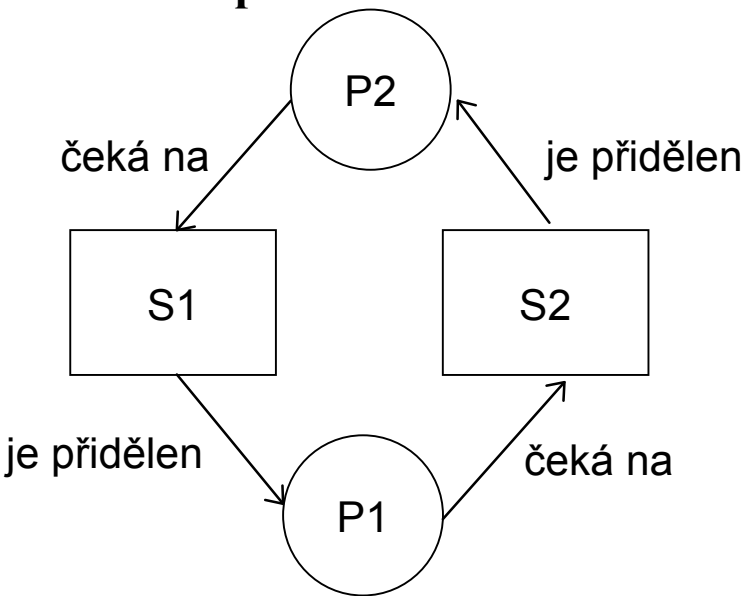
10. Přidělování prostředků a uváznutí

Uváznutí - čekání na událost, která nemůže nastat díky čekání

Příklad:

```
P1:     lock(S1);          P2:     lock(S2);
        lock(S2);                  lock(S1);
```

Graf alokace prostředků:



Kdy nastává uváznutí?

1. Zamykání semaforů, zámků, souborů, záznamů - viz výše
2. Přidělování paměti, V/V bufferů

```
P1:     request(M);        P2:     request(M);
        request(M);                request(M);
```

   Pokud zbývají 2 jednotky prostředku M, mohou se přidělit po jedné P1 a P2 a žádný proces nemůže být dokončen

3. Zasílání zpráv

```
P1:             P2:             P3:
receive(p3);    receive(p1);    receive(p2);
send(p1);       send(p2);       send(p3);
```

   Stačí dva procesy, ale většinou více.

Kdy nastává uváznutí?

1. Přidělený prostředek může používat pouze jeden proces.
2. Proces, který má přidělené prostředky, se při alokaci dalších nevzdá přidělených prostředků, uvolní je až po ukončení.
3. Proces získává prostředky sekvenčně, oddělenými alokacemi.
4. Prostředek nemůže být preemptivně odebrán, proces uvolňuje prostředky explicitně.

Problém uváznutí:

1. **Detekce** - jak zjistit, které procesy v paralelním systému uvázly? (Stačí cyklus?)
2. **Zotavení** - jak se nejlépe dostat z uváznutí?
3. **Prevence** - jak se vyhnout uváznutí (stačí jiná posloupnost alokací?)

Dijkstra (1965), R.C.Holt (1971)

Model systému:

P_i procesy
R_i prostředky
C_i kapacita prostředku R_i

Přechody stavů systému:

1. požadavek (request)
2. přidělení (allocation)
3. uvolnění (release)

Typy prostředků:

SR - opakovaně použitelné - serially reusable
CR - jednorázově použitelné - consumable resources

Stav systému

- Stav systému reprezentuje stav alokace prostředků v systému.
- Stav systému je měněn procesy při požadavku, získání nebo uvolnění prostředku.
- Pokud není proces v daném stavu systému blokovaný (čeká na přidělení), může potenciálně změnit stav systému.

Def.:

Systém: (σ, π, S_0)

Stavy systému: $\sigma = \{ S_0, S_1, S_2, ... \}$

Akce procesů: $\pi = \{p_1, p_2, ... \}$

Počáteční stav: S_0 (všechny prostředky volné)

Změny stavu: $p_i: \sigma \rightarrow \sigma^N$, $p_i(S_x) = \{S_j\}$, $S_j \in \sigma$
proces P_i může změnit stav systému S_x (žádostí, přidělením, uvolněním) na některý stav z množiny stavů $\{S_j\}$

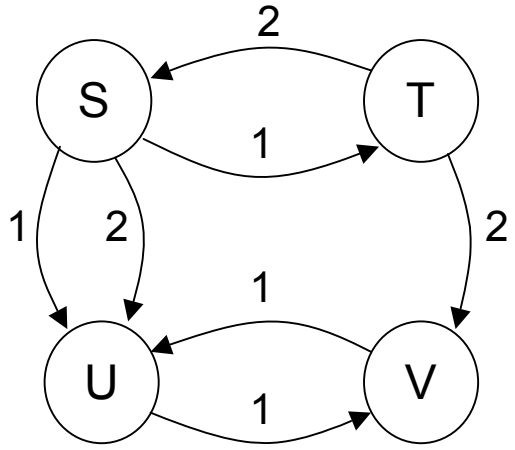
P_i může změnit stav S na stav T (p_i): $S \xrightarrow{i} T$

Systém může přejít ze stavu S do T: $S \xrightarrow{*} T$

$$S \xrightarrow{*} T \Leftrightarrow S = T \vee \exists p_i,\ S \xrightarrow{i} T \vee \exists p_i,\ S \xrightarrow{i} W \wedge W \xrightarrow{*} T$$

Příklad:

$\sigma = \{S, T, U, V\}$
$\pi = \{p_1, p_2\}$
$p_1(S) = \{T, U\}$
$p_1(T) = \phi$
$p_1(U) = \{V\}$
$p_1(V) = \{U\}$
$p_2(S) = \{U\}$
$p_2(T) = \{S, V\}$
$p_2(U) = \phi$
$p_2(V) = \phi$



Def.: Proces P_i je blokován v daném stavu S, pokud nemůže žádným způsobem změnit stav systému (způsobit přechod z S):

$$\nexists \mathrm{T},\ S \xrightarrow{i} T$$

Def.: Proces P_i uváznul v daném stavu S, pokud je blokován ve stavu S a bez ohledu na následující změny stavu systému zůstává stále blokovaný:

$$\forall \mathrm{T},\ S \xrightarrow{*} T\ ,\ P_i \text{ je blokovaný v T}$$

Příklad:
proces P_1 je blokovaný ve stavu T
proces P_2 je uváznutý ve stavech U a V

Def.: Stav S je stavem uváznutí, pokud existuje proces P_i uváznutý ve stavu S.

Princip prevence uváznutí: omezení přechodů mezi stavy systémů tak, aby všechny dostupné stavy nebyly stavem uváznutí.

Def.: Stav S je bezpečný, pokud $\forall \mathrm{T},\ S \xrightarrow{*} T$, T není stavem uváznutí.

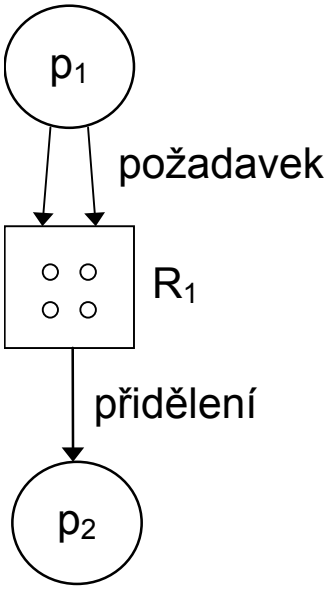
Problém: ke konstrukci grafu stavu systémů je nutno znát množinu přechodů = akce požadavek, přidělení a uvolnění, což je dopředu neznámé. Pro detekci uváznutí (resp. vyhnutí se) je nutný jiný mechanismus.

SR prostředky (opakovaně použitelné)

Další vlastnosti:

1. Prostředek se skládá z konstantního počtu stejných jednotek
2. Jednotka prostředku je buď volná nebo přidělená
3. Proces může uvolnit jednotku, pokud ji má přidělenu

Graf alokace SR prostředků (Resource Allocation Graph) = 1 uzel grafu stavů systému



orientovaný bipartitní multigraf (N, E, σ)

$N = \pi \cup \rho,\ \pi \cap \rho = \phi$ množina uzlů

$\pi = \{p_1,..., p_n\}$ procesy

$\rho = \{R_1,...,R_m\}$ prostředky

$E = \{e_1,...,e_k\}$ množina hran

$\sigma: E \rightarrow N \times N,$ incidence grafu

$\sigma(e) = (a,b),\ a \in \pi \wedge b \in \rho \vee E\ a \in \rho \wedge b \in \pi$

požadavek *přidělení*

Změny stavu systému = změny v grafu alokace SR prostředků

C_i kapacita prostředku R_i

r_i počet volných jednotek prostředku R_i

Omezení přechodů:

$$\sum_{i \in \pi} \left|(R_j, p_i)\right| \le C_j, \forall j \in \rho$$

O1: Prostředek může být přidělen do max. kapacity

O2: Proces nesmí žádat o více než je kapacita prostředku

$$\left|(R_j, p_i)\right| + \left|(p_i, R_j)\right| \le C_j, \forall j \in \rho, \forall i \in \pi$$

Změny stavu grafu:

1. Požadavek

Pokud proces p_i nemá žádné požadavky, pak může žádat k jednotek prostředku R_j (při splnění O1, O2).

$E' = E \cup F, \quad |F| = k$

$\sigma' = \sigma \cup \sigma_f, \quad \sigma_f(e_i) = (p_i, R_j), \forall e_i \in F$

2. Přidělení

Pokud má proces p_i požadavek na k jednotek prostředku R_j a požadavek je uspokojitelný:

$$\left|(p_i, R_j)\right| + \sum_{k=1,n} \left|(R_j, p_k)\right| \leq C_j$$

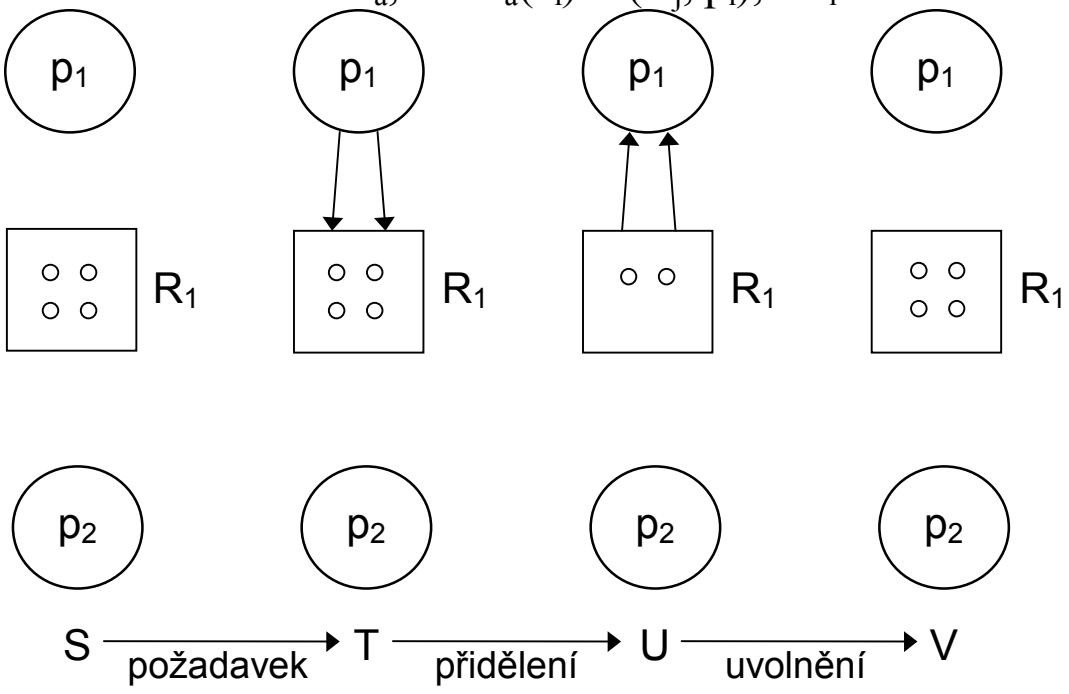
	požadavek	přidělení
$E' = E \cup A - F,$	$\lvert A\rvert = \lvert F\rvert = k,$	$r_j = r_j - k$
$\sigma' = \sigma \cup \sigma_a - \sigma_f,$	$\sigma_a(a_i) = (R_j, p_i), \forall a_i \in A$	
	$\sigma_f(e_i) = (p_i, R_j), \forall e_i \in F$	

3. Uvolnění

Pokud nemá proces p_i žádný požadavek a má přiděleno k jednotek prostředku R_j, může uvolnit l jednotek prostředku:

$E' = E - A, \quad |A| = l \leq k, \quad r_j = r_j + l$

$\sigma' = \sigma - \sigma_a, \quad \sigma_a(a_i) = (R_j, p_i), \forall a_i \in A$



1. Detekce uváznutí

Je třeba detekovat, zda některý proces může pokračovat a převést systém do jiného stavu (požadavkem, přidělením, uvolněním).

Proces p_i je **zablokován**, pokud nemůže provést žádnou operaci
- nastává v případě, že p_i má neuspokojitelné požadavky, tj.

$$\exists j, \left|(p_i, R_j)\right| + \sum_k \left|(R_j, p_k)\right| > C_j$$

Protože požadavek nemůže být uspokojen, nemůže p_i generovat další požadavek, ani uvolnit prostředky.

Redukce grafu alokace prostředků nezablokovaným procesem p_i, který není izolovaný (má požadavky nebo prostředky) = odstranění všech hran vedoucích z/do p_i. Reprezentuje přidělení všech požadovaných prostředků, dokončení procesu a uvolnění všech přidělených prostředků tomuto procesu (neřešíme případné budoucí požadavky tohoto procesu na další prostředky, ty se budou řešit až vzniknou = jiný stav systému).

Neredukovatelný graf - nelze jej redukovat žádným procesem.

Úplně redukovatelný graf - existuje sekvence redukcí, která odstraní všechny hrany grafu.

Věta: Všechny sekvence redukcí grafu alokace SR prostředků vedou ke stejnému neredukovatelnému grafu.
Význam: není nutno testovat různé sekvence redukcí - když nalezneme jednu, která generuje neprázdný (prázdný) graf, ostatní jsou ekvivalentní.

Věta: Paralelní systém je ve stavu uváznutí právě tehdy, když graf alokace prostředků v tomto stavu není úplně redukovatelný.

Algoritmus detekce:

1. Opakovaný průchod seznamem procesů a postupné redukce. Při každé úspěšné redukci je třeba průchod opakovat. Složitost: $O(m.n^2)$

2. Pomocná informace w_i - počet prostředků, na které čeká p_i

$L := \{p_i | w_i = 0\}$;
for all p **in** L **do**
begin
    for all R_j, $|(R_j, p)| > 0$ **do**
    begin
        $r_j := r_j + |(R_j, p)|$;
        for all p_i, $0 < |(p_i, R_j)| \leq r_j$ **do**
        begin
            $w_i := w_i - 1$;
            if $w_i = 0$ **then** $L := L \cup \{p_i\}$;
        end
    end
end;
if $L \neq \{p_1,...,p_n\}$ **then** DEADLOCK;

Složitost: O(m.n)

Další výsledky:

Věta: Nutnou podmínkou uváznutí je cyklus v grafu alokace SR prostředků. (Pokud graf neobsahuje cyklus, nemůže nastat uváznutí, opačně to ale neplatí)

Speciální případy systémů:

a) Systém s okamžitým přidělením
Pokud je požadavek uspokojitelný, je při požadavku realizováno okamžitě přidělení. V grafu alokace SR prostředků pak zůstávají pouze neuspokojitelné požadavky.

Věta: V systému s okamžitým přidělováním je "knot" v grafu alokace SR prostředků postačující podmínka pro uváznutí.

Knot = silná komponenta, ze které nevede žádná hrana ven, ale pouze dovnitř. Silná komponenta grafu je maximální silně souvislý podgraf.

b) Systém s prostředky kapacity 1

Věta: V systému s jednotkovými prostředky je nutnou a zároveň postačující podmínkou uváznutí cyklus v grafu alokace.

c) Systém s jednotkovými požadavky
Proces může žádat jedním požadavkem pouze jeden prostředek.

Věta: Systém s jednotkovými požadavky je ve stavu uváznutí právě tehdy, když graf alokace prostředků po uspokojení všech neblokovaných požadavků (g.usp.p) obsahuje knot.

Algoritmus detekce knotu:

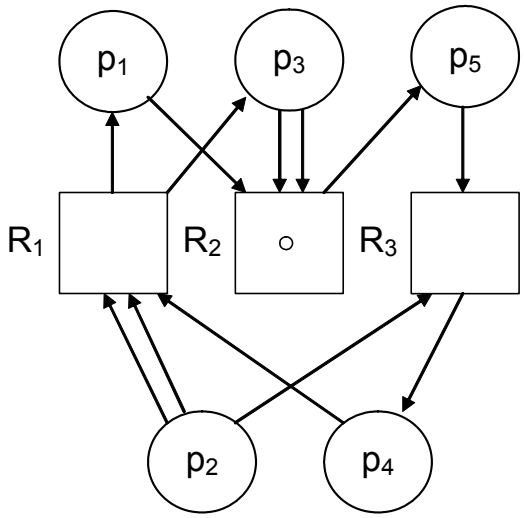
```
S := {i | i je listem};        list = uzel, ze kterého nevedou hrany
for all i in S do
      for all j, existuje hrana (j, i) do
            if not j in S then S := S ∪ {j};
      end;
end;
knot := ¬(S=N);
```

Příklad:

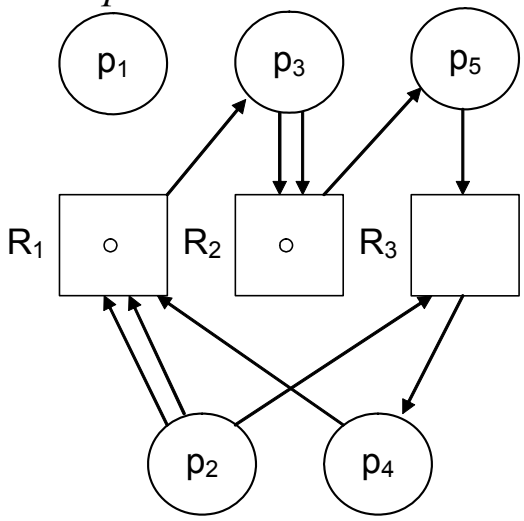
Systém se nachází v tomto stavu alokace SR prostředků:

	alokováno			**požadavky**			**počáteční kapacita**		
	R1	**R2**	**R3**	**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
P1	1	0	0	0	1	0	2	2	1
P2	0	0	0	2	0	1			
P3	1	0	0	0	2	0			
P4	0	0	1	1	0	0			
P5	0	1	0	0	0	1			

Zakreslete graf alokace prostředků reprezentující tento stav a určete, zda je systém ve stavu uváznutí (graf obsahuje cyklus).



Graf alokace prosředků v daném stavu



Redukce procesem P1, atd.

2. Zotavení z uváznutí

- Násilné ukončení některého uváznutého procesu
- Odebrání přidělených prostředků

Výběr procesu:
1. priorita procesu
2. cena znovuprovedení
3. typ procesu (systémový, uživatelský, apod.)

Cílem je vybrat proces s nejnižší cenou, ukončit jej a uvolnit tím prostředky. Graf alokace je tím redukován a celé se to opakuje až do té doby, dokud není úplně redukovatelný.

Algoritmus minimální ceny: hledá minimální podmnožinu ukončovaných procesů, která odstraní uváznutí a má nejnižší cenu.
Složitost: $O(n!)$, pro $n=m$

Pro speciální případy je realizovatelný - systém s jednotkovými požadavky, systém s okamžitou alokací, stačí zrušit knot = najít v něm proces, jehož zrušení má nejnižší cenu.
Složitost: $O((n+m)^2)$

Odebrání prostředků:
přímé odebrání – proces musí čekat na přidělení, čekání je ukončeno se stavem „prostředky byly odebrány“
nepřímé odebrání – návrat k předchozímu známému stavu (rollback na checkpoint)

3. Prevence

Prevence spočívá ve vyhnutí se stavům uváznutí = omezení přechodů na přechody, které jsou bezpečné.

Metody:

1. Povolit sdílené použití prostředků (spooling) – porušení podmínky výlučného používání
2. Přidělit prostředky jen jednomu procesu (silně omezující) – nemůže vzniknout cyklus v grafu alokace prostředků
3. Přidělovat vždy všechny požadované prostředky najednou jedním požadavkem (zbytečné blokování prostředků, které nejsou využity po celý běh procesu) - dtto
4. Vzdání se přidělených prostředků při požadavku na další (aplikovatelné u zámků v databázích, ale ne u prostředků typu tiskárna) – dtto
5. Podmíněný požadavek na prostředek – pokud již nějaké prostředky proces vlastní, požadavek na další je realizován pouze pokud je prostředek volný, jinak končí signalizací neúspěšnosti (neblokuje a nečeká) – proces musí vrátit prostředky a pak zkusit znovu
6. Žádat prostředky vždy ve vzrůstajícím pořadí očíslovaných tříd prostředků (nejpoužívanější metoda, nemůže vzniknout cyklus, **locking order**)

Příklad:

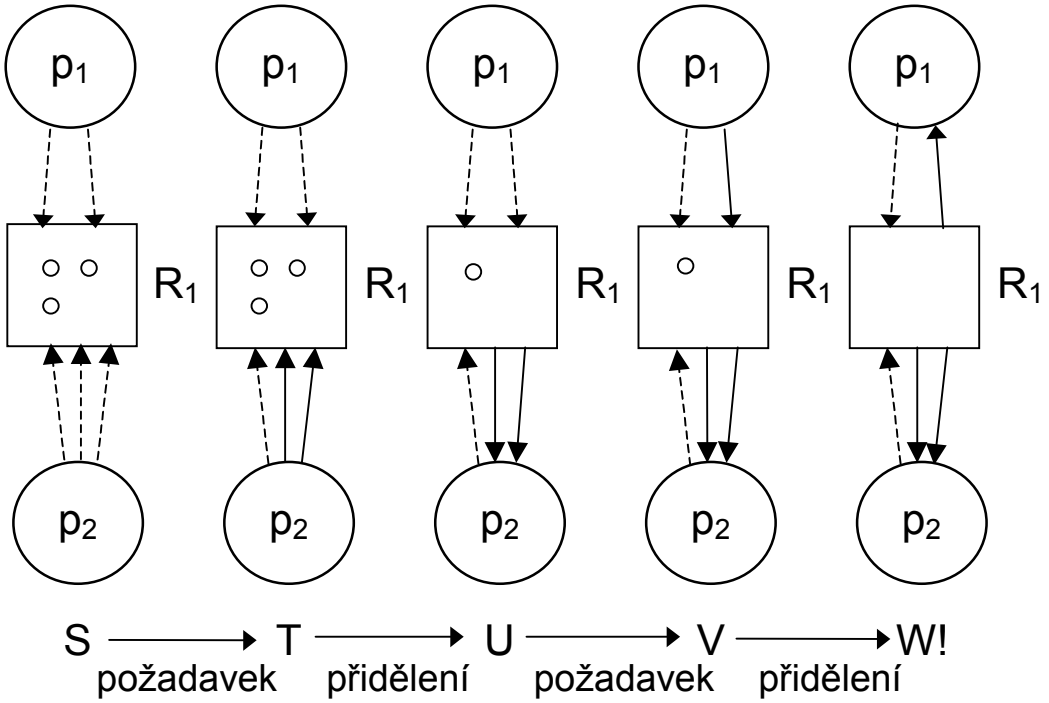
Pořadí S1, S2.

P1:	P2:
lock(S1)	lock(S2);
nyní může provést lock(S2);	**nyní nesmí provést lock(S1)** = má nižší pořadí než prostředek, který má přidělen

7. Bankéřův algoritmus - systém s maximálními požadavky (Dijkstra – 1965)
Deklarace maximálních požadavků na prostředky - c_{ij} pro p_i (maximální čerpání z účtu v měně, finanční prostředky v různých měnách = prostředky s kapacitou, cílem je uspokojit požadavky klientů a nezbankrotovat)
Omezení přechodů:
O3: přidělené + požadované jednotky nesmí překročit deklarované maximum:

$$\left|(p_i, R_j)\right| + \left|(R_j, p_i)\right| \leq c_{ij} \leq C_j, \forall j \in \rho, \forall i \in \pi$$

Graf alokace prostředků je zakreslován včetně hran maximálních požadavků:



Stav W není stavem uváznutí, ale pokud by p_1 nebo p_2 požádal o další jednotku (což může), došlo by k uváznutí. Přidělení R_1 pro p_1 nesmí být uskutečněno, dokud se nějaké jednotky nevrátí. Požadavek a přidělení pro p_2 může být uskutečněno.

Princip algoritmu: Povolit pouze takové přidělení, po kterém existuje alespoň jedna posloupnost uspokojení maximálních požadavků všech procesů.

Úplná redukovatelnost grafu maximálních požadavků - je třeba odstranit všechny hrany, včetně hran reprezentujících zatím nerealizované požadavky. Problém – je třeba testovat všechny postupy, složitost O($m.n^2$).

Věta: Pokud je systém v bezpečném stavu a není povoleno **přidělení** prostředku, jehož výsledkem je graf, který není úplně redukovatelný, pak jsou všechny stavy systému bezpečné a nemůže nastat uváznutí.

Kontinuální detekce - pokud je systém v bezpečném stavu, pak je graf úplně redukovatelný po přidělení požadovaných prostředků procesu p_i právě tehdy, když je možná redukce procesem p_i.
Význam: Přidělení může být realizováno, pokud je *počet volných jednotek prostředku* R_j >= *maximální počet* o kolik může ještě požádat proces p_i.

Algoritmus detekce bezpečného stavu:
alloc[i, j] = Prostředek R_j je přidělen procesu P_i
1. kopie *alloc* do *newalloc*
2. pokud je *newalloc*[i, j] rovno 0 pro všechny i,j, je stav bezpečný
3. pro všechny prostředky *avail*[j] = C_j - *newalloc*[*, j]
4. vyhledat P_i, pro který platí c_{ij} - *newalloc*[i, j] <= *avail*[j] pro všechny prostředky 1<=j<=m
5. pokud takový neexistuje, stav není bezpečný
6. nastavit *newalloc*[i, *] na 0 (uvolnění všech prostředků P_i).
7. skok na krok 2

Příklad:
Stav bez neuspokojených požadavků:

	přiděleno - alloc			**max. (c_{ij})**			**může žádat**		
	R_0	**R_1**	**R_2**	**R_0**	**R_1**	**R_2**	**R_0**	**R_1**	**R_2**
P_0	0	1	0	7	5	3	7	4	3
P_1	2	0	0	3	2	2	1	2	2
P_2	3	0	2	9	0	2	6	0	0
P_3	2	1	1	2	2	2	0	1	1
P_4	0	0	2	4	3	3	4	3	1
	přiděleno			kapacita			volno (avail)		
prostř.	7	2	5	10	5	7	3	3	2

1. Je stav bezpečný? **Návod:** nalézt řádek, na kterém je volno větší než může žádat = maximální požadavek uspokojen, tento proces odstranit, uvolnit všechny prostředky, které měl, zvýšit o tento počet volno a to celé opakovat. Pokud nezůstane nic, nalezli jsme cestu.
2. Lze uspokojit požadavek P1 na (1, 0, 2)?
3. Lze uspokojit požadavek P4 na (3, 3, 0)?
4. Lze uspokojit požadavek P0 na (0, 2, 0)?

Prostředky s kapacitou 1

Věta: Všechny stavy jsou bezpečné právě tehdy, když neorientovaný graf maximálních požadavků neobsahuje neorientovaný cyklus.

Příklad:
5 filozofů: Vidličku lze přidělit vždy, pokud není poslední zbývající. Pokud je poslední, může být přidělena pouze filozofovi, který už jednu vidličku má. (pozor - neřeší stárnutí)

CR prostředky (jednorázově použitelné)

- počet dostupných jednotek prostředků se mění přidělováním (konzumace) a uvolňováním (produkce)
- počet jednotek není omezen
- zasílání zpráv, signály

1. Neomezený systém

Stav systému se mění při: požadavku, přidělení a uvolnění

Věta: Systém uváznul právě tehdy, když jsou všechny procesy zablokované.

Pokud není některý proces zablokovaný, může uvolnit (vygenerovat) libovolný počet jednotek kteréhokoli prostředku a tím ostatní procesy odblokovat.

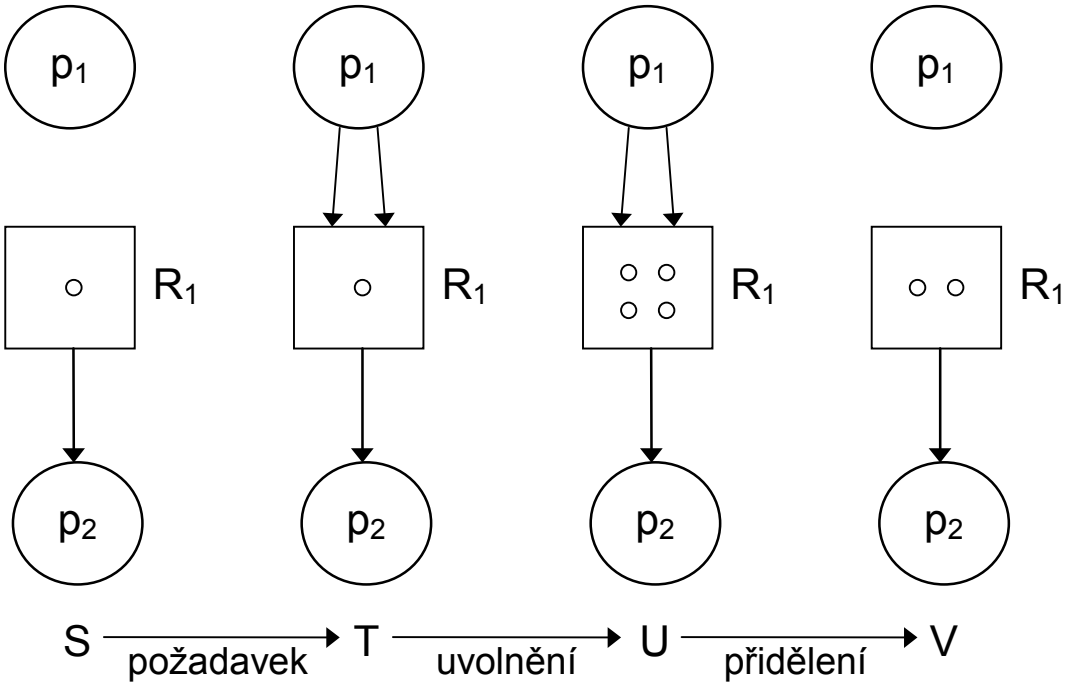
Věta: Žádný stav systému není bezpečný (nejde prevence). Nezablokované procesy mohou požádat postupně o více, než je k dispozici.

2. Systém se známými producenty

Graf alokace CR prostředků:

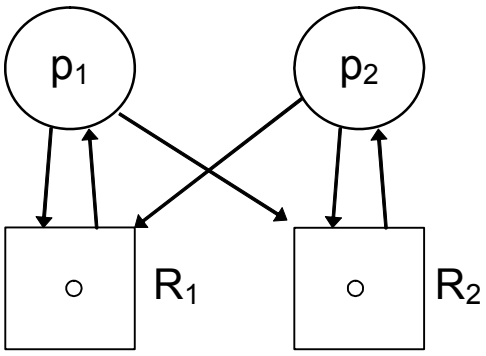
- orientovaný bipartitní multigraf (N, E, σ)
- množina hran E
- množina uzlů $N = \pi \cup \rho$, $\pi = \{p_1,...,p_n\}$, $\rho = \{R_1,...,R_m\}$
- $\sigma: E \rightarrow N \times N$, $\sigma(e) = (a,b)$, $a \in \pi \wedge b \in \rho \vee a \in \rho \wedge b \in \pi$
  požadavek *produkce*
- hrany produkce zůstávají v grafu trvale
- počet volných jednotek prostředku R_i: r_i

Pravidla pro přechody:
Požadavek - proces p_j, který nemá žádný požadavek, může požádat o konečný počet jednotek prostředku R_i; výsledný graf je doplněn o hrany (p_j, R_i) dle počtu požadovaných jednotek.
Přidělení - požadavek procesu p_j na prostředek R_i může být uspokojen, pokud je $r_i \geq |(p_j, R_i)|$; ve výsledném grafu jsou zrušeny hrany (p_j, R_i) a $r_i = r_i - |(p_j, R_i)|$.
Uvolnění - proces p_j, který nemá žádný požadavek a je producentem prostředku R_i, může vyprodukovat konečný počet jednotek prostředku n; ve výsledném grafu je $r_i = r_i + n$.



Redukce grafu: graf může být redukován procesem p_i, který není zablokovaný (nemá neuspokojitelný požadavek) a není izolovaný (má požadavek nebo je producentem).
1. Uspokojení požadavků p_i zrušením všech hran (p_i, R_j) a odpovídajícím snížením r_j
2. Pro všechny R_j, pro které je p_i producentem, nastavit dostatečný počet jednotek na uspokojení všech požadavků = ω $(\forall i, \omega > i \wedge \omega + i = \omega - i = \omega)$

Zatímco počet jednotek prostředku se při redukci SR grafu zvyšuje, u CR grafu se může uspokojováním (spotřebou) zmenšovat → pořadí redukcí je důležité!
Věta o ekvivalenci různých sekvencí redukcí SR grafu zde neplatí. Stejně tak neplatí věta o ekvivalenci stavu uváznutí a neexistující úplné redukce grafu.
Příklad:



Graf není úplně redukovatelný (pro druhý proces nezbude žádná jednotka), přitom není stavem uváznutí (nejsou všechny procesy zablokované).

Věta: Proces p_i neuváznul ve stavu S právě tehdy, když existuje v tomto stavu S sekvence redukcí grafu alokace CR prostředků vedoucí ke stavu T, ve kterém není p_i zablokován (nemá neuspokojitelný požadavek v grafu CR prostředků).

Pro detekci stavu uváznutí je třeba detekovat uváznutí pro všechny zúčastněné procesy = pro každý zjistit, zda neuváznul. Nestačí tedy najít jednu sekvenci vedoucí na neredukovatelný graf, protože jiná sekvence redukcí může vést na úplně redukovatelný graf - O(n!). Pokud existuje aspoň jedna cesta úplné redukce grafu, systém není ve stavu uváznutí.

SR prostředky - uváznutí nastává pouze při požadavku, CR prostředky - uváznutí může nastat při požadavku i přidělení.

Věta: Žádný stav systému není bezpečný (nejde prevence).
Věta: Nutnou podmínkou uváznutí je cyklus v grafu alokace CR prostředků (stejně jako u SR).

Systém s okamžitým přidělováním

Věta: Pokud jsou všechny uspokojitelné požadavky okamžitě realizovány (v systému jsou pouze blokované požadavky), je knot v grafu alokace CR postačující podmínkou uváznutí.

Systém s jednotkovými požadavky

Věta: Systém uváznul právě tehdy, když graf alokace CR prostředků, který neobsahuje žádné uspokojitelné požadavky (g.usp.p) obsahuje knot (jako SR).
Pokud jsou v grafu uspokojitelné požadavky, tak je nejprve odstraníme jejich uspokojením (odebereme hrany a jednotky prostředků).

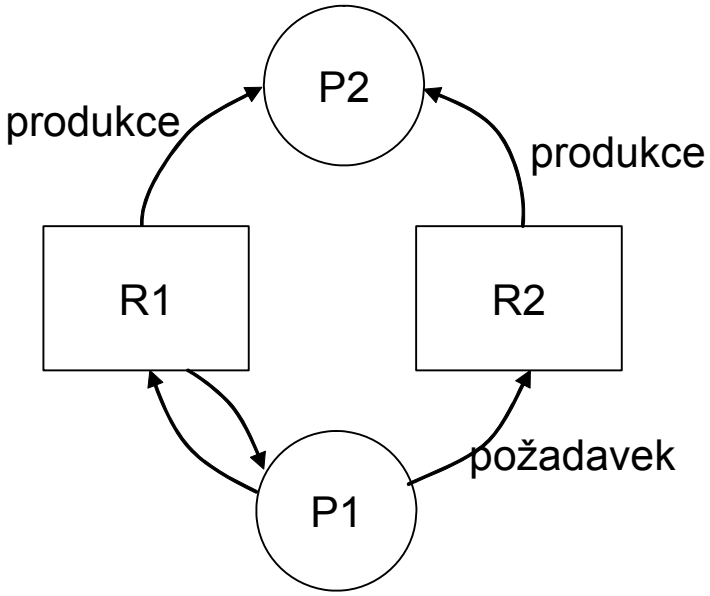
Zotavení: zrušení zablokovaného procesu, pokud možno ne producenta nějaké prostředku - zaslání chybových zpráv.

Prevence: nelze

3. Systém se známými producenty a konzumenty

Pro každý prostředek je definována množina konzumentů a producentů. V tomto systému lze sestrojit algoritmus prevence.

Graf blokovaných požadavků (claim-limited) = graf CR prostředků, kde všichni konzumenti mají požadavek na právě jednu jednotku prostředku a všechny prostředky jsou prázdné.



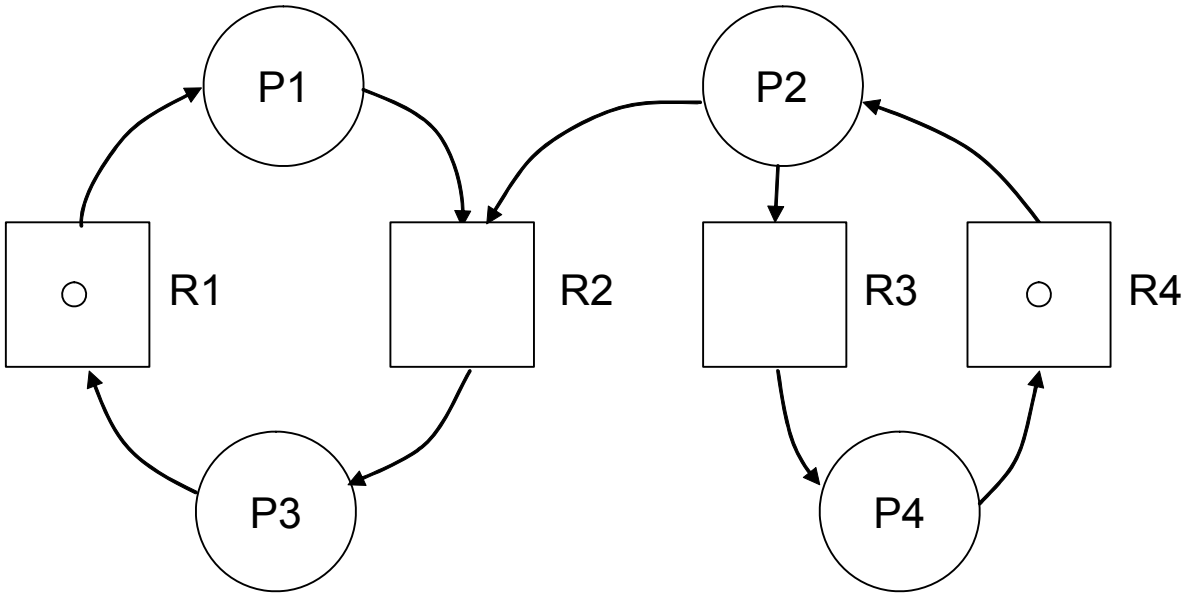
Věta: Všechny stavy systému jsou bezpečné, pokud je graf blokovaných požadavků úplně redukovatelný.

Věta: Na pořadí redukcí nezáleží, každá sekvence vede ke stejnému neredukovatelnému grafu.

Aplikace: viz SR systémy, obdobný význam cyklu v grafu (nutná podmínka uváznutí, vyhnutím se cyklu lze vyhnout se uváznutí).

Důsledek: systém není bezpečný, pokud neexistuje alespoň jeden proces, který nepožaduje (nekonzumuje) žádné prostředky.

Příklad:



1. Které procesy jsou zablokované?
2. Je tento stav stavem uváznutí?
3. Obsahuje graf knot? Lze použít detekci knotu pro detekci uváznutí (co bychom museli s grafem udělat, viz g.usp.p)?

Shrnutí

	podmínky neuváznutí	***podmínky uváznutí***
SR prostředky	*redukovatelný graf <=> nenastalo uváznutí*	*knot g.usp.p => uváznutí* *uváznutí => cyklus*
CR prostředky	*redukovatelný graf => nenastalo uváznutí*	*knot g.usp.p => uváznutí* *uváznutí => cyklus*
CR claim-lim.	*redukovatelný graf <=> nenastalo uváznutí*	
jednotkové požadavky SR 1prostředky		*knot g.usp.p <=> uváznutí* *cyklus <=> uváznutí*