INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA STROJNÍ

PROGRAMOVÁNÍ ŘÍDÍCÍCH SYSTÉMŮ

Petriho sítě

Ing. Ivo Špička, Ph.D.

Ostrava 2013

ISBN 978-80-248-3041-4

Tento studijní materiál vznikl za finanční podpory Evropského sociálního fondu (ESF) a rozpočtu České republiky v rámci řešení projektu: CZ.1.07/2.2.00/15.0463, MODERNIZACE VÝUKOVÝCH MATERIÁLŮ A DIDAKTICKÝCH METOD

OBSAH

12 PETRIHO SÍTĚ

Obsah kapitoly:

definujte evoluční pravidla

popište Petriho sítě

CÍL:

Po prostudování tohoto odstavce budete umět:

- definovat evoluční pravidla
- popsat Petriho sítě

12.1 PETRIHO SÍTĚ

Princip abstrakce, jenž je snad vůbec nejdůležitějším principem modelováni, vedl v sedmdesátých letech k vytvoření řady matematických modelů paralelních systémů. Účelem těchto modelů je popis abstrakce paralelního systému z hlediska koordinace a synchronizace procesů. Tato abstrakce se označuje jako řídicí struktura paralelního systému. K nejznámějším matematickým modelům paralelních systémů patři Petriho síť.
V této kapitole se seznámíme s řídicí strukturou paralelního systému ve tvaru p-sítě, jež představuje unifikovaný model řídicí struktury téměř identický s Petriho sítí. Tohoto modelu budeme využívat pro popis synchronizačních úloh, kde p-sít představuje jakýsi "vývojový diagram" synchronizační úlohy, jenž poskytuje návod pro její strukturalizaci programu i popis paralelnosti a násobnosti procesů.
Definice p-sítě
p-sít je pětice R = <P, T, ∑, S, F> kde
P = (P1, ..., pn) je množina míst
T = (ti, ..., tm) je množina přechodů
∑ = je abeceda operaci
S Є P je počáteční místo
F Є P je koncové místo.
Každý přechod tj Є T je uspořádaná trojice
tj = (σj, Ij, Oj)
kde σj je operace přislušející přechodu tj,
Ij je multimnožina vstupních míst přechodu tj
Oj je multimnožina výstupních míst přechodu tj
Multimnožina se od množiny liší tím, že připouští více výskytů prvků množiny. Typickým příkladem multimnožiny je rozklad přirozeného čísla na prvočinitele, kde prvky multimnožiny jsou jednotliví prvočinitelé. Počet výskytů prvku x v multimnožině M je dán funkci #(x, M).
Platí
#(x, M)≥0; je-li #(x, M) = 0, pak neplatí, že x Є M.
Příklad:
Uvažujeme p-sít Ri (která: modeluje řídicí strukturu jednoduchého počítačového systému).
R1 - < {p1, P2, P3, P4, P5, P6, P7, P8},
{ tl, t2, t3, t4, t5, t6},
{a, b, c, d, e, f},
P1, P8>
kde prvky množiny T jsou tyto trojice:
t1 = (a, {p1}, {p2, p7, p4})
t2 = (b, {p2}, {p2, p3})
t3 = (c, {p3, p4}, {p5})
t4 = (d, {p5}, {p6, p7})
t5 = (e, {p6, p7}, {p7})
t6 = (f, {p7}, {p8})
Druhými a třetími složkami přechodu tj jsou v tomto případě množiny, nikoliv multimnožiny.
Přiklad: Příklad přechodu s multimnožinou vstupních a výstupních míst
t = (σ, {p1, p1, p1, p2, p2} , {p3, p3})
Pro snažší referenci jednotlivých složek přechodu tj zavedeme trojici funkcí přechodu tj; vstupní funkci I, výstupní funkci O a vyhodnocovací funkci σ, jejichž hodnotami jsou multimnožina výstupních míst, resp. prvek operací ∑
I(tj) = Ij, O(tj) = Oj a σ(tj) = σj

Vstupní a výstupní funkce přechodu jsou definovány jako multimnožiny míst. Analogicky definujeme vstupní a výstupní funkce místa pk, I(pk) a O(pk), jako množiny, pro něž platí

(tj, I(pk)) = # (pk, O(tj))
(tj, O(pk)) = # (pk, I(tj))

pro všechna pk Є P a Tj Є T

Příklad:

Pro p-sít Ri (viz příklad 1) dostáváme tyto vstupní,výstupní a vyhodnocovací funkce přechodů a vstupní a výstupní funkce míst:

I(t1) = {p1}	O(t1) = {p2, p7, p4}	σ(t1) = a
I(t2) = {p2}	O(t2) = {p2, p3}	σ(t2) = b
I(t3) = {p3, p4}	O(t3) = {p5}	σ(t3) = c
I(t4) = {p5}	O(t4) = {p6, p4}	σ(t4) = d
I(t5) = {p5, p6}	O(t5) = {p7}	σ(t5) = e
I(t7) = {p7}	O(t6) = {p8}	σ(t1) = f
I(p1) = O	O(p1) = {t1}	
I(p2) = {t1, t2}	O(p2) = {t2}	
I(p3) = {t2}	O(p3) = {t3}	
I(p4) = {t1, t4}	O(p4) = {t3}	
I(p5) = {t3}	O(p5) = {t4}	
I(p6) = {t4}	O(p6) = {t5}	
I(p7) = {t1, t5}	O(p7) = {t5, t6}	
I(p8) = {t6}	O(p8) = 0	

12.2 GRAF P-SÍTĚ

Graf p-sítě je orientovaný multigraf, jehož množina uzlů je sjednoceni množiny míst P a množiny přechodů T, a jehož množina hran je definována takto:

1. pro každý výskyt místa pk v multimnožině O(tj) vede hrana z přechodu tj do místa pk - výstupni hrana přechodu tj
2. pro každý výskyt místa pk v multimnožiné I(tj) vede hrana z místa pk do přechodu tj - vstupní hrana přechodu tj

Graficky jsou uzly příslušející místům a přechodům p-síti rozlišeny: místa jsou označována kroužkem a přechody úsečkou.

Evoluční pravidla p-sítě

Provádění p-sítě jako abstraktního stroje je řízené existenci příznaků, které v určitém počtu příslušejí každému místu p-sítě. Graficky se příznaky označují jako černé tečky umístěné v místech p-sítě. Dříve, než tato pravidla uvedeme, je nutné definovat dva nové pojmy.

Přechod ty se nazývá realizovatelný přechod, jestliže pro každé pk (leží) P existuje alespoň #(pk,I(tj)) příznaků v místě pk. Tzn., že vstupní místa přechodu ty mají tolik příznaků, že lze každé vstupní hraně přiřadit jeden příznak.

Přechod ty je realizován, jestliže se odstraní #(pk, I(tj)) příznaků z každého vstupního místa a přidá #(pi, O(tj)) příznaků každému výstupnímu místu pi. Realizován může být pouze realizovatelný přechod.

12.3 EVOLUČNÍ PRAVIDLA

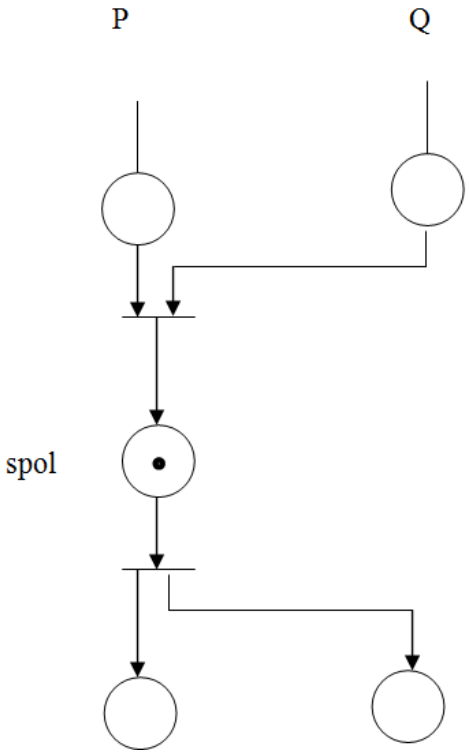
1. p-sít je inicializována umístěním jednoho příznaku do počátečního místa S
2. Vypočítá se množina realizovatelných přechodů W, W ≤ T
3. je-li W # 0, pak je vybrán jeden z realizovatelných přechodů z W a je realizován. V opačném případě provádění p-sítě skončilo.

Kroky (2) a (3) jsou opakovaně prováděny tak dlouho, až neexistuje žádný realizovatelný přechod. Kdykoliv je přechod realizován mění se počet nebo umístění příznaků v místech p-sítě a tím se mění stav p-sítě.
Příklad: Aplikaci evolučních pravidel ukážeme na příkladě p-sítě R2, zadané grafem p-sítě (obr.). Počáteční, resp. koncové místo je místo P1 , resp. p8
Modelováni prostřednictvím p-sítí p-sítě jako ředicí struktury modelů paralelních systémů se používají ve funkci modelů na takové hladině abstrakce, jež je vhodná pro popis a řízení interakci paralelních procesů. Základní komponenty p-síti, místa a přechody, popisuji podmínky a operace (abstrakce události) a jejich vzájemné vztahy.
Výskyt operace σj je obecně závislý na splnění určitých podmínek. Tyto podmínky jsou pro přechod tj modelující operaci specifikovány multimnožinou vstupních dat. Splnění podmínek pro výskyt operace σj odpovídá v předchozím odstavci definovanému pojmu - realizovatelnosti přechodu, tj. přechod je realizovatelný, právě když jsou splněny podmínky implikující možnost výskytu operace σj (vstupní místa mají dostatečný počet příznaků). Samotný výskyt operace pak odpovídá realizaci přechodu. Operace, která v systému nastala, může obecně vést ke splněni jiných podmínek, jež implikuji možnost výskytu jiných událostí. Změna platnosti podmínek systému po provedeni operace a~ je modelována přesunem příznaků ze vstupních míst Ij do výstupních míst Oj po realizaci přechodu tj.
Petriho sítě se používají pro znázornění paralelismu a synchronizace paralelních procesů. Místa se znázorňují kolečky a přechody čárkou kolmou na hrany. Místa a přechody se propojují orientovanými hranami tak, aby žádné dva sousední uzly nebyly téhož typu. Synchronizační úlohy budeme znázorňovat pomoci tzv. označkované Petriho sítě. V této síti je každému místu přiřazen určitý počet značek (znázorněny jako černé tečky). Přechodová pravidla pro označkovanou Petriho sít určují přípustné změny jejího značkováni. Každá změna značkováni se provede tak, že se provede jeden z jejich připravených přechodů. Přechod mezi místy p a q je připraven, jestliže každé misto p obsahuje alespoň tolik teček, kolik vede z místa p do t hran. Přechod se pak provede takto:

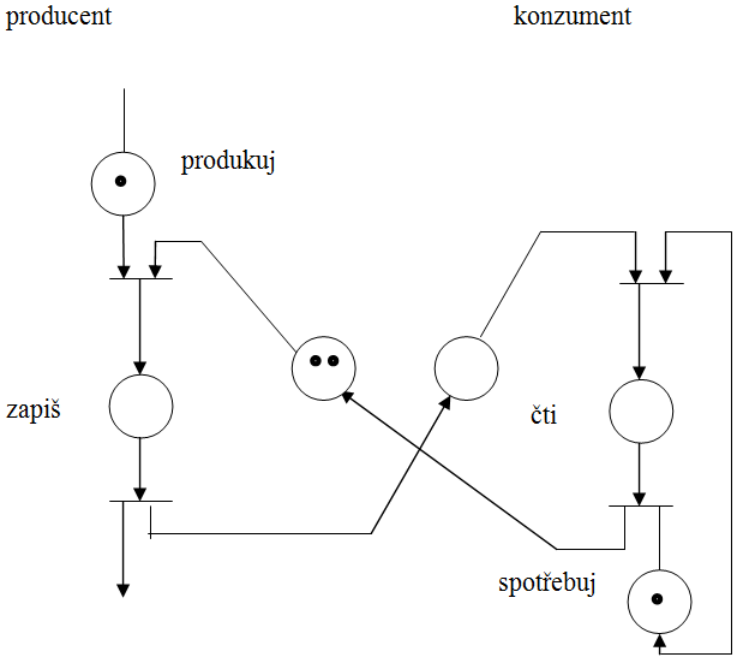
1. Z každého místa p se odeber tolik teček, kolik vede hran z místa p do místa t
2. Do každého místa q se přidá tolik teček, kolik vede hran z místa t do místa q.

Při modelování paralelních procesů používáme Petriho sítě tak, že každému příkazu bude odpovídat nějaké místo sítě. Tečka pak bude symbolizovat prováděni odpovídajícího příkazu. Přechod z příkazu A na příkaz B bude v Petriho síti znamenat provedení přechodu, který je mezi místy A a B. Běh procesu se proto v Petriho síti jeví jako pohyb tečky. Paralelismus se projeví existencí více teček.
souběh

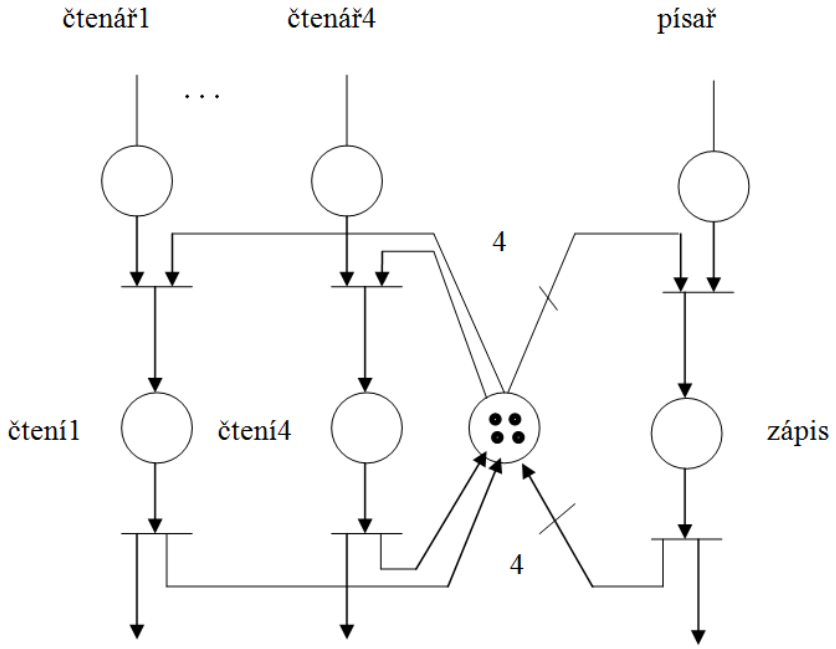
Souběh – rendezvous



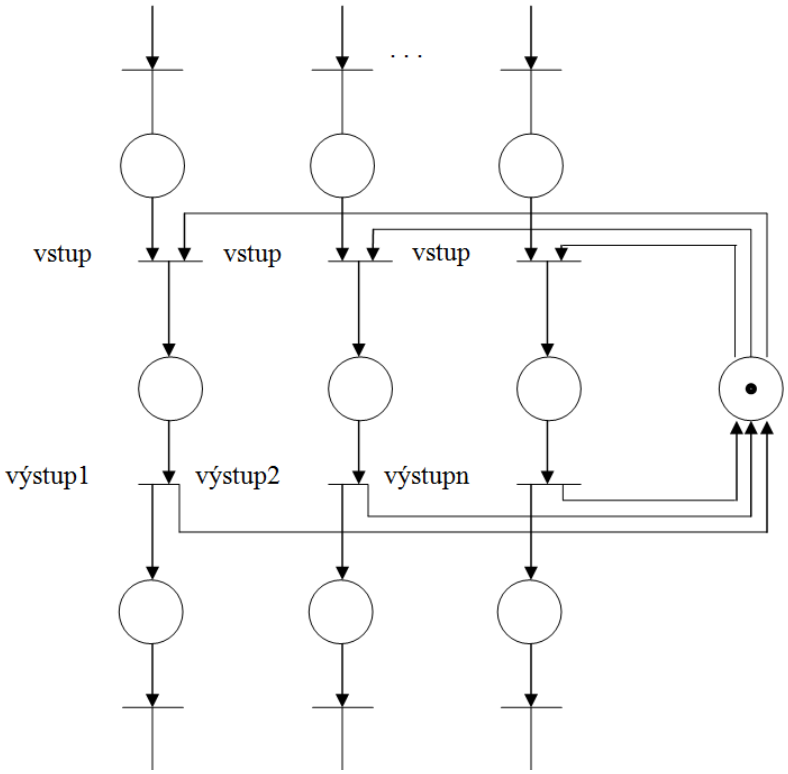
Producent a konzument



Čtenáři a písaři



Model kritických sekcí



Úkol zní: Jak zabránit časové závislým chybám?
Podstata řešení spočívá v synchronizaci procesů, tj. v koordinaci jejich postupu. Pokud nejsou splněny Bernsteinovy podmínky, je třeba rozvážit jednotlivé případy časové závislosti a ty které jsou nevhodné odstranit synchronizaci procesů.
Procesy synchronizujeme tak, že je doplňujeme o synchronizační operace.
Nejjednodušší synchronizační operace jsou ty, kterými koordinujeme postup procesu s tokem fyzikálního času. Příkladem takové operace je operace běžná ve všech RT - systémech, kterou často vyjadřujeme punkcí DELAY (T).
Jde o synchronizaci v čase absolutní, která způsobí pozastavení procesu na T časových jednotek (například milisekund). Použiti této instrukce je však problematické, neboť vyžaduje

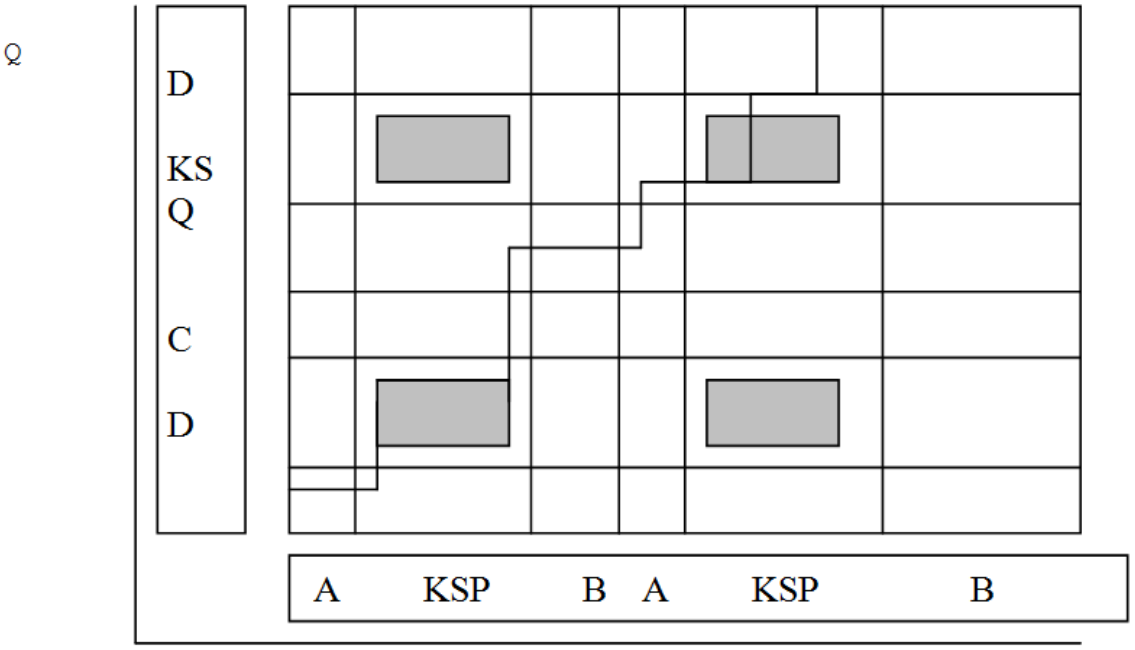
znalost toho, na jak dlouho máme řešení procesu pozastavit a vlastně vyžaduje vědomost o tom, jak se chovají ostatní procesy, resp. jaké jsou jejich okamžité nároky na sdílené zdroje. To je požadavek naprosto nereálný, protože ho nelze vlastně vůbec odpovědět v důsledku velmi dynamicky se měnícího prostředí; ať již jde o prostředí operačního systému, nebo technologického procesu.
Častějším, velmi obecným případem je použití takových synchronizačních operací, kterými dosahujeme uspořádání relativní rychlosti postupu procesů. Než přejdeme k dalšímu výkladu, seznámíme se s pojmem kritické oblasti, který v tomto výkladu použijeme.

KRITICKÁ OBLAST,POSTUPOVÁ CESTA

Uvažujme o dvou procesech P a Q. Představme si, že mají následující cyklickou strukturu

Proces P:	Proces Q:
LOOP A: KSP: B: END	LOOP C: KSQ: D: END,

Příkazy A,KSP,B,C,KSQ,D zastupují úseky instrukcí programu a jak naznačeno, cyklicky se opakují. Znázorníme si časový postup procesů P a Q na vodorovné a svislé ose.



Při běhu procesů dochází opakovaně k situaci, že má proces vstoupit do oblasti KSP, resp. KSQ. Tak je označena v obou procesech ta část kódu, která pracuje se společnými proměnnými nebo objekty obou procesů a kde je nebezpečí, že jeden proces naruší chod druhého procesu. Tato část kódu programu se proto nazývá kritická sekce, nebo kritická oblast a ty kritické sekce všech procesů, které operuji na společných objektech se nazývají spražené kritické sekce, nebo spražené kritické oblasti.
Při souběžném chodu procesů dochází k přepínáni kontextu a k střídavému využívání procesoru procesy, v našem případě dvěma. To je na obrázku znázorněno klikatou čarou, která se nazývá postupová cesta a znázorňuje střídáni procesů při využiti procesoru. Je zřejmé, že je žádoucí aby postupová cesta neprocházela přes průnik kritických sekcí procesů, který je na obrázku vyznačen tmavě a který nazýváme zakázaná oblast.
Jak to zařídit je úkolem synchronizace. Podíváme-li se na obrázek, vidíme, že máme v podstatě dvě možnosti.

1. Postupová cesta vede tak, aby neprocházela zakázanou oblastí, což znamená, že musíme v určitém okamžiku omezit možnost přepínáni kontextu.
2. Zakázaná oblast se odsune jinam, aby postupová cesta prošla mimo ni.

Podstatou prvního přístupu je metoda tzv. pasivního čekání, zatímco v druhém případě jde o přistup, jehož podstatou je metoda tav. aktivního čekáni. Je nyní pouze otázkou, jakými prostředky dosáhneme efektu požadovaného v bodě 1) nebo 2). Těmito prostředky jsou synchronizační nástroje.

12.4 SYNCHRONIZAČNÍ NÁSTROJE

Synchronizační nástroje jsou technické nebo programové prostředky, kterými umožňujeme synchronizovaný postup kooperujících procesů v čase. Ukažme si nejdříve obecnou ideu, které stoji v základě jakéhokoliv synchronizačního prostředku.
Představme si, že máme synchronizovat dva procesy P a Q a dále, že máme k dispozici zatím ještě blíže nespecifikovanou spraženou dvojici operaci Z a K s vlastností, že jakmile některý proces provede operaci Z, nemůže jiný proces tuto operaci provést, dokud první proces neprovede operaci K.
Tyto operace můžeme použít k tomu, abychom zabránili současnému vstupu procesů do zakázané oblasti následovně. Dříve, než libovolný proces vstoupí do kritické oblasti, vykoná synchronizační operaci Z. Následkem toho se postup druhého (ostatních)procesů pozastaví. Po výstupu zkritické oblasti provede první proces operaci K, která povolí provedeni operace Z jinému procesu a tím umožní čekajícím procesům vstoupit do jejich kritických oblastí.
Jsou asi zřejmé dvě věci:

1. Předpokladem úspěšné synchronizace je, aby procesy (tj. programátoři !!!) postupovali korektně, ukázněně, aby používali synchronizační operace vždy při vstupu a výstupu do a z kritické oblastí.
2. Implementovat synchronizační operace Z a K nebude asi úplně jednoduché, protože minimálně operace Z, nebo alespoň její část musí být nedělitelná. To proto, aby během jejího provedení nemohl být procesor předělen jinému procesu, který by vstoupil do téže synchronizační operace. Současně je třeba řešit otázku co s procesem, který nemůže provést nebo dokončit operaci Z.

Není proto divu, že existuje více různých synchronizačních nástrojů, které řeší v podstatě tentýž problém. že to dáno různými přístupy k jejich implementaci a vyplývá 2 toho i to, že se v různých situacích mohou jednotlivé nástroje různě osvědčit. Ze shora uvedeného obecného modelu synchronizace však vychází většina universálních synchronizačních nástrojů.