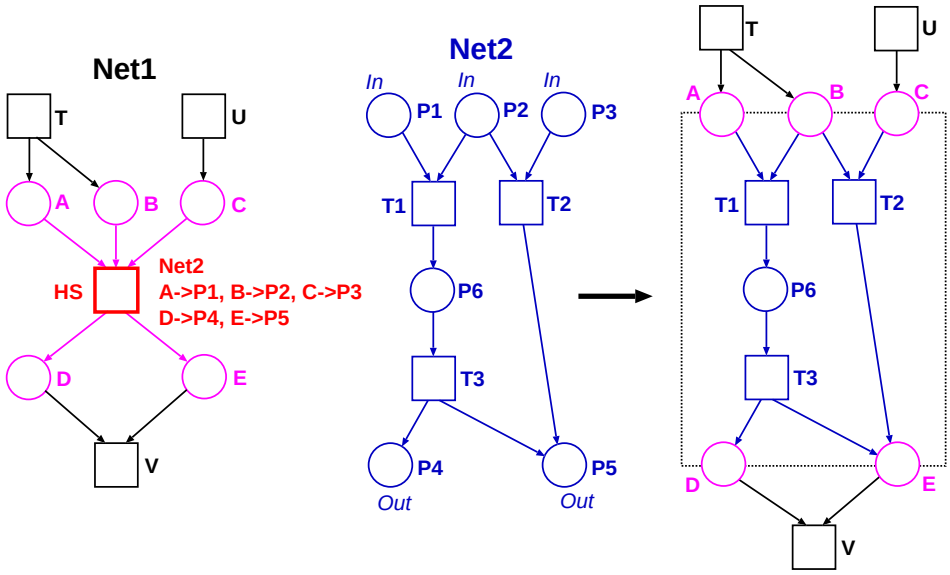
13.1.1 Substituce přechodů

Substituce přechodů je *statický mechanismus*, při kterém je substituovaný přechod nahrazen příslušnou podsítí podobně jako je tomu při rozvoji maker v běžných programovacích jazycích. Chceme-li tento mechanismus aplikovat na určitý přechod (jako je tomu např. na obrázku 13.1), postupujeme následovně:

1. Označíme přechod, za který chceme substituovat, jako *substituovaný přechod* (na našem obrázku je užito označení `HS`).
2. U substituovaného přechodu uvedeme, která *podsíť* bude za něj substituována (na našem obrázku je to síť `Net2`). Je zřejmé, že pro použití tohoto mechanismu musíme být tedy schopni jednotlivé sítě (v terminologii nástroje DesignCPN se nemluví o sítích, ale o tzv. *stránkách*) symbolicky pojmenovat.
3. U podsítě označíme její vstupní a výstupní místa (tzv. *porty*).
4. U substituovaného přechodu uvedeme, jak se mají *propojit* jeho vstupní a výstupní místa se vstupními a výstupními místy (porty) příslušné podsítě.

5. Musíme označit jednu síť jako *hlavní*, tj. tu v níž se mechanismus substituování zahájí.

Síť, ve které je užito substituce přechodů je možno samozřejmě *rozvinout* do ploché, nestrukturované sítě – cyklická substituce je zakázána. Na obrázku 13.1 máme v pravé části uveden příklad takového rozvoje.



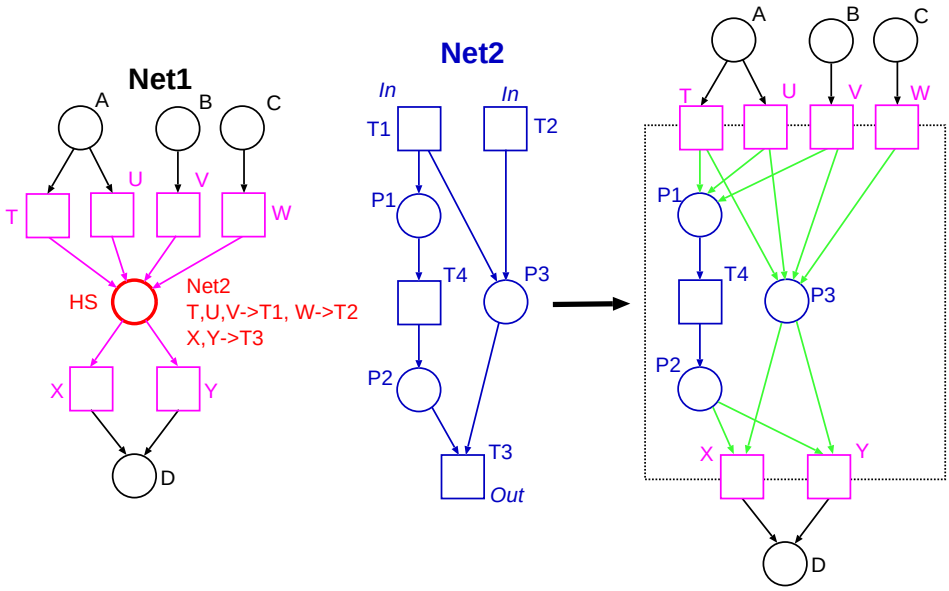
Obrázek 13.1: Substituce přechodů

13.1.2 Substituce míst

Substituce míst je mechanismus velmi podobný substituci přechodů, proto ho již zde nebudeme dále rozebírat. Čtenáři ale doporučujeme zamyslet se z cvičných důvodů nad příkladem substituce místa a jejího rozvoje, který je uveden na obrázku 13.2.

13.1.3 Invokace přechodů

Invokace přechodů je ze syntaktického hlediska velmi podobná substituci přechodů – viz obrázek 13.3. Sémanticky se ale jedná o něco naprosto odlišného. Zatímco substituce přechodů je *statická* a jako její paralelu jsme si uváděli rozvoj maker (které řeší překladač při překladu), invokace přechodů je záležitost *dynamická*, podobná volání procedur či funkcí v běžných programovacích jazycích. Invokaci přechodů nelze tedy (snadno) syntakticky odstranit a je zapotřebí s ní



Obrázek 13.2: Substituce míst

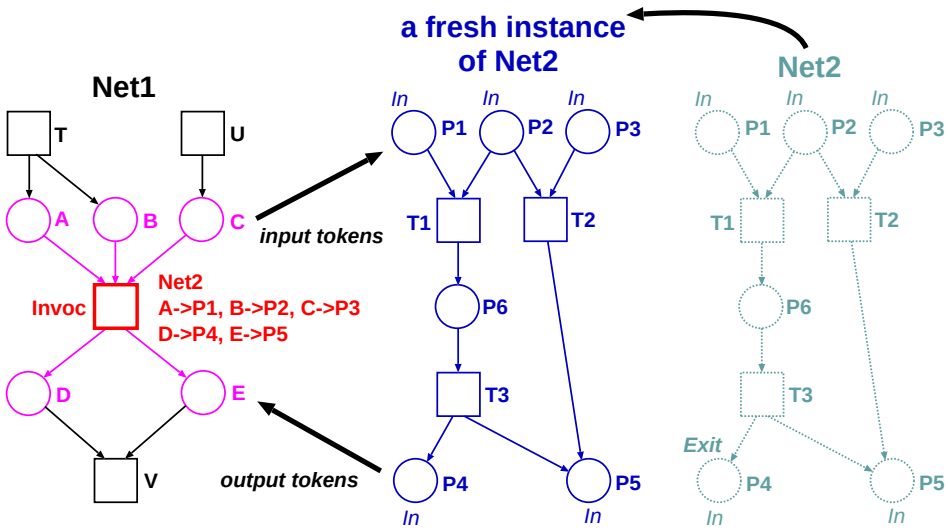
počítat při provádění příslušné sítě[1]: Při provedení příslušného přechodu se odebere jeho vstupní značení, vytvoří se instance invokované sítě a do ní se umístí její vstupní značení. Tato instance pak běží paralelně se všemi doposud vytvořenými instancemi až do okamžiku, kdy se objeví značení v jejich výstupních místech. Značení z výstupních míst instance sítě invokované přechodem je pak předáno síti, která tento přechod vyvolala. Invokace přechodů samozřejmě umožňuje i rekurzívní provázání jednotlivých sítí.

Mechanismus invokace bohužel není v běžných nástrojích pro práci s CPN (jako jsou DesignCPN či CPNtools) implementován. Stal ale základem návrhu objektově orientovaných Petriho sítí, které zmíníme v sekci 13.2.

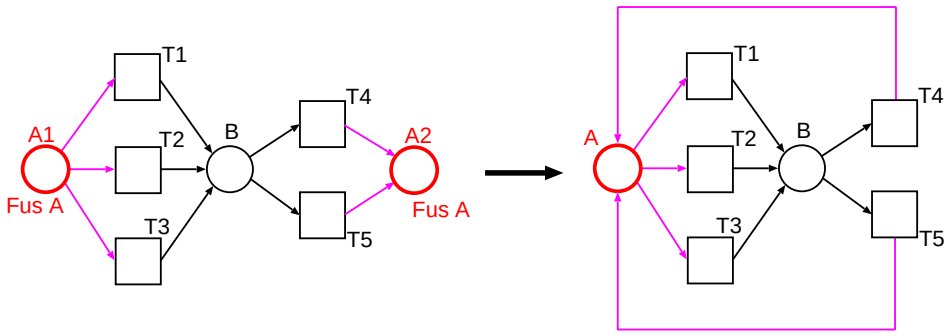
13.1.4 Fúze míst

Fúze míst je velmi praktický mechanismus, který umožňuje vytvářet množiny míst, které z hlediska sémantiky vytvořeného modelu budou sloučeny do místa jednoho. Tento mechanismus je ilustrován na obrázku 13.4 – fúze se zajistí tak, že se místa, která se mají sloučit, označí jako fúzní místa se stejným jménem cílového (tj. sloučeného) místa. Pomocí fúze míst je možné vyhnout se vytváření dlouhých a křížících se hran vedoucích z jedno okraje sítě do druhého.

[1]Invokaci lze odstranit odlišením všech značek v dané síti jejich rozšířením na dvojici obsahující číslo instance, které se bude automaticky generovat při novém vyvolání příslušného přechodu.



Obrázek 13.3: Invokace přechodů



Obrázek 13.4: Fúze míst

Při kombinaci fúze míst se substitucí přechodů či míst (případně s invokací přechodů) je možné zavést *různé typy fúze*:

- lokální v rámci jedné instance sítě,
- napříč všemi instancemi dané sítě a
- napříč všemi instancemi všech sítí.

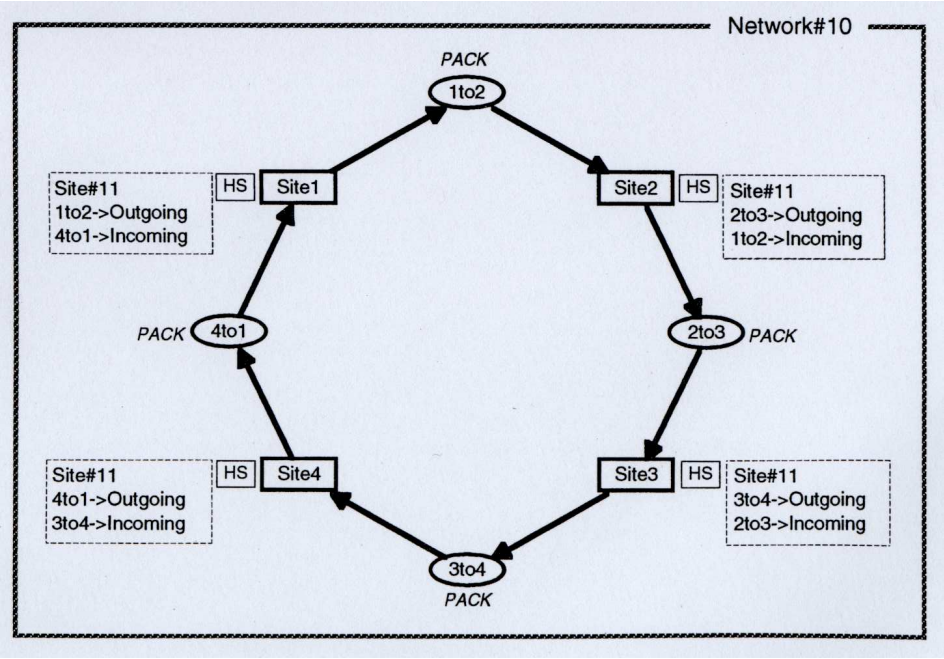
V případě nelokální fúze je samozřejmě role fúze míst výraznější nežli jen zabránění křížení míst – umožní propojit instance míst, které vzniknou až při rozvoji hierarchicky strukturované sítě do sítě „ploché“. (U fúze kombinované s invokací se jedná o ještě mocnější mechanismus, který vytváří obdobu globálních proměnných známých z běžných programovacích jazyků.)

x+y

■ **Příklad 13.1**

Kombinace fúze míst se substitucí přechodů je ilustrována na obrázcích 13.5 až 13.8. Aniž bychom zde zabíhali do detailů modelovaného systému, můžeme uvést, že se jedná o model komunikace v kruhové počítačové síti.

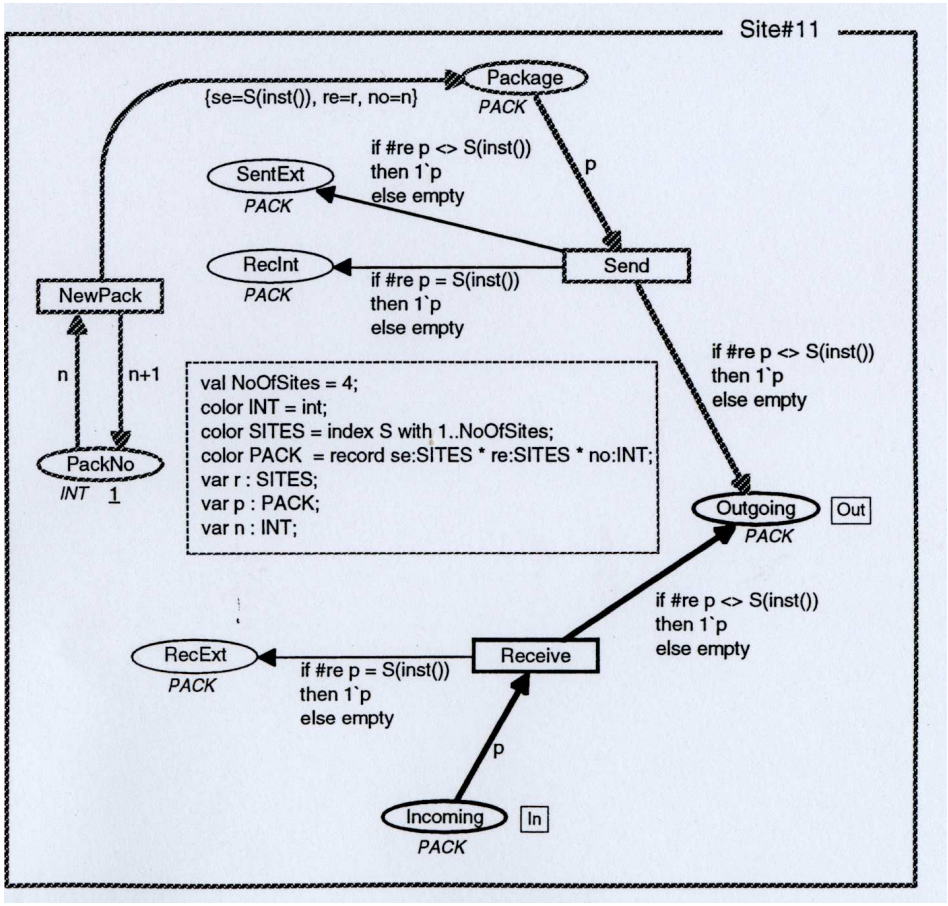
Hlavní síť modelu je uvedena na obrázku 13.5. Do ní se substituují sítě odpovídající jednotlivým stanicím v síti, jež jsou na nejvyšší úrovni modelu reprezentovány substitučními přechody `Site1` až `Site2` (místa mezi těmito přechody modelují komunikační médium). Podsíť, jejíž instance se substituují za zmíněné přechody, je uvedena na obrázku 13.6. Navíc předpokládáme, že mezi místy `RecInt` a `RecExt` je definována fúze v rámci instance sítě (vyhýbáme se nepřehlednému křížení hran) a že u místa `SentExt` je definována globální fúze. Globální fúze způsobí, že všechny čtyři vznikající instance sítě modelující komunikační stanici budou propojeny v místě `SentExt`.



Obrázek 13.5: Hlavní síť CPN modelu kruhové počítačové sítě [32]

Rozvinutím substitučních přechodů získáme CPN uvedenou na obrázku 13.7. Po provedení fúze míst pak získáme síť z obrázku 13.8.

□



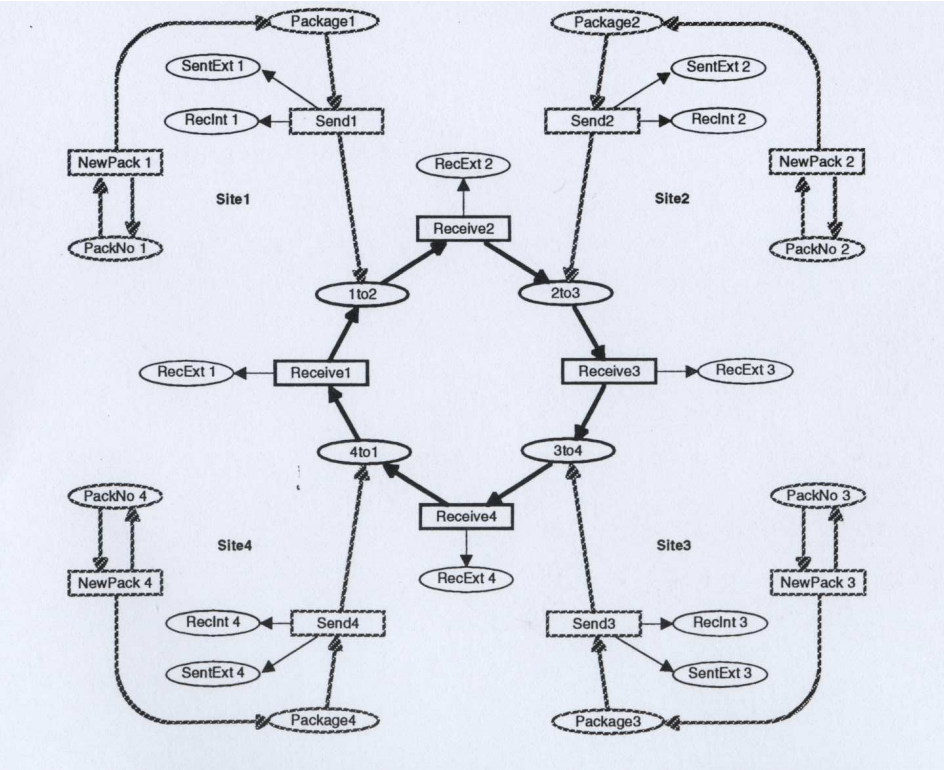
Obrázek 13.6: CPN model jedné stanice v kruhové počítačové sítě – jedná se o podsíť substituovanou do sítě z obrázku 13.5 [32]

13.2 Objektově orientované Petriho sítě

Existují různé koncepty (např. [36, 37, 38]) zavedení objektové orientace do Petriho sítí, které v zásadě staví na mechanismech podobných invokaci přechodů. Cílem je kombinace synchronizačních mechanismů nabízených Petriho sítěmi a moderního objektově orientovaného strukturování s dědičností, polymorfismem apod.

Jako příklad objektově orientovaných Petriho sítí (OOPN) můžeme uvést sítě spojené s nástrojem *PNtalk* vyvinutým na FIT VUT (viz `www.fit.vutbr.cz/~janousek/pntalk/`). PNtalk pracuje s:

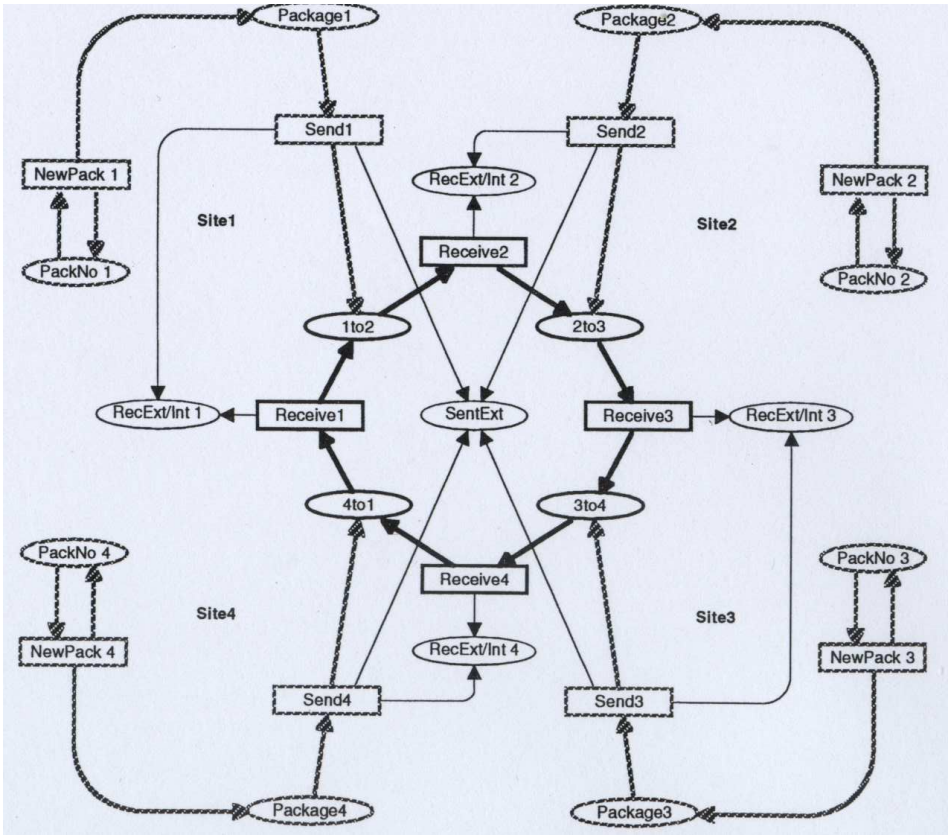
- *třídami* s *dědičností*,
- *objektovými sítěmi* – v každé třídě popisují strukturu stavu jejích instancí a jejich případné aktivní chování,



Obrázek 13.7: CPN model, jež vznikne substitucí podsítě z obrázku 13.6 do hlavní sítě z obrázku 13.6 [32]

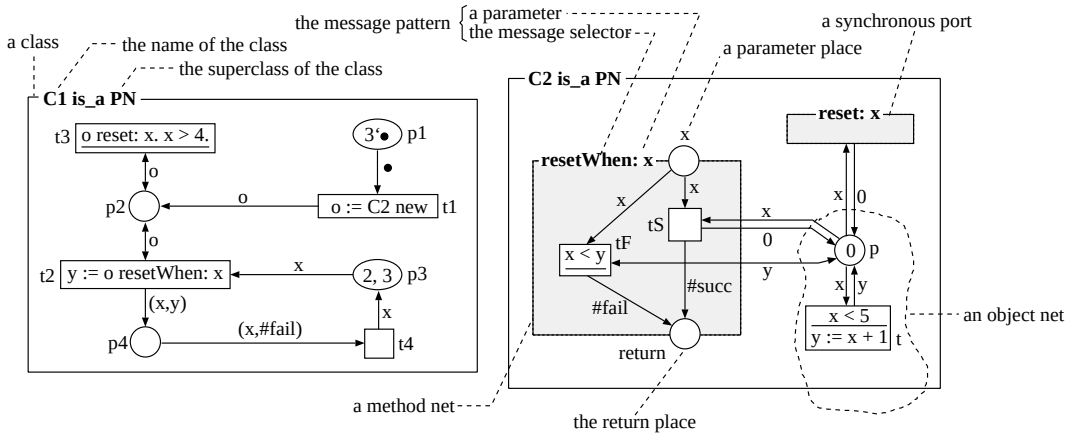
- *metodami* – mohou být vyvolány k asynchronní komunikaci s objekty,
- *synchronními porty* – zvláštní forma přechodů aktivovaná voláním a umožňující synchronní komunikaci s objekty,
- dynamicky vytvářenými *objekty* jako instancemi tříd a s běžícími *instancemi* metod,
- *pozdní vazbou* a *polymorfismem*.

V objektově orientovaných Petriho sítích spojených s nástrojem PNtalk jsou *výpočty umístěny do stráží a akcí přechodů* (na rozdíl od dříve popsaného konceptu CPN) a používá se *inskripční jazyk inspirovaný Smalltalkem* (na rozdíl od SML). Příklad dvou tříd PNtalku v modelu systému s čítači (jedná se o v zásadě umělý model manifestující na malé ploše většinu rysů systému PNtalk) je vidět na obrázku 13.9.



Obrázek 13.8: CPN model, jenž vznikne fúzí míst v síti z obrázku 13.7 [32]

Na závěr poznamenejme, že některé objektově orientované Petriho sítě, např. ty spojené s nástrojem PNtalk, nejsou určeny jen k modelování a analýze systémů. Jedním z jejich předpokládaných využití je návrh systémů založený na modelech (tzv. *model-based design*). Předpokládá se tedy, že vytvořená OOPN se použije přímo jako implementace navrhovaného systému. Za tím účelem je také PNtalk navrhován tak, že umožňuje lehkou kombinaci s rozsáhlými částmi kódu napsanými přímo ve Smalltalku (či jiných vhodných jazycích a formalismech – např. DEVS). Tímto způsobem je možné sítě v PNtalku mj. také propojit s různými hardwarovými zařízeními, jež by mohly být pomocí vytvořených sítí řízeny (jednou z předpokládaných aplikačních oblastí jsou vestavěné a řídicí systémy).



Obrázek 13.9: Dvě třídy objektově orientovaných Petriho sítí spojených s nástrojem PNtalk

Pojmy k zapamatování

- substituce míst, fůze míst
- substituce přechodů, invokace přechodů

Shrnutí

V této kapitole jsme se soustředili na základy možností hierarchického strukturování modelů založených na barvených Petriho sítích. To se sestává ze substituce přechodů, substituce míst, invokace přechodů a fůze míst. Při kombinaci fúze míst se substitucí přechodů, substitucí míst nebo s invokací přechodů je také možné zavést různé typy fúze.

Příklady k procvičení

1. Uveďte postup substituce míst a přechodů.
2. V čem se ze sémantického hlediska liší substituce přechodů od invokace přechodů?
3. Vyjmenujte typy fůze míst.

Kapitola 14

Závěr

Petriho sítě patří k významným nástrojům pro modelování paralelních systémů a systémů s diskrétním časem. Tato publikace představuje svým čtenářů pouze úvod do rozsáhlé problematiky Petriho sítí, přičemž se soustřeďuje na

- C/E (Condition-Event) Petriho sítě, kde počet značek v jednom místě nabývá hodnoty 0 nebo 1
- P/T (Place-Transition) Petriho sítě, kde počet značek v jednom místě nabývá celé nezáporné hodnoty

Závěrem části věnované P/T Petriho sítím jsou zmíněna některá další rozšíření Petriho sítí (časované, stochastické, apod.), ty však nejsou předmětem této publikace, proto nejsou studována důkladněji. Stejně jako Barvené Petriho sítě a jejich hiearchické strukturování, ze kterých je zde uveden jen základní úvod.

Primárním účelem této publikace je podpořit výuku teorie a aplikace Petriho sítí. Petriho sítě a jejich vlastnosti jsou zde definovány formálně, tj. pomocí algebraických struktur, tvrzení, algoritmů, apod. Tento formální popis je doplněn rovněž příklady či ilustracemi, které zvyšuji srozumitelnost textu. Některé z doprovodných příkladů modelují v praxi často používané konstrukce (např. vzájemné vyloučení, model čtenářů a písařů) a tak slouží čtenářům jako vodítko pro aplikaci modelů vytvořených pomocí Petriho sítí v praxi.

Literatura

[1] Manna, Z. *Matematická teorie programů.* Překlad z angličtiny. SNTL, Praha 1981.

[2] Hopcroft, J.E., Ullman J.D. *Formal Languages and Their Relation to Automata.* Addison-Wesley Publishing Company, Inc. 1969.

[3] Brookshear, J.G. *Theory of Computation.* The Benjamin/Cummings Publishing Company, Inc. Redwood City, California, 1989.

[4] Cormen, T., Leiserson, Ch.E., Rivest, R.L.*Introduction to Algorithms.* The MIT Press, McGraw-Hill Book Company, 1990.

[5] Češka, M., Rábová, Z. *Gramatiky a jazyky.* Nakladatelství VUT Brno, skriptum, 1992.

[6] Češka, M., Motyčková, L., Hruška, T. *Vyčíslitelnost a složitost.* Nakladatelství VUT Brno, 217 stran, 1992.

[7] Reisig, W. *A Primer in Petri Net Design.* Springer-Verlag, 1992.

[8] Peterson, J. *Petri Net Theory and the Modeling of Systems.* Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[9] Reisig, W. *Petri Nets: An Introduction.* Springer-Verlag, Berlin, Heidelberk, New York, Tokyo, 1985.

[10] Mayer, E.W. *The Complexity of the Final Containment Problem for Petri Nets.* Technical Report 181, Cambridge, Mass MIT Lab for Computer Science, 1977.

[11] Ginsburg, S. *The Mathematical Theory of Context Free Languages.* McGraw-Hill, New York, 1966.

[12] Peterson, J. *Modeling of Parallel Systems.* Ph.D. Disertation Thesis, Stanford University, 1973.

[13] Češka, M., Masár, L. *Analýza Petriho sítí.* Technická zpráva, Katedra počítačů FE VUT, Brno, 1985.

[14] Murata, T., Koh, J. *Reduction and Expansion of Live and Safe Marked Graphs.* IEEE Transaction on Circuits and Systems, CAS-27, No 1, 1980.

[15] Shepardson, J., Sturgis, H. *Computability of Recursive Functions.* Journal of ACM, Vol.10, No 2, April 1963.

[16] Merlin, P. *A Study on the Recoverability of Computer System.* Thesis, Department of Comuter Science, University of California, Irvine, 1974.

[17] Ramchandani, C. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.* MIT, Project MAC, Technical Report 120, 1974.

[18] Molloy, M. *Performance Analysis Using Stochastic Petri Nets.* IEEE Transactions on COmputer, Vol C-3, No 9, September 1982.

[19] Jensen, K., Rozenberg, G. (Eds.) *High-Level Petri Nets.* Springer Verlag, Berlin, Heidelberg, New York, Tokyo, 1991.

[20] Chmel, M. *Implementace metod analýzy Petriho sítí.* Diplomová práce, Katedra IVT FE VUT Brno, 1991.

[21] Skácel, M. *Implementace prostředků pro práci s Petriho sítěmi pod MS Windows.* Diplomová práce, Katedra IVT FE VUT Brno, 1992.

[22] Karásek, P. *Podsystém pro práci se stochastickými Petriho sítěmi.* Diplomová práce, Katedra IVT FE VUT Brno, 1992.

[23] Martinez, J., Silva, M. *A Simple and Fast Algorithm to Obtain All Invariants of a Generalised Petri Net.* In: Girault, C., Reisig, W. (Eds.) Application and Theory of Petri Nets. Informatic Fachbereite Nr. 52, pp 301–310, Springer, 1982.

[24] Pagnoni, A. *Stochastic Petri Nets.* In: Brauer, W., Reisig, W., Rozenberg, G. (Eds.) Petri Nets: Central Models and Their Properties – Advances in Petri Nets 1986, Part I. pp 461–474, Springer Verlag, Berlin,Heidelberg, New York, London, Paris, Tokyo, 1987.

[25] Marsan, M.A., Chiola, G. *Modeling of Discrete Event Systems with Stochastic Petri Nets.* Proceedings of ISCAS, pp. 739–742, 1985.

[26] Češka, M., Janoušek, V. *On the Implementation of Hierarchical Petri Nets.* MOSIS'93, Olomouc, 1993.

[27] Pinci, V.O, Shapiro,R.M. *An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets.* In: Royenberg, G. (Ed.) Advances in Petri Nets 1991, Lecture Notes in Computer Science 524, Springer Verlag, 1991.

[28] Jensen, K. *Computer Tools for Construction, Modification, and Analysis of Petri Nets.* In: Brauer, W., Reisig, W., Rozenberg, G. (Eds.) Petri Nets: Applications and Relationship to Other Models of Concurrency – Advances in Petri Nets 1986, Part II Proceedings of Advances Course, Bad Honef, Sept 1986, Lecture Notes in Computer Science 255, Springer Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1987.

[29] Rábová, Z. aj. *Modelování a simulace.* Skriptum, 216 stran, Nakladatelství VUT Brno, 1992.

[30] Češka, M., Janoušek, V., Skácel, M. *Petri Net Support for Information Engineering.* International Conference CATE'93, pp 353–356, Brno, 1993.

[31] Češka, M., Skácel, M. *Petri Net Tool PESIM.* Fifth International Workshop on Petri Nets and Performance Model, pp.1–10, Toulouse, France, 1993.

[32] Jensen, K.: *Coloured Petri Nets.* Monographs in Theoretical Computer Science, Springer-Verlag, 1992-1997. Volumes: Basic Concepts, Analysis Methods, and Industrial Case Studies.

[33] Huber, P., Jensen, K., Shapiro, R.M.: *Hierarchies in Coloured Petri Nets.* In: Advances in Petri Nets 1990, Lecture Notes in Computer Science 483, Springer-Verlag, 1991.

[34] Best, E., Fleischhack, H., Fraczak, W., Hopkins, R.P., Klaudel, H., Pelz, E.: *A Class of Composable High Level Petri Nets.* In: Proceedings of the 16th International Conference on Application and Theory of Petri Nets – ICATNP'95, Lecture Notes in Computer Science 935, Springer-Verlag, 1995.

[35] Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S. *Stochastic Well-Formed Coloured Nets for Symmetric Modelling Applications.* In: IEEE Transactions on Computers, 42(11):1343-1360, November 1993.

[36] Lakos, C.A., Keen, C.D. *LOOPN++: A New Language for Object-Oriented Petri Nets*. Technical Report R94-4, Department of Computer Science, University of Tasmania, Hobart, Tasmania 7001, April 1994.

[37] Sibertin-Blanc, C. Cooperative Nets. In: Proceedings of the 15th International Conference on Application and Theory of Petri Nets – ICATPN'94, Lecture Notes in Computer Science 815, Springer-Verlag, 1994.

[38] Janoušek, V. *Modelování objektů Petriho sítěmi*. PhD thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering and Computer Science, Technical University of Brno, Czech Republic, 1998.