1 Úvod

Tento dokument pojednává o funkčnosti, tvorbě a základních informacích o tvorbě simulátoru modelů P/T[1] Petriho sítí ve formě knihovny k programovacímu jazyku C++. Za pomocí statické a nebo dynamické knihovny je možné vytvořit model Petriho sítě a následně jej nasimulovat. Jedná se stochastické černobílé P/T Petriho sítě s rozšířenými vlastnostmi o pravděpodobnosti, prioritu a časování přechodů, které jim dávají výpočetní sílu Turingova stroje, tedy jsou schopny počítat všechny vyčíslitelné funkce.

1.1 kapitola

Na projektu pracoval dvoučlenný tým složený ze studentů bakalářského studia, úkoly byly rozděleny následovně:

- Dušan Kovačič (vedoucí týmu) - jádro simulace, dokumentace.
- Jaroslav Sendler - kalendář, generátory pseudonáhodných čísel, statistika a výpisy, dokumentace.

Pro účely řešení tohoto projektu nebyl využit žádný pravidelný poradce, tudíž nebylo použito žádných rad z třetích stran až na jednu konzultaci s hodnotícím, panem Hrubým. Odborná a pomocná literatura:

1. Rábová Z. a kol: Modelování a simulace, VUT Brno, 1992, ISBN 80-214-0480-9
2. Soubor materiálů prezentovaných na přednáškách

1.2 kapitola

V jakém prostředí a za jakých podmínek probíhalo experimentální ověřování validity modelu ? pokud čtenář/zadavatel vaší zprávy neuvěří ve validitu vašeho modelu, obvykle vaši práci odmítne už v tomto okamžiku.

2 Rozbor tématu a použitých metod/technologií

Petriho sít je matematická reprezentace diskrétních distribuovaných systémů. Vznikla za účelem modelovat řídící systémy.

Definice 2.1. Základní zápis Petriho sítě je možný v grafické formě popisující stavy a přechody mezi stavy, umožňuje vyjádření paralelismu a nedeterminismu.....viz prevzato ze opeora IMS str 31

Definici v podobě matematického zápisu viz. Definice P/T Petriho sítě na straně 126 ve slajdech.

Definice 2.2. Petriho síť je šestice $\Sigma = (P, T, F, W, C, M_0)$

kde:

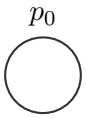
- P je množina míst
- T je množina přechodů, $P \cap T = 0$
- $F \subseteq (P \times T) \cup (T \times P)$ incidenční relace
- $W : F \to \{1,2, ...\}$ váhová funkce
- Kapacity míst. $C : P \to N$
- M_0 počáteční značení, $M_0 : P \to N$
- (M se nazývá značení Petriho sítě)

Petriho síť je tvořena pomocí tří následujících objektů:

[1] Place/Transitions

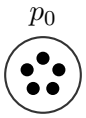
Místa

Místa (Places), modelují parciální stavy systému. Jejich vizuální vyjádření v grafu Petriho sítí je ukázáno na obrázku 1. Vlastnosti míst jsou definovány pomocí značek (tokens), které vyjadřují okamžitý stav systému.



Obrázek 1: Vyjádření místa v grafu

Tento stav je zobrazen na obrázku číslo 2. Je-li značka v místě přítomna, modeluje to skutečnost, že dané



Obrázek 2: Vyjádření místa s 5 značkami v grafu

stanovisko, uplatňované při posuzování, stavu je momentálně aktuální, nebo-li podmínka je splněna. Každé místo má nadefinovaný počáteční stav (počet značek na začátku) a kapacitu (maximální počet značek na místě). Místo bez ohodnocení je chápáno jako místo s neomezenou kapacitou.

Přechody

Přechod (Transitions), definují vzory možných událostí. Mezi vlastnosti jež definují přechod patří vstupní a výstupní místa, které jsou vyjádřeny orientovanými hranami spojující místa a přechody (viz.2).

Obrázek 5 zobrazuje grafický náčrt přechodů. Část A ukazuje zapsání vstupu do přechodu a část B zapsání výstupu.

Obrázek 3: A - Vstup do přechodu, B - výstup z přechodu

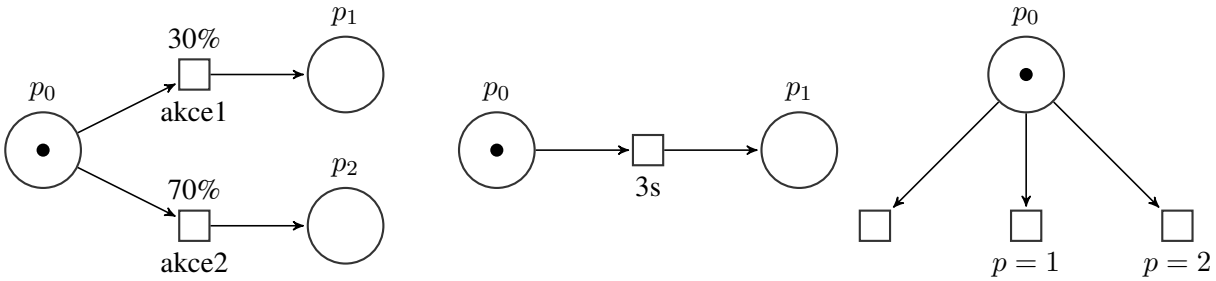
Přechod je proveditelný pouze pokud splňuje vstupní a následně výstupní podmínky. Jako vstupní podmínky se jeví obsah značek v místě vstupu. Při provádění přechodu se značky ze vstupního místa odeberou a přesunou se na výstupní místo, tímto se použijí výstupní podmínky. Zavedením rozšíření pro Petriho sítě zvětšujeme jejich popisnou sílu. Jsou to:

priority přechodů - je-li přechod s vyšší prioritou proveditelný, tak se provede dřív než kterýkoliv proveditelný přechod s nižší prioritou. Pokud přechodu neurčíme velikost priority jako defaultně je brána hodnota *0*, tato hodnota je i rezervována pro přechody časované. Díky prioritě je možno řešit deterministické konflikty.

pravděpodobnosti přechodů - je udávána v procentech. Celkový součet pravděpodobnosti všech přechodů ve stejné úrovni musí být *1*, neboli *100%*.

časování přechodů - do Petriho sítí přináší možnost pracovat s časem. Doposud byly všechny změny v síti provedeny okamžitě. Nyní změny mohou trvat určitou dobu.

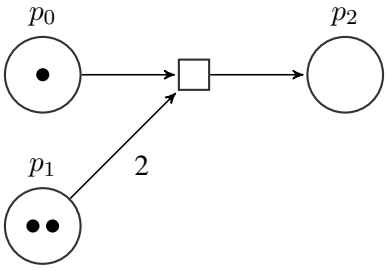
Tato rozšíření se navzájem nesmí kombinovat. Obrázek 4 ukazuje použití P/T s rozšířeními.



Obrázek 4: A - Vstup do přechodu, B - výstup z přechodu

Hrany

Hrany (Arcs), spojují místa s přechody nebo přechody s místy. Vlastnost, která udává počet značek, které se současně po hraně přesunou se označuje *kapacita*. Defaultně se chápe s hodnotou *1*.



Obrázek 5: A - Vstup do přechodu, B - výstup z přechodu

2.1 kapitola

Simulátor byl navržený tak, aby uspokojoval nejzákladnější parametry využití a byl jednoduchý a srozumitelný pro uživatele. Základní cíle byly tedy stanoveny na jednoduchost a intuitivnost. S trochou představivosti je absolutně předvídatelné co každá zadaná funkce dělá a jaké je jej použití. Na výsledný simulátor by bylo poměrně jednoduché navázat grafické uživatelské rozhraní. Dalším významným kriteriem při tvorbě, byla nutnost eliminovat co nejvíce chyb při samotném zápise zkoumaného modelu. Při prvním pohledu na složitější model, může být jeho význam nejasný, ale po podrobnějším prostudování je koncepčně totožný s grafickým zápisem. Samotný nápad o zápis modelu a jeho ovládání je výhradně dílem autora, inspirovaný podobnou knihovnou SIMLIB.

2.2 kapitola

PetriSim 1.0.0 byl vytvořený jen z poznatků a nápadů autorů, bez ovlivnění studováním funkčnosti jiných podobných nástrojů pro simulaci. Použité algoritmy ne jen, že umožňují zápis a simulaci Petriho sítě s rozšířeními definovanými v úvodě, ale i sémantickou kontrolu správnosti výsledného modelu a schopnost adekvátní simulace. Bez této kontroly by byl například možný nekonečný cyklus v nulovém čase, a nebo průběh nekonečně

trvající simulace. V případě takovýchto zjištění, je uživatel okamžitě upozorněný o vzniklé situaci a je mu navrhnuto řešení. Největší motivací pro nekopírování nápověd byla tužba vymyslet něco originálního. Autoři při začátku tvorby již měli zkušenosti s danou problematikou.

3 Koncepce modelu

3.1 kapitola

3.2 kapitola

- obrázek/náčrt/schéma/mapa (možno čitelně rukou)
- matematické rovnice - u některých témat (např. se spojitými prvky, optimalizace, ...) naprosto nezbytné. Dobré je chápat, že veličiny (fyzikální, technické, ekonomické) mají jednotky, bez kterých údaj nedává smysl.
- stavový diagram (konečný automat) nebo Petriho síť - spíše na abstraktní úrovni. Petriho síť nemá zobrazovat výpočty a přílišné detaily. Pokud se pohodlně nevejde na obrazovku, je nepoužitelná. Možno rozdělit na bloky se zajímavými detaily a prezentovat odděleně abstraktní celek a podrobně specifikované bloky (hierarchický přístup).

4 Architektura simulačního modelu/simulátoru

Pro samotné použití knihovny je nutné si ve zdrojovém souboru jazyka C++ přidat hlavičkový soubor `PetriSim.h`, který plně zpřístupní danou knihovnu. Na výběr je statická a nebo dynamická knihovna, kterou je nutné přilinkovat při kompilaci výsledného programu.

4.1 Třídy

SCBase

Základní rozhraní pro třídy SCPlace a SCTransition

`string` **`GetName()`**	
parametry	žádné
funkčnost	vrátí aktuální jméno objektu
vrácená hodnota	jméno objektu
`void` **`SetName(`**`string name`**`)`**	
parametry	jméno - `string`
funkčnost	nastavení jména objektu
vrácená hodnota	žádná
`int` **`GetStatus()`**	
parametry	žádné
funkčnost	vrátí status objektu
vrácená hodnota	aktuální stav objektu, bližší informace v `StatusList.h`

SCPlace : SCBase

Je náhradou miesta v grafickej reprezentácii. Rodicia: SCBase Metody:

`int` **`SetArgCapacity(`**`unsigned int capacity`**`)`	
parametry	kapacita - `unsigned int`
funkčnost	nastavení místu maximální kapacitu
vrácená hodnota	aktuální stav místa
`int` **`SetArgStartVal(`**`unsigned int startVal`**`)`	
parametry	počáteční hodnota - `unsigned int`
funkčnost	nastavení počáteční hodnotu značek
vrácená hodnota	aktuální stav místa

SCTransition:SCBase

Je náhradou prechodu v grafickej reprezentácii.

METODY

`int` **`SetArgPrio(`**`unsigned int prio`**`)`	
parametry	priorita - `unsigned int`
funkčnost	nastavení přechodu zadanou prioritu
vrácená hodnota	aktuální stav přechodu
`int` **`SetArgTime(`**`double time, int type = TIME_ABS`**`)`	
parametry	čas - `double`
	typ času - `int`
	`TIME_NORM` - základní čas, absolutní čas
	`TIME_EXP` - exponenciální rozložení
funkčnost	nastaví čas a jeho typ
vrácená hodnota	aktuální stav přechodu
`int` **`SetArgTime(`**`double from, double to, int type = TIME_NORM`**`)`	
parametry	interval čas od - `double`
	interval čas do - `double`
	typ času - `int`
	`TIME_NORM` - základní čas, gaussovo rolzožení
funkčnost	nastaví čas a jeho typ
vrácená hodnota	aktuální stav přechodu
`int` **`SetArgProbability(`**`double probability`**`)`	
parametry	pravděpodobnost - `double`
funkčnost	nastaví pravděpodobnost dabému přechodu
vrácená hodnota	žádná

4.2 Funkce

4.3 kapitola

Minimálně je nutno ukázat mapování abstraktního (koncept.) modelu do simulačního (resp. simulátoru). Např. které třídy odpovídají kterým procesům/veličinám a podobně.

5 Podstata simulačních experimentů a jejich průběh

Nezaměňujte pojmy testování a experimentování (důvod pro bodovou ztrátu)!!! Zopakovat/shrnout co přesně chcete zjistit experimentováním a proč k tomu potřebujete model. Pokud experimentování nemá cíl, je celý projekt špatně. Je celkem přípustné u experimentu odhalit chybu v modelu, kterou na základě experimentu opravíte.

`void` **`SetSimulationLimit(`**`double time`**`)`	
parametry	maximální délka simulace - `double`
funkčnost	nastaví maximální délku běhu simulace
vrácená hodnota	žádná
`int` **`Run()`**	
parametry	žádné
funkčnost	provede celou simulaci, zkontroluje validitu modelu a správný běh simulace
vrácená hodnota	v případě jak nastane chyba s modelem nebo s jeho úspěšným dokončením, upozorní u v případě úspěšného provedení simulace vrátí `0`

Pokud se v některém experimentu nechová model podle očekávání, je nutné tento experiment důkladně prověřit a chování modelu zdůvodnit (je to součást simulačnické profese). Pokud model pro některé vstupy nemá důvěryhodné výsledky, je nutné to zdokumentovat. Pochopitelně model musí mít důvěryhodné výsledky pro většinu myslitelných vstupů.

5.1 kapitola

Naznačit postup experimentování ? jakým způsobem hodláte prostřednictvím experimentů dojít ke svému cíli (v některých situacích je přípustné "to zkoušet tak dlouho až to vyjde", ale i ty musí mít nějaký organizovaný postup).

5.2 kapitola

Dokumentace jednotlivých experimentů - souhrn vstupních podmínek a podmínek běhu simulace, komentovaný výpis výsledků, závěr experimentu a plán pro další experiment (např. v experimentu 341. jsem nastavil vstup x na hodnotu X, která je typická pro ... a vstup y na Y, protože chci zjistit chování systému v prostředi ... Po skončení běhu simulace byly získány tyto výsledky ..., kde je nejzajímavější hodnota sledovaných veličin a,b,c které se chovaly podle předpokladu a veličin d,e,f které ne. Lze z toho usoudit, že v modelu není správně implementováno chování v podmínkách ... a proto v následujících experimentech budu vycházet z modifikovaného modelu verze ... Nebo výsledky ukazují, že systém v těchto podmínkách vykazuje značnou citlivost na parametr x ... a proto bude dobré v dalších experimentech přesně prověřit chování systému na parametr x v intervalu hodnot ... až ...)

5.3 kapitola

Závěry experimentů ? bylo provedeno N experimentů v těchto situacích ... V průběhu experimentování byla odstraněna ... chyba v modelu. Z experimentů lze odvodit chování systémů s dostatečnou věrohodností a experimentální prověřování těchto ... situací již napřinese další výsledky, neboť ...

6 Shrnutí simulačních experimentů a závěr

Nezaměňujte pojmy testování a experimentování (důvod pro bodovou ztrátu)!!! Zopakovat/shrnout co přesně chcete zjistit experimentováním a proč k tomu potřebujete model. Pokud experimentování nemá cíl, je celý projekt špatně. Je celkem přípustné u experimentu odhalit chybu v modelu, kterou na základě experimentu opravíte. Pokud se v některém experimentu nechová model podle očekávání, je nutné tento experiment důkladně prověřit a chování modelu zdůvodnit (je to součást simulačnické profese). Pokud model pro některé vstupy nemá důvěryhodné výsledky, je nutné to zdokumentovat. Pochopitelně model musí mít důvěryhodné výsledky pro většinu myslitelných vstupů.

- do závěru se nehodí psát poznámky osobního charakteru (např. práce na projektu mě bavila/nebavila, ...). Technická zpráva není osobní příběh autora.

- absolutně nikoho nezajímá, kolik úsilí jste projektu věnovali, důležitá je pouze kvalita zpracování simulátoru/modelu a obsažnost simulační studie (rozhodně ne např.: projekt jsem dělal ... hodin, což je víc než zadání předpokládalo. Program má ... řádků kódu). Pokud zdůrazňujete, že jste práci dělali významně déle než se čekalo, pak tím pouze demonstrujete vlastní neschopnost (to platí zejména v profesním životě).
- o závěru se velmi nehodí psát "auto-zhodnocení"kvality práce, to je výhradně na recenzentovi/hodnotiteli (např. v projektu jsem zcela splnil zadání a domnívám se, že můj model je bezchybný a výsledky taktéž). Statisticky častý je pravý opak autorova auto-zhodnocení. Pokud přesto chcete vyzdvihnout kvalitu svého díla (což je dobře), tak vaše výroky musí být naprosto nepopiratelně zdůvodněny a prokázány (např. pomocí jiného referenčního přístupu, matematického důkazu, analýzy, ...).