

Modelování hodnotového toku MHTOK, 3. Pravděpodobnost a algoritmy

David Kruml

30. 10. 2024

Nahodilost a pravděpodobnost

Cvičení: Zkuste nematematikovi definovat náhodný jev a pravděpodobnost.

Přístupy k nahodilosti

- ▶ Stoický přístup: svět je deterministický, náhoda je jen nedostatek informací.
- ▶ Odfiltrování části informací může být záměrné.
- ▶ Filozofie často řeší jen nahodilost důsledků, nejistota se ale může týkat i příčin.
- ▶ Pravděpodobnost vyjadřuje míru očekávání, že se něco stane. (To se ale určitě stane, nebo určitě nestane — klasická logika.)
- ▶ L. Zadeh, nespokojenost s pojetím náhody a rigidností teorie pravděpodobnosti — *fuzzy logika*, víc „míra platnosti“ než „míra jistoty“.

Přístupy k nahodilosti

- ▶ Klasická pravděpodobnost: poměr počtu příznivých instancí jevu k celkovému počtu. Musí se předpokládat konečnost, stejná váha, atp.
- ▶ Pokročilá pravděpodobnost: geometrická, abstraktně zadaná mírou, podmíněná, atd.
- ▶ Statistika — náhoda se týká sběru vzorku, řeší se oprávněnost závěrů.
- ▶ Frekventismus vs. bayesiánství.
- ▶ Číselná parametrizace jevového prostoru — *náhodná veličina*, výzkum typových pravděpodobnostních rozdělení.

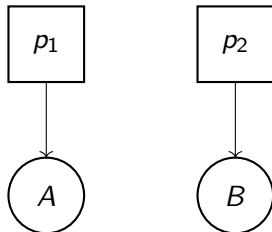
Cvičení: Co je průměrná mzda a co mediánová mzda?

Jak se určují?

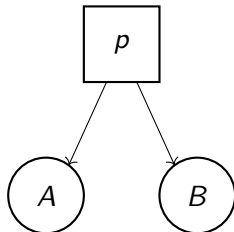
Co jsou zde náhodné jevy, veličiny, pravděpodobnost?

Procesní pojetí náhody — příklad s náhodným výstupem

deterministické procesy



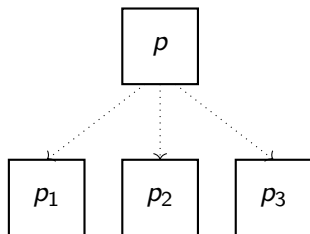
nedeterministický proces



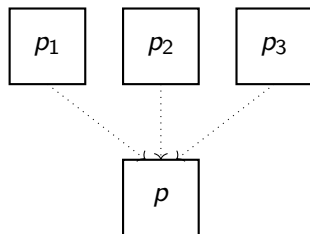
- ▶ p lze chápat jako agregaci (deterministických) procesů p_1, p_2 .
- ▶ Nahodilost výstupu = vznikla ztotožněním procesů při pokračujícím rozlišování zdrojů.

Spor o ideje

Platón



Aristotelés

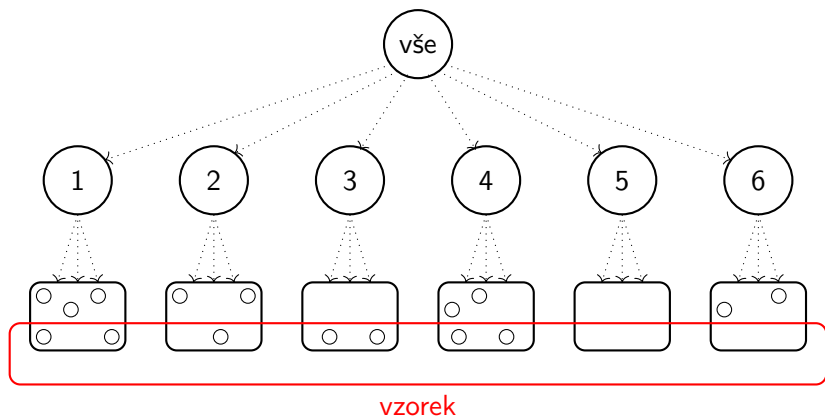


- ▶ Platón: Instance jsou (nedokonalé) obrazy idejí.
- ▶ Aristotelés: Ideje jsou zobecnění instancí.
- ▶ Kopírování a specifikace vs. zobecňování a slučování.

Agregace procesů a zdrojů

- ▶ Nahodilost je produktem *agregace* událostí.
- ▶ Agregace je dvojího typu:
 - ▶ podle blízkosti (navazující nebo souběžné procesy, zboží pro stejnou zakázku),
 - ▶ podle podobnosti.
- ▶ Rozdíl mezi typy lze zrelativizovat — i blízkost lze chápat jako druh podobnosti.
- ▶ Podobnost lze definovat různými hledisky, agregace pak může probíhat postupně a různými cestami.
- ▶ Místo rozpadové struktury je pak lepší uvažovat nestromová uspořádání.

Příklad s kostkou



Entropie I

- ▶ *Entropie* vyjadřuje míru nejistoty nebo neuspořádanosti.
- ▶ Příklad (1) $A \dots 80\%$, $B \dots 10\%$, $C \dots 10\%$ — hodně věřím A , varianty B , C jsou marginální.
(2) $A \dots 40\%$, $B \dots 30\%$, $C \dots 30\%$ — všechny tři varianty jsou docela podobně možné, nevím, čemu věřit.
Situace (2) má větší entropii než (1).
- ▶ Morseova abeceda — četná písmena mají krátký kód a vysílají se tak rychleji. U běžného textu dochází (oproti méně promyšlenému kódování) ke komprimační úspoře.
- ▶ Princip lze užít obecněji než na písmena a rozšířit na podřetězce atd., kódovat polyalfabeticky, atd. (jazyk „ptydepe“ V. a I. Havlových).
- ▶ V teorii informace entropie vyjadřuje nepřekonatelnou minimální velikost kódu (velikost v „dokonalé morseovce“).

Entropie II

- ▶ V příkladu (1) potřebujeme „méně než bit“ na kódování A . Naproti tomu pro B, C by bylo dobré volit 3-4 bity, protože $1/2^4 < 1/10 < 1/2^3$.
- ▶ Entropie se počítá vzorcem

$$E = - \sum_i P_i \log_z P_i,$$

kde základ logaritmu z vychází z typu aplikace (pravděpodobnost, informace, fyzika).

- ▶ Výpočet pro příklad (s logaritmem \log_2):

$$E_1 \approx 0.58, \quad E_2 \approx 1.25$$

Udává průměrnou spotřebu bitů na 1 znak.

Entropie, závěry

- ▶ Entropie je velmi univerzální charakteristika nahodilosti a současně velikosti informace.
- ▶ Nepotřebujeme jev transformovat na náhodnou veličinu.
- ▶ Může být vodítkem pro propojení datového skladu se statistickými nástroji.

Typická rozdělení ve výrobě

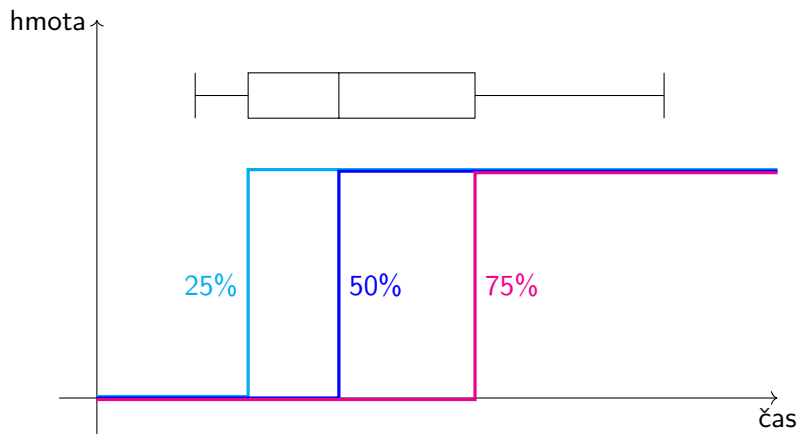
- ▶ Alternativní rozdělení — dobrý výrobek/zmetek.
- ▶ Normální rozdělení — „přírodní“ variabilita procesů (nebo materiálu)
- ▶ Exponenciální rozdělení — nehody a jiné katastrofy, událost se někdy stane, ale nevíme kdy. Jiné typ — méně závažné nehody jsou časté, závažnější se stávají řidčeji (černé labutě).
- ▶ Skládání efektů — obecnější rozdělení s pozitivní šikmostí, typicky unimodální (s jedním hrbem).
- ▶ Rozdělení vzniklá „prohozením os“, např. Poissonovo.

Cvičení: Připomeňte si pojmy hustota, distribuční funkce, kvantil.

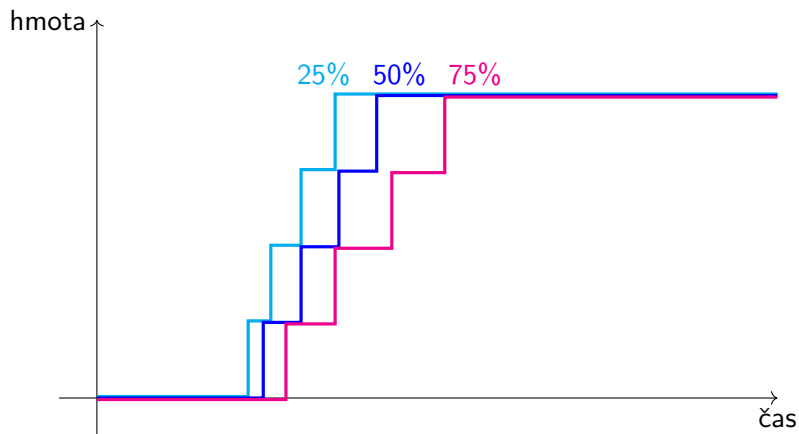
Připomenutí hustoty a distribuční funkce

- ▶ U spojitých veličin pravděpodobnost znázorňujeme *hustotou*. (Samotná bodová pravděpodobnost je nulová.)
- ▶ *Distribuční funkce* udává „celkový nárůst“ pravděpodobnosti, vidíme z ní *kvantily*.
- ▶ Jedno z druhého dostaneme derivací/integrací.
- ▶ Interpretace distribuční funkce pro procesní čas: je-li v bodě x hodnota $F(x) = p$, očekáváme, že z N běhů procesu bude pN rychlejších a $(1 - p)N$ pomalejších.
- ▶ Jinými slovy, proces se stihne do času x se spolehlivostí p .
- ▶ *Boxplot* — pětibodová charakteristika, všechny tři *kvantily* a dva okrajové kvantily (např. 5%, 95%). Případy za okraji se pokládají za nepodstatné.

Projev nahodilosti v časohmotovém diagramu



Typický průběh opakovaného náhodného procesu



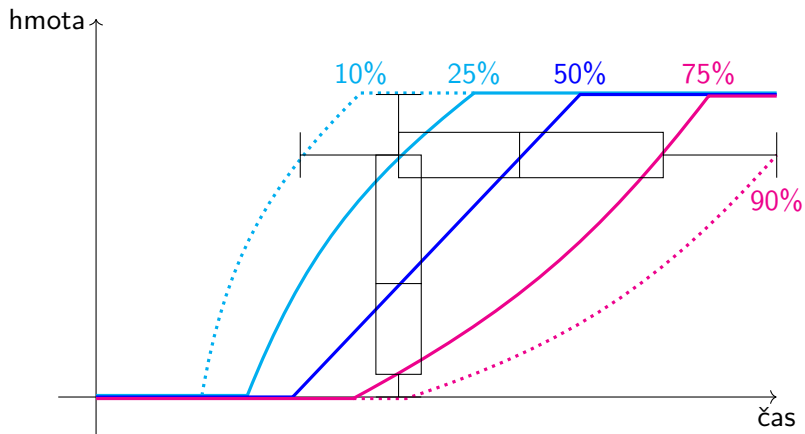
Komentáře k opakovanému náhodnému procesu

- ▶ Průběh lze modelovat metodou Monte Carlo — přičítání generovaných náhodných čísel pro dané rozdělení.
- ▶ Kvantily snadno vytěžíme ze vzorků (reálných dat nebo simulovaných): hodnoty seřadíme podle velikosti a podíváme se proporčně na pk -tou hodnotu, kde k je počet běhů.
- ▶ Kvantil nemusí odpovídat žádnému běhu — na některém výrobku může jeden běh předběhnout jiný. Pro každé n potřebujeme samostatné řazení.
- ▶ Kvantily se nemohou „překroutit“ — pro $p < q$ bude průběh p -tého kvantilu rychlejší/menší než q -tého (porovnávání jako 2-morfismus).
- ▶ Průběh kvantilových „schodišť“ lze aproximovat parabolickými křivkami — pro $p < 1/2$ konkávní, pro $p > 1/2$ konvexní. Brzy vysvětlíme.

Výměna os

- ▶ Naším cílem je výpočet zásob v zásobníku v pravděpodobnostním prostředí.
- ▶ Připomeňme, že zásobník je souběžně napájen kladnými (vstupními) a zápornými (výstupními) signály a ty je třeba efektivně sečíst.
- ▶ Sčítáme hmotu (zásoby), nikoli čas. Nahodilost časovou potřebujeme převést na nahodilost hmotovou. Místo „kdy bude hotovo n kusů?“ se ptáme „kolik vyrobíme do času t ?“
- ▶ Druhé z rozdělení je diskrétní (pokud je výroba diskrétní). Nicméně aproximace nahrazují schodiště spojitými křivkami a vedou tak ke spojitému popisu hmotové nahodilosti.
- ▶ Banální, ale důležité pozorování — rychlost výroby lze porovnávat dvěma ekvivalentními způsoby:
 - ▶ stejný objem vyrobíme v kratším čase,
 - ▶ za stejný čas vyrobíme více.
- ▶ V důsledku toho horizontální a vertikální náhodné veličiny „sdílí kvantily“.

Sdílení kvantilů



Modelářovo dilema

- ▶ V matematickém modelování často stojíme před dilematem, zda volit model
 - ▶ jednoduchý, ale nepřesný,
 - ▶ nebo přesný, ale složitý.
- ▶ Ve druhém případě složitost může způsobit nepřesnost jiného typu.
- ▶ Příklad: Reálná výrobní dat velmi pečlivě „rozškatulkujeme“, čímž dostaneme jemné členění na všechny myslitelné případy. Tím se ovšem zmenší vzorky a vzroste statistická chyba. Např. místo abych počítal všechny běhy procesu p , zaměřím se jen na běhy ve čtvrtek a v listopadu, protože dnes je čtvrtek a je listopad „a co kdyby to mělo nějaký vliv“.
- ▶ S ohledem na kvalitu dat je na místě uvážit i zjednodušení kalkulu pro práci s náhodnými veličinami.

Klasické sčítání náhodných veličin

- ▶ Necht' X, Y jsou nezávislé spojité náhodné veličiny s hustotami f, g . Pak je hustota h veličiny $X + Y$ dána konvolucí

$$h(z) = \int_{-\infty}^{\infty} f(x)g(z - x)dx.$$

- ▶ Příklad s diskrétními rozděleními — hod dvěma kostkami:

$$\begin{aligned} P(X + Y = 7) &= P(X = 1)P(Y = 6) + \\ &+ P(X = 2)P(Y = 5) + \\ &+ \dots + \\ &+ P(X = 6)P(Y = 1). \end{aligned}$$

- ▶ Spočítat integrál se často neumí, výsledkem součtu nemusí být „pěkné“ pravděpodobnostní rozdělení.
- ▶ Rezignujme na přesnost a počítejme součet přibližně a jednodušeji.

Momenty náhodné veličiny

- ▶ střední hodnota $E(X)$ — ve statistice a diskrétním rozdělení vážený průměr, u spojitých

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx.$$

- ▶ n -tý *centrální moment*:

$$m_n = \int_{-\infty}^{\infty} (x - E(X))^n f(x)dx.$$

- ▶ m_2 je *rozptyl*.
- ▶ m_3/m_2 je *šikmost*.
- ▶ Označme si střední hodnotu μ , rozptyl σ^2 a třetí centrální moment γ . Je-li je třeba rozlišit pro různé náhodné veličiny, napíšeme veličinu do indexu.

Aditivita momentů μ, σ^2, γ

- ▶ Všechny tři sledované momenty jsou *aditivní*, tj. pro nezávislé náhodné veličiny X, Y máme:

$$\mu_{X+Y} = \mu_X + \mu_Y,$$

$$\sigma_{X+Y}^2 = \sigma_X^2 + \sigma_Y^2,$$

$$\gamma_{X+Y} = \gamma_X + \gamma_Y.$$

- ▶ Tato vlastnost se bohužel pro momenty vyšších řádů pokazí, pokračovat lze s tzv. *kumulanty* (viz teorie momentových vytvořujících a charakteristických funkcí v teorii pravděpodobnosti).
- ▶ Pokud sčítáme n stejných nezávislých kopií téže veličiny X , označme výsledek $n * X$. Dostáváme:

$$\mu_{n*X} = n\mu_X,$$

$$\sigma_{n*X}^2 = n\sigma_X^2,$$

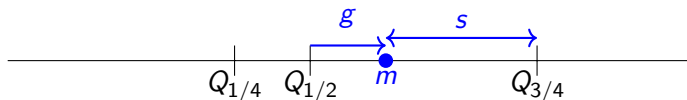
$$\gamma_{n*X} = n\gamma_X.$$

Vztahy mezi momenty a kvantily

- ▶ Pro normální (Gaussovo) rozdělení je známo, že kvantily jsou tvaru $\mu + k\sigma$ pro vhodné k ($k < 0$ pro $p < 1/2$ a $k > 0$ pro $p > 1/2$, μ splývá s mediánem).
- ▶ Pro opakovaný proces máme $\sigma_{n^*X}^2 = n\sigma_X^2$, a tedy $\sigma_{n^*X} = \sqrt{n}\sigma_X$. To je ono slibované vysvětlení parabolického vývoje kvantilů.
- ▶ Experimentálním pozorováním jsme také zjistili, že rozdíl střední hodnoty a mediánu se s n nemění. Rozdíl souvisí se šikmostí, což je v souladu s rovností

$$\frac{\gamma_{n^*X}}{\sigma_{n^*X}^2} = \frac{n\gamma_X}{n\sigma_X^2} = \frac{\gamma_X}{\sigma_X^2}.$$

Pseudomomenty



- ▶ Zavedeme si náhrady za momenty m, s, g podle vzorců:

$$m = \frac{Q_{1/4} + Q_{3/4}}{2}, \quad s = \frac{Q_{3/4} - Q_{1/4}}{2}, \quad g = \frac{Q_{1/4} + Q_{3/4}}{2} - Q_{1/2}.$$

- ▶ Tvařme se, že se chovají jako $\mu, \sigma, \gamma/\sigma^2$, tj. položme

$$m_{X+Y} = m_X + m_Y, \quad s_{X+Y} = \sqrt{s_X^2 + s_Y^2}, \quad g_{X+Y} = \frac{g_X s_X^2 + g_Y s_Y^2}{s_X^2 + s_Y^2}.$$

- ▶ Pro iterovaný proces zřejmě dostaneme

$$m_{n*X} = nm_X, \quad s_{n*X} = \sqrt{ns_X^2}, \quad g_{n*X} = g_X.$$

Komentáře k pseudomomentům I

- ▶ Převod kvantilů na pseudomomenty je lineární, inverzní převod je dán vzorcí:

$$Q_{1/4} = m - s, \quad Q_{3/4} = m + s, \quad Q_{1/2} = m + g.$$

- ▶ Odvodili jsme zjednodušený „fuzzy“ kalkulus pro sčítání náhodných veličin.
- ▶ Připomínáme předpoklady „rozumnosti“ uvažovaných veličin.
- ▶ Pro ně experimenty ukázaly překvapivě výborné výsledky (rozdíl v jednotkách procent vzhledem k σ).
- ▶ Nepřekvapivě horší jsou výsledky na extrémně odlišných rozděleních, např. alternativních, tvar U. Problémy jsou i na okrajích kampaní.
- ▶ U opakovaných procesů rozdělení velmi rychle tíhne k normálnímu (centrální limitní věta). Kdybychom zanedbali šikmost, asi by se nic hrozného nestalo a v praxi se to běžně dělá.

Komentáře k pseudomomentům II

- ▶ Metodu lze zobecnit a místo mediánu řešit obecný kvantil Q_p (třeba pro požadovanou spolehlivost), myšlenka je pořád podobná.
- ▶ Připomínáme, že s, g se kromě principiální nepřesnosti od původních momentů liší i násobkem.
- ▶ Pro odhad m (náhražka za střední hodnotu μ) se v popisné statistice někdy používá přesnější odhad

$$m = \frac{Q_{1/4} + 2Q_{1/2} + Q_{3/4}}{4}.$$

Zatím jsme nevyužili. Nepřesnosti se transformacemi tam a zpět částečně vykompenzují.

- ▶ Transformují se 3 kvantily na 3 pseudomomenty a zpět. Co je vhodným protějškem pro pětibodovou charakteristiku?

Strategie výpočtu stavu zásobníku v čase t

1. Signály modelujeme trojicí kvartilů.
2. Pro čas t překlopíme osy, tj. zjistíme hodnoty kvartilů na hmotové ose.
3. Kvartily transformujeme na pseudomomenty.
4. Sečteme pseudomomenty.
5. Pseudomomenty transformujeme na kvartily a odhadneme z nich kvantil = spolehlivost zásobníku pro čas t .

Lineární validace zásobníku I

- ▶ Jak bylo dříve naznačeno, signály *perfektních* opakovaných procesů lze shora i zdola aproximovat lineární funkcí.
- ▶ Aproximace může být i víceúrovňová (heijunka).
- ▶ Z takto aproximovaných signálů odhadujeme i signál na zásobníku.
- ▶ Přitom je třeba:
 - ▶ Rozdělit studovaný časový intervaly na úseky tak, aby na všech podintervalech byly všechny signály lineární, tj. hledáme společné zjemnění se všemi zlomovými body (událostmi).
 - ▶ Horní odhad počítat jako součet horních aproximací.
 - ▶ Dolní odhad počítat jako součet dolních aproximací.

Lineární validace zásobníku II

- ▶ Dále sledujeme zásoby ve zlomových bodech. Mohou nastat tyto situace:
 - ▶ Dolní odhad přeteče horní mez — zásobník jistě přetéká.
 - ▶ Horní odhad přeteče horní mez, ale dolní odhad ne — nevíme nic, musíme zpřesnit odhad další iterací.
 - ▶ Horní odhad nepřeteče horní mez — zásobník jistě nepřetéká.
- ▶ Duálně řešíme situaci na dolní mezi.
- ▶ Na sousedním bodu dělení může nastat:
 - ▶ stejná situace, pak se stejně chovají i všechny body uvnitř intervalu,
 - ▶ jiná situace, pak dochází k protnutí některého nebo obou odhadů s přímkou mezní zásoby.
- ▶ Případné průsečíky určí nové dělení a další (přesnější) iteraci provádíme jen na podintervalech, kde neznáme odpověď.

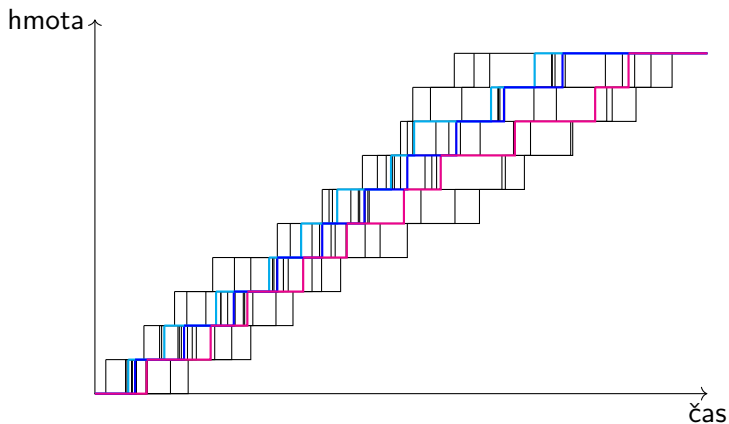
Lineární validace — komentáře

- ▶ Čím je výroba pravidelnější, tím víc aproximace šetří výpočetní čas.
- ▶ Iterace algoritmu (hlavní cyklus) by měla být navržena tak, aby přidávala (dříve přeskokované) události, a to jen ty nezbytné.
- ▶ Výsledkem algoritmu je rozčlenění studovaného intervalu na úseky, kde je zásobník funkční (v mezích) a kde selhává (mimo meze).
- ▶ V realitě selhání zásobníku může znamenat blokování nebo hladovění navazujících procesů.
- ▶ Průsečíky se počítají snadno, protože veškeré objekty jsou spojnice událostí, tedy úsečky.
- ▶ Přetečení a podtečení zásobníku je vhodné validovat samostatně, v souběžné validaci by bylo třeba diskutovat velké množství různých situací a vytvářet zbytečně jemné dělení intervalu.

Validace zásobníku s nahodilostí

- ▶ Připomeňme, že u náhodných procesů dochází k „rozmazání“ signálu. Není jasné, jak přesně výroba proběhne.
- ▶ Prostor všech možných průběhů popisujeme pomocí křivek kvantilů, což jsou hranice pomalejších a rychlejších průběhů pro danou pravděpodobnost.
- ▶ Kvantil stále „vypadá jako signál“, tj. (v diskrétní výrobě) jde o lomenou čáru definovanou událostmi.
- ▶ Aproximací kvantilu pro opakovaný proces je parabola (až na medián, kde je to přímka). V dalším se zaměříme i na úlohu, jak parabolu lineárně aproximovat.
- ▶ Rozšířený validační algoritmus tedy zohledňuje:
 - ▶ Místo jednoho (determinovaného) průběhu bereme v potaz několik význačných kvantilů — zatím nám stačí kvartily.
 - ▶ Pro každý kvantil uvažujeme horní i dolní odhad.
 - ▶ Lineární aproximace parabol vyžaduje další zjemnění členění intervalu na události. (Kombinaci tohoto členění s iteracemi algoritmu jsme zatím neřešili — pěkné téma pro BP.)

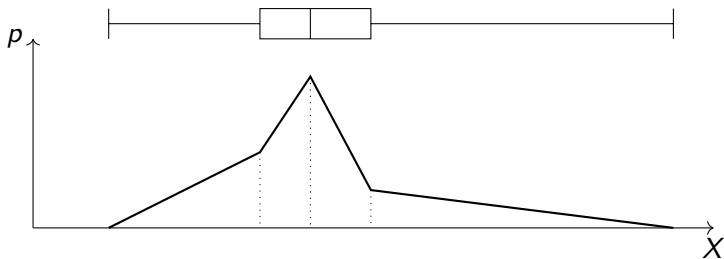
Simulace opakovaného náhodného procesu



15 běhů, vybíráme 4., 8. a 12. v pořadí.

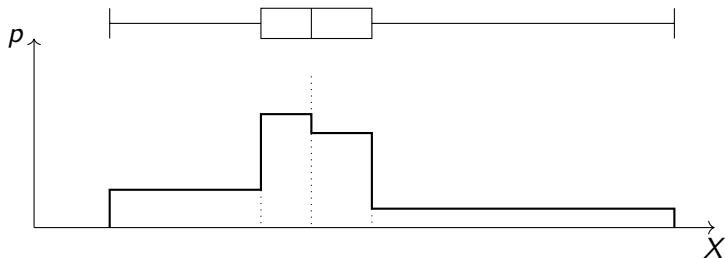
Co vyjde? I

- ▶ Pro každý bod dělení intervalu převádíme odhady kvantilů na *odhady pseudomomentů*, sečteme je pro všechny signály zásobníku a výsledek můžeme zpětně převést na kvantily.
- ▶ Pro kvartily pak bychom mohli obdržet odpověď typu: „Zásobník v čase t nepodtéká s pravděpodobností větší než 50%, ale menší než 75%.“
- ▶ „Přesnou“ spolehlivost si můžeme vymodelovat podle vzdálenosti meze od kvantilů, např. pomocí *lichoběžníkového rozdělení*:



Co vyjde? II

- ▶ Lichoběžníkové rozdělení dobře vypadá, ale trochu hůř se s ním počítá, protože distribuční funkce (integrál) je kvadratická, potřebujeme znát celou pětibodovou charakteristiku a levá a pravá polovina se nemusí potkat .
- ▶ Někdo by proto mohl dát přednost *obdélníkovému rozdělení*. To má lineární distribuční funkci a stačí nám znát jen okolní kvantily:



Změna dotazu

- ▶ Jinou možností je už od začátku počítat s jinou množinou kvantilů, např. $Q_{1/4}$, $Q_{3/4}$, Q_p , kde p je požadovaná spolehlivost.
- ▶ Místo doplňovací otázky „Jaká je spolehlivost v čase t ?“ řešíme otázku zjišťovací „Je spolehlivost v čase t aspoň p ?“
- ▶ Obě otázky mají své opodstatnění, zjišťovací forma má jednodušší výpočet.

Odmocninová rovnice I

- ▶ Z dřívějších úvah jsme dostali, že při výrobě n kusů v opakovaném procesu předpokládáme všechny kvantily procesního času ve tvaru:

$$t = nm + a\sqrt{ns} + bg,$$

kde a, b jsou vhodné konstanty pro danou pravděpodobnost a zvolený způsob modelování rozdělení.

- ▶ Výměna os odpovídá obrácení otázky „Kolik výrobků bude hotovo v čase t ?“, tj. řešíme rovnici v n pro parametr t .
- ▶ Po úpravách dostáváme:

$$\begin{aligned}as\sqrt{n} &= t - nm - bg, \\(as)^2 n &= m^2 n^2 + (t - bg)^2 - 2m(t - bg)n, \\m^2 n^2 - ((as)^2 + 2m(t - bg))n &+ (t - bg)^2 = 0.\end{aligned}$$

Odmocninová rovnice II

- ▶ Substitucí

$$A = \frac{a^2}{m^2}, \quad B = \frac{t - bg}{m}$$

rovnici zjednodušíme na tvar

$$n^2 - (A + 2B)n + B^2 = 0.$$

- ▶ Diskriminant je

$$D = (A + 2B)^2 - 4B^2 = A^2 + 4AB = A(A + 4B).$$

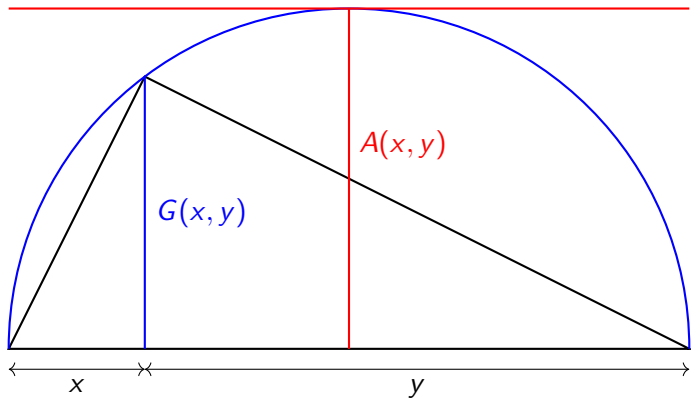
Odmocninová rovnice III

- ▶ Dostáváme kořeny

$$n_{1,2} = \frac{(A + 2B) \pm \sqrt{A(A + 4B)}}{2}.$$

- ▶ Zajímá nás jeden z kořenů, druhý nedává smysl. Výběr kořene určuje konstanta a , tj. zda řešíme konvexní/konkávní parabolickou funkci (tj. kvantil pro $p < 1/2$ resp. $p > 1/2$).
- ▶ Všimneme si, že $A + 2B$ je *aritmetický průměr* čísel A a $A + 4B$, zatímco $\sqrt{D} = \sqrt{A(A + 4B)}$ je jejich *geometrický průměr*.
- ▶ Připomeňme, že mezi průměry platí nerovnost $GP \leq AP$ a podle Eukleidovy věty o výšce je náš GP roven výšce pravouhlého trojúhelníku, v němž pata výšky dělí přeponu právě na úseky délek A a $A + 4B$.

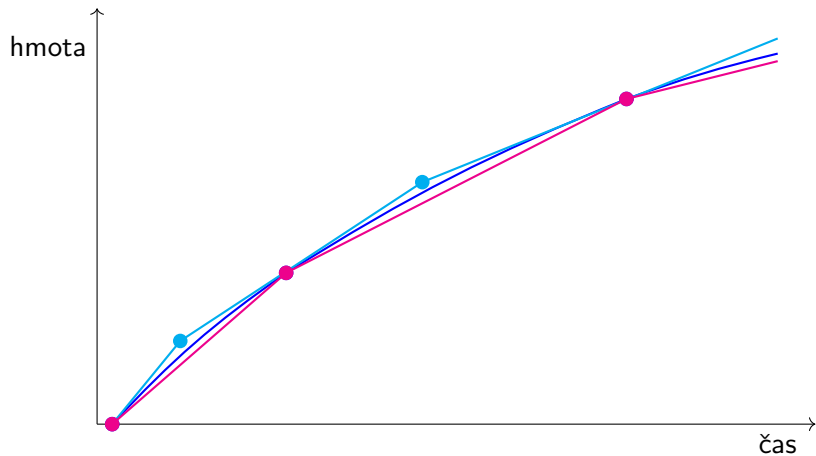
Eukleidova věta o výšce a AG nerovnost



Lineární aproximace paraboly

- ▶ Kvůli kombinaci požadavků na horní/dolní odhad a konvexnost/konkávnost potřebujeme umět parabolu aproximovat zvenku i zevnitř.
- ▶ Z mnoha možností dáme přednost řešení, kde
 - ▶ zpřesnění aproximace se provede zjemněním dělení — chceme jen přidat nové události a nezahazovat výpočty pro staré,
 - ▶ koeficienty aproximačních přímků budou racionální — tím se odbourá část problémů s počítačovým zaokrouhlováním u reálného typu.
- ▶ Vnější aproximace je náročnější — po nalezení dotykových bodů musíme ještě počítat tečny a jejich průsečíky.

Vnitřní a vnější aproximace



Racionální aproximace

- ▶ Druhý požadavek je konkrétněji položen takto: místo výpočtu odmocninou

$$n = \frac{(A + 2B) \pm \sqrt{A(A + 4B)}}{2}$$

hledáme přibližné řešení ve tvaru

$$n \approx k_1 A + k_2 B,$$

kde k_1, k_2 jsou racionální.

- ▶ K tomu je zřejmě dostatečné a nutné, abychom našli hustou podmnožinu bodů na Thaletově kružnici, v níž bude poměr výšky (GP) k přeponě (2AP) racionální.

Pythagorejské trojice

- ▶ *Pythagorejskou trojicí* rozumíme trojici přirozených čísel (a, b, c) splňujících Pythagorovu větu:

$$a^2 + b^2 = c^2.$$

- ▶ Pythagorejské trojice lze triviálně množit vynásobením přirozeným číslem.
- ▶ Nesoudělné trojice se nazývají *primitivní*, tj. nejsou výsledkem netriviálního násobení jiné PT. Je známo, že i primitivních trojic je nekonečně mnoho.
- ▶ Pokud prvočíslo p dělí dvě strany, musí dělit i stranu třetí. Nesoudělnost v PT tedy stačí testovat na libovolných dvou stranách.
- ▶ Eukleides dokázal, že pro $x, y \in \mathbb{N}, x > y$ je trojice

$$(x^2 - y^2, 2xy, x^2 + y^2)$$

pythagorejská a všechny PT vzniknou tímto způsobem.

Posloupnosti pythagorejských trojic

- ▶ V Eukleidově klasifikaci můžeme volit speciální x, y a tvořit posloupnosti PT, např.

- ▶ Pythagorova posloupnost, $x = y + 1$:

$$(3, 4, 5), \quad (5, 12, 13), \quad (7, 24, 25), \quad (9, 40, 41), \quad \dots$$

- ▶ Platónova posloupnost, $y = 1$:

$$(3, 4, 5), \quad (8, 6, 10), \quad (15, 8, 17), \quad (24, 10, 26), \quad \dots$$

- ▶ Pythagorovu posloupnost lze až na násobek zadat volbou $y = 1/2$.
- ▶ Zaměříme se na posloupnosti s konstantním y a ještě konkrétněji na posloupnosti s $y = 2^k$. Lze ukázat, že zvýšením $k - 1 \rightarrow k$ „půlíme“ předchozí iteraci $y = 2^{k-1}$, tj. pro stávající PT zdvojnásobíme index v posloupnosti a nové PT vkládáme na liché pozice.
- ▶ V každé posloupnosti se zmenšuje úhel při jednom z vrcholů (a u druhého zvětšuje). Jedná se tak o pohyb po oblouku Thaletovy kružnice (a potažmo po parabole v původní úloze).

Topologie pythagorejských trojic

- ▶ Lze klasifikovat dvojice (x, y) , které generují primitivní PT.
- ▶ Nás primitivita moc nezajímá, potřebujeme hlavně nějak generovat vzájemně nepodobné PT.
- ▶ Distribuce reprezentací PT na kružnici, v Gaussově celočíselné rovině, atp. nápadně připomínají konstrukci množiny racionálních čísel a její vložení do \mathbb{R} . Krácení zlomků odpovídá redukci PT na primitivní. V rámci teorie PT jsou známy různé zajímavé reprezentace a transformace.
- ▶ Pro nás je podstatné, že pravoúhlý vrchol skutečně vykresluje pro PT na Thaletově kružnici hustou podmnožinu, jak jsme potřebovali. Jinými slovy, každý pravoúhlý trojúhelník je v určitém smyslu limitou pythagorejských.
- ▶ Hustá je i množina získaná speciálními volbami $y = 2^k$, $k \in \mathbb{N}$. (To je analogie faktu, že zlomky tvaru $z/(2^k)$ tvoří hustou podmnožinu v \mathbb{R} .)

Komentáře k validaci I

- ▶ Kombinací odhadů kvantilů a lineárních aproximací paraboly bychom měli získat algoritmus pro validaci zásobníku v pravděpodobnostním prostředí. Strategie výpočtu založená na přidávání událostí v neobjasněných intervalech zůstává stejná. (Problém čeká na dokončení.)
- ▶ Ve srovnání se simulací Monte Carlo má lineární validace výhody:
 - ▶ Poskytuje stále stejné výsledky a neprojevuje se efekt statistické chyby generovaných dat.
 - ▶ Pro rozsáhlejší kampaně a/nebo větší rozptyl očekáváme rychlejší výpočet.

A nevýhody:

- ▶ Nevidíme skutečný projev kolizí na celkové zpomalení toku — nepočítáme LT .
 - ▶ Pro menší dávky/rozptyl může být rychlejší zkusit MC.
- ▶ Téma k zamyšlení/práci: kde je ta hranice?

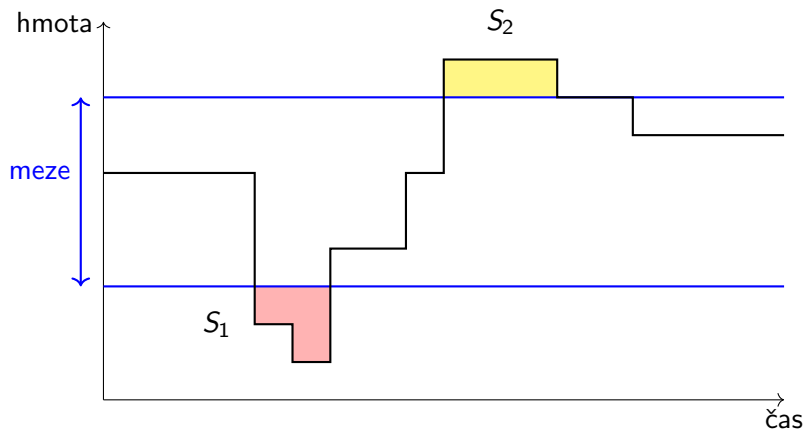
Komentáře k validaci II

- ▶ Připomeňme, že validace zásobníků lineárními aproximacemi se vyplácí pro výrobu se „střední pravidelností“. Vysoce pravidelná výroba (linky, nepřetržitý provoz) nezná kolize, kusová výroba nezná opakované procesy.
- ▶ V kusové výrobě se nahodilost často vyskytuje. Součet veličin na hmotové ose se popsanou metodou určitě nevyplatí (a navíc je nepřesný), rozumné je použít diskrétní konvoluci.
- ▶ Stejná poznámka se může týkat i situací, kde rozptyl a/nebo velikost kampaně je menší a hmotové rozdělení tak má malý *nosič*.
- ▶ V praxi se často řeší obrácená úloha „zjistit minimální kapacitu zásobníku / počáteční zásoby“. V našem pojetí odpovídá stanovení mezí tak, aby se simulovaný signál (s požadovanou spolehlivostí) mezi ně vešel.

Defekt

- ▶ Validací zásobníky jsme schopni měřit i míru selhání — velikost plochy v čase a hmotě. Opět se můžeme spokojit i s prostým odhadem aproximacemi.
- ▶ Přiřazením *váhy* příslušnému selhání (kolize) definujeme *defekt*.
- ▶ Defekt tedy číselně vyjadřuje míru problému, tj. jak moc nám kolize vadí.
- ▶ Defekty jsou sice nežádoucí, ale mohou být nevyhnutelné a stanou se předmětem úvah typu „něco za něco“.
- ▶ Je tedy vhodné rozmyslet si co nejširší váhové ohodnocení zásobníků.
- ▶ *Souhrnný defekt* je součet všech defektů. Je možné ho považovat za *účelovou funkci* pro optimalizaci plánu.

Ilustrace defektů



$$w_1 S_1 + w_2 S_2$$

Příklady defektu

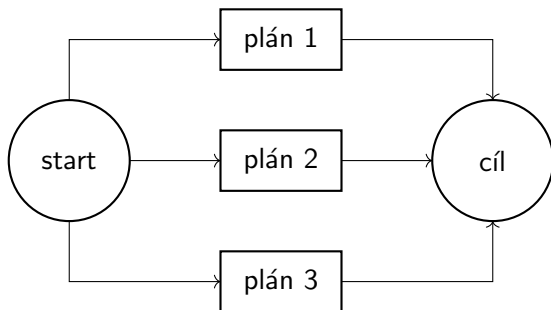
- ▶ „Základní“ příklad — podtečení nebo přetečení zásobníku s materiálem. Nelze vyrábět kvůli hladovění nebo blokování procesu.
- ▶ Nevyužitá kapacita zařízení — pokutujeme stav vyčkávání (idle), tj. vytvoříme pro stroj zásobník, v němž je stav 0 horní mezí a stav 1 je už vnímán jako „přetečení“.
- ▶ Kolize procesů na zařízení — stejný zásobník jako výše, ale kolizí z něj odebíráme stroj „dvakrát“. Ve výsledku je tak v zásobníku „-1 připravených strojů“. Stav pokutujeme jako nežádoucí podtečení dolní meze 0.
- ▶ Nesprávný režim stroje — pro proces můžeme vytvořit virtuální zásobník, kam si proces „odplivuje“, abychom nemuseli měnit způsob výpočtu defektu.
- ▶ Nedodržení termínu expedice — kolize posledního výrobního procesu a procesu expedice na zásobníku se zakázkou.
- ▶ Bonifikace rychlé výroby — stanovíme si nesplnitelný „vnitřní deadline“ a řešíme stejně jako předchozí.

Obecnější způsoby definice defektu

- ▶ V pravděpodobnostním prostředí můžeme defekt rozčlenit podle kvantilů a přirozeně tak navíc vážit pravděpodobnostní mírou, což vyjadřuje průměrný příspěvek jednoho náhodného běhu k defektu.
- ▶ Pokutování za defekt může být progresivní — tj. malé překročení vadí málo, větší překročení je průšvih. Princip výpočtu je podobný jako u předchozího.
- ▶ Při agregaci nemusí být „celková nelibost“ prostým součtem přispívajících defektů. Lze vytvořit pomocný zásobník pro sdružený stav a stanovit pro něj váhu. Příklad: Úloha o vlku, koze a zelí. Čtyři konstelace jsou přijatelné, čtyři nejsou. Defekt není možné vyjádřit prostým součtem „po zvířatech“.
- ▶ Pokud si představíme tok rozložený na jednotlivé výrobky, odpovídající zásobníky, pravděpodobnostní běhy, atd., zjistíme, že stále řešíme tentýž obecný problém.

Souhrnný defekt plánu

- ▶ Podle předchozího získáme souhrnný defekt jako součet všech dílčích defektů (vážené plochy), tj. předpokládaných ztrát.
- ▶ Pro všechny myslitelné kolize je třeba stavit váhu, což může být nesnadné a pracné.
- ▶ Za předpokladu stejného zisku za vyhotovení smluvené zakázky je tak souhrnný defekt rozdílovým kritériem pro výběr lepšího plánu.



Optimalizace I

- ▶ Cílem *optimalizace* je minimalizace účelové funkce při dodržení omezení.
- ▶ *Omezení* jsou pevná, jejich překročením bychom se dostali mimo prostor *přípustných řešení*.
- ▶ Neštěstí optimalizace: Globální optimum obvykle není „sjednocením“ lokálních optim. Nicméně v některých případech lze předpokládat určitou nezávislost komponent a úlohu rozdělit.
- ▶ Optimalizační metody obvykle postupují prostorem přípustných řešení a snižují účelovou funkci.
- ▶ Spoustu práce může ušetřit vhodná volba počátečního plánu.

Optimalizace II

- ▶ Úspěšnost metody lze vyjádřit počtem cyklů — čím méně, tím lépe.
- ▶ Užitečná je schopnost vědět „kam jít“ — volba pivota v simplexovém algoritmu lineárního programování, metoda největšího spádu, atd.
- ▶ Pokud množina přípustných řešení není rozumně parametrizovatelná, volba slibného směru nemusí být možná.
- ▶ Optimalizace pak probíhá metodou pokus/omyl, je zdlouhavá a nabízí se ukončení v suboptimálním řešení.

Simulované žíhání

- ▶ Podstatou *simulovaného žíhání* je „ochota“ provádět velké (a riskantní) změny spíš na začátku optimalizace. Postupně se tato ochota zmenšuje — chlazení.
- ▶ Změnou v plánu rozumíme např. přesun úkolu v čase (vodorovný posun v Ganttovi), výměnu úkonů ve frontě, velkou změnou může být obdoba pro složený proces (např. výměna celého středěčního rozvrhu za úterní).
- ▶ Ochotou ke změně rozumíme pravděpodobnost, algoritmus je náhodný.
- ▶ Pokud je nalezené řešení lepší než předchozí, určitě jej bereme. Pokud je horší, pokračujeme v průzkumu s aktuální pravděpodobností.
- ▶ Po čase se pravděpodobnost sníží natolik, že už se žádná změna téměř jistě nestane — zamrznutí.
- ▶ Velké skoky jsou užitečné jako možnost opustit oblast prostoru řešení směřující k „méně optimálnímu“ lokálnímu minimu.

Metrizace prostoru plánů

- ▶ Pro potřeby simulovaného žíhání je vhodné zavést *metriku* na prostoru plánů měřící jejich odlišnost.
- ▶ Nabízí se měřit vzdálenosti odpovídajících událostí v časoprostorohmotě, tedy velikost souhrnné deformace měnící jeden plán na druhý.
- ▶ Problém může být zajímavý v kategoriálním vyjádření — plány jsou funktoři, události jsou propojeny morfismy přirozené transformace.
- ▶ Není doděláno, prostor pro výzkum.
- ▶ U simulovaného žíhání mohou pomoci „zjevné korekční tahy“, např. posunutí procesů ve frontě, aby k sobě přiléhali podle procesních časů. (Při vyšší složitosti sítě ale mohou něco rozbít jinde.)

Shrnutí řešené problematiky

- ▶ Validace — hledání kolizí v toku, odpovědi typu ano/ne, případně pravděpodobnost.
- ▶ Evaluace — kolize jsou měřeny a je jim přisouzena váha, výsledek nazýváme defekt. Součet všech defektů dává hodnocení toku.
- ▶ Optimalizace — změnou plánu se snažíme snižovat celkový defekt a najít (sub)optimální řešení.

Jazyk toku

- ▶ Pokusíme se přepsat tok do kódu.
- ▶ Kód je současně datovým záznamem i jistým „programem“ — lze podle něj naplánovat a spustit výrobu, nebo aspoň její simulaci.
- ▶ Kód může být předmětem dalšího zpracování — rozvinutí, zobecnění, optimalizace, atd. Princip *funkcionálního programování*.
- ▶ Potřebujeme popsat procesy i zásobníky. Připomeňme, že tok lze konstruovat jako diagram $D : \mathcal{D} \rightarrow \mathcal{K}$, kde \mathcal{K} je kategorie zdrojů spojených procesy.
- ▶ Skládání lze dobře vystihnout *značkovacím jazykem*, pro rychlé vyhledávání jsou vhodné *relační databáze*.
- ▶ Pro vodorovné/svislé oddělovače položek použijeme csv notaci, tj. čárky/středníky.

Principy přepisu

- ▶ Programujeme „objektově“, tj. uděláme si základní kostru, pak rozepisujeme detaily.
- ▶ Rozmysleme si, kdy procesy probíhají nebo zdroje se spotřebovávají v daném pořadí a kdy je pořadí zaměnitelné.
- ▶ Každý proces „nahlíží“ na zdroje určitým typem a úrovní agregace. Vzniká u něj *složený zásobník* (vysvětlíme na příkladech).
- ▶ Opakované procesy vyjádříme cyklem (s počtem opakování příslušnému velikosti kampaně).

Označení pro kompozice

- ▶ $\{a, b, c\}, n * \{a\}$ — typ (multi)množina, složky jsou nezávislé, zaměnitelné, proveditelné paralelně, je asociativní i komutativní:

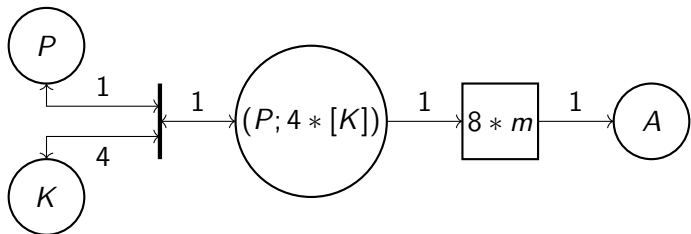
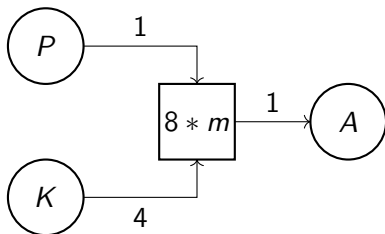
$$\{a, b\} = \{b, a\}, \quad \{\{a, b\}, c\} = \{a, \{b, c\}\}$$

- ▶ $(a, b, c), n * (a)$ — typ uspořádaná n -tice/seznam, pevné řazení složek, sériové provedení v daném pořadí, jen asociativní:

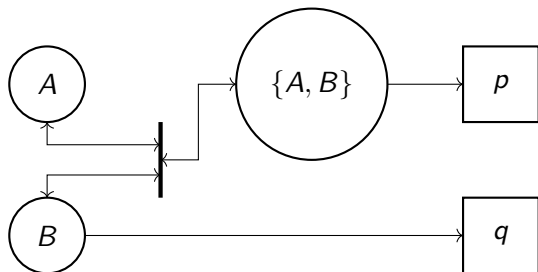
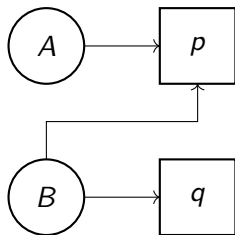
$$((a, b), c) = (a, (b, c))$$

- ▶ $[a, b, c], n * [a]$ — podobné jako kulatá závorka, ale není ani asociativní, složky jsou zpracovány nerozbalené, závorka chrání svůj obsah před „rozpuštěním“ do nadřazeného výrazu.

Příklad s virtuálním supermarketem I

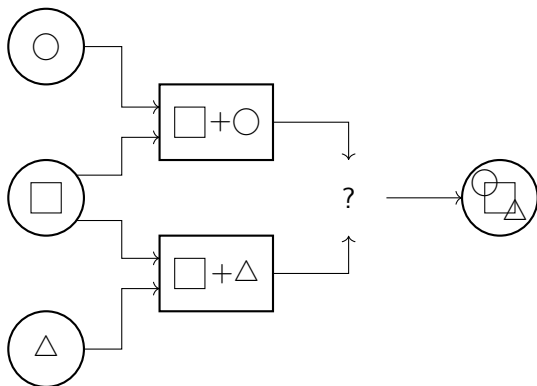


Příklad s virtuálním supermarketem II

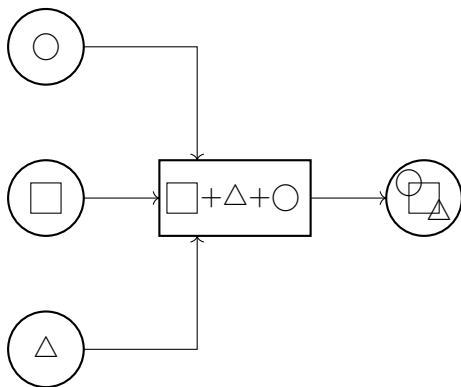


Příklad s nezávislými procesy (R. Koutenský)

- ▶ Uvažujeme dvojici procesů, kde na celek (např. auto \square) je třeba přimontovat dva doplňky (např. volant \circ a zrcátko \triangle).
- ▶ Procesy mohou být provedeny:
 1. zcela nezávisle (klidně i souběžně),
 2. v libovolném pořadí (ale jeden po druhém).

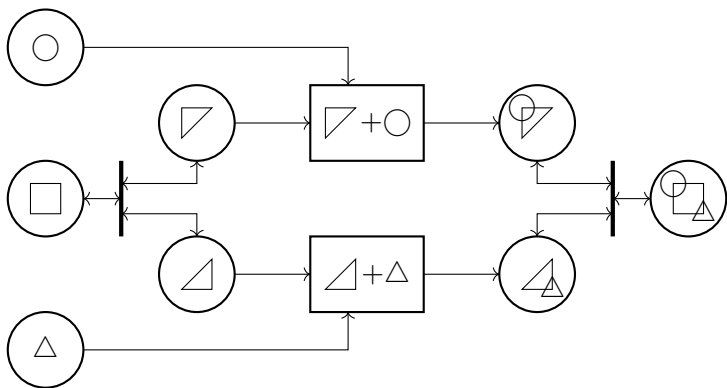


Naivní řešení 1. verze

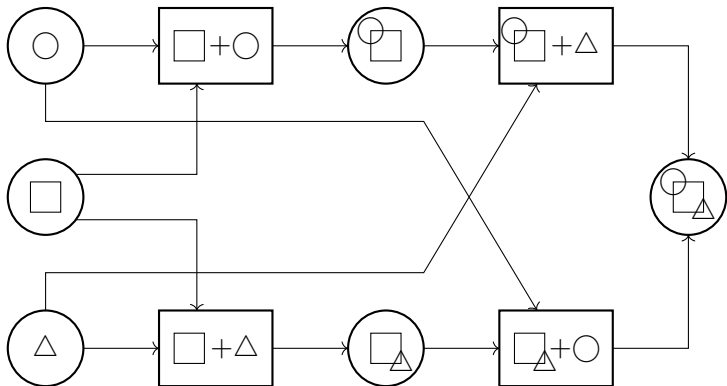


Chování složeného procesu nemáme pod kontrolou, nic nespočítáme.

Dvoušterbinové řešení 1. verze

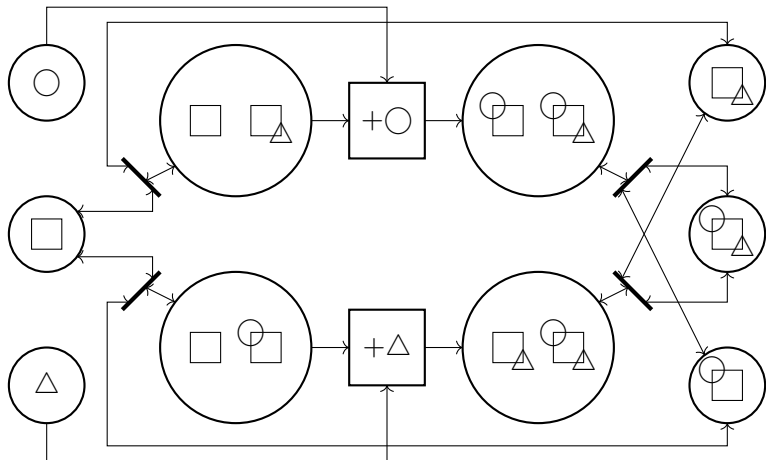


Dvoucestné řešení 2. verze



Nepodařilo se sloučit stejné procesy. Složitost schématu rychle roste s počtem procesů.

Schrödingerovsko–kočkovité řešení 2. verze



Databáze procesů

vstup	proces	výstup
$(A; 3 * [B])$	p	C
C	q	$(D; E)$
$(A; 3 * [B])$	(p, q)	$(D; E)$

- ▶ Jazyk zatím v náznaku, zřejmě bude ještě nutné rozlišit horizontální a vertikální skládání.
- ▶ „Přepážkový“ proces ve schématech je obousměrný a jen překládací (analogie s polními částicemi).
- ▶ Závorky odpovídají standardním databázovým operacím sjednocení a spojení.
- ▶ Vše směřuje k relačnímu popisu toku. Kategorie **Rel** je dagger kompaktní stejně jako konečné vektorové prostory.
- ▶ Propojení s jazykem kvantových protokolů vypadá velmi nadějně.