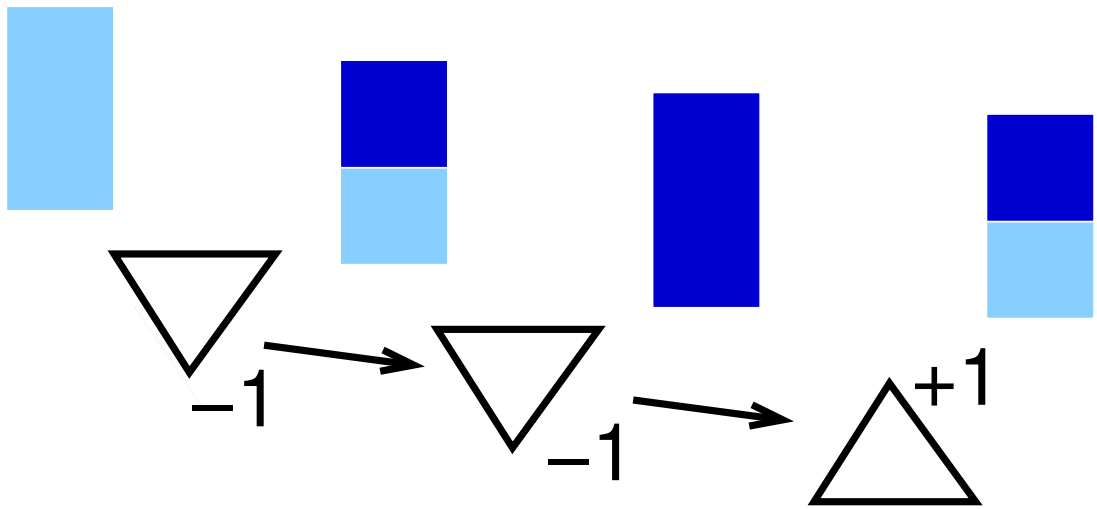
Rozvrhování s omezujícími podmínkami (dokončení)

21. března 2023

Produkovatelné/spotřebovatelné zdroje

- Zdroj = rezervoár
- Aktivita konzumuje nějaké množství zdroje cap(A)<0 nebo
  aktivita produkuje nějaké množství zdroje cap(A)>0
- Požadována minimální kapacita zdroje (při konzumaci)
  a maximální kapacita zdroje nemůže být překročena (produkcí)



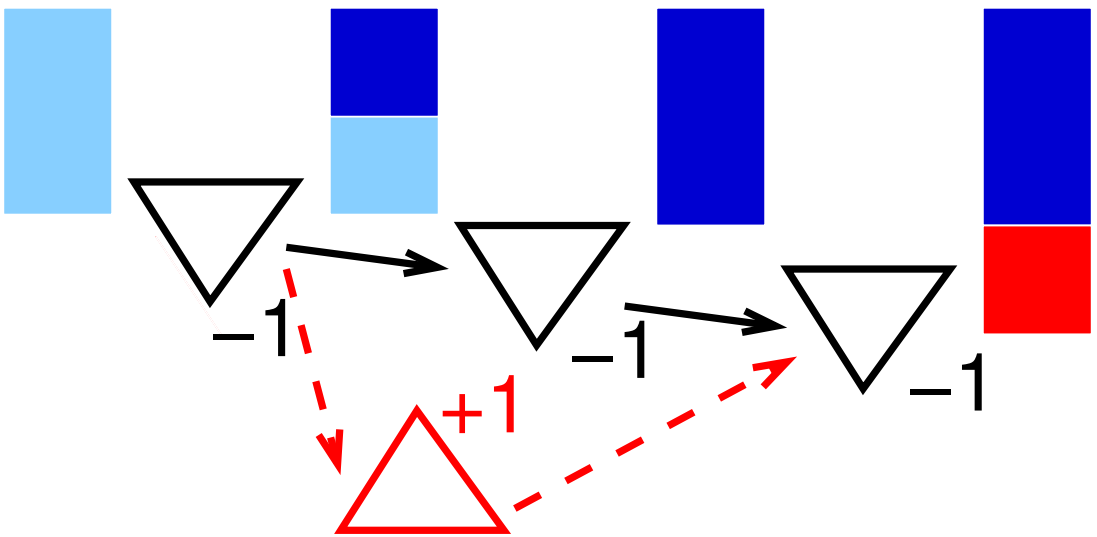
- Kumulativní zdroj může být chápan jako speciální případ rezervoáru
  - každá aktivita konzumuje cap(A) při startu
    a produkuje cap(A) na konci

Relativní uspořádání

- Pokud je čas relativní (uspořádání aktivit)

  potom techniky edge finding a agregovaných požadavků nic neodvodí
- Pořád ale můžeme používat informace o uspořádání aktivit a spotřebě/produkci daného zdroje
- Příklad:

  reservoár: aktivity konzumují a produkují zdroj



lze odvodit nezbytnost přídavné aktivity produkující jednotku zdroje

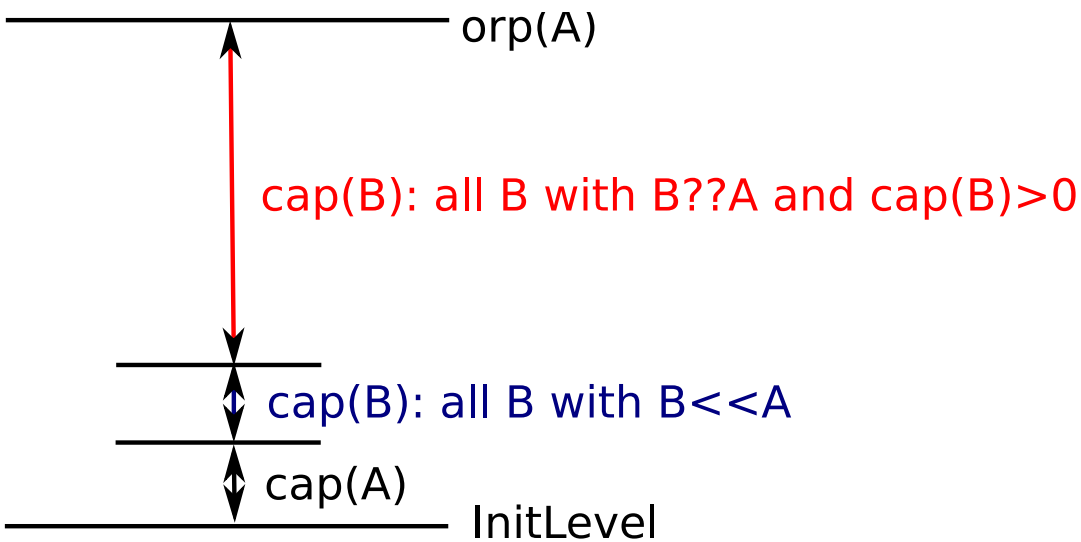
Optimistický zdrojový profil (*orp*) Cesta&Stella 1997

orp(*A*): maximální možná úroveň zdroje v čase, kdy se A začne zpracovávat

Aktivity, které musí být před A, se vezmou dohromady s produkčními aktivitami, které mohou být před A

$$orp(A) = InitLevel + cap(A) + \sum_{B<<A} cap(B) + \sum_{B??A \,\&\, cap(B)>0} cap(B)$$

Příklad:

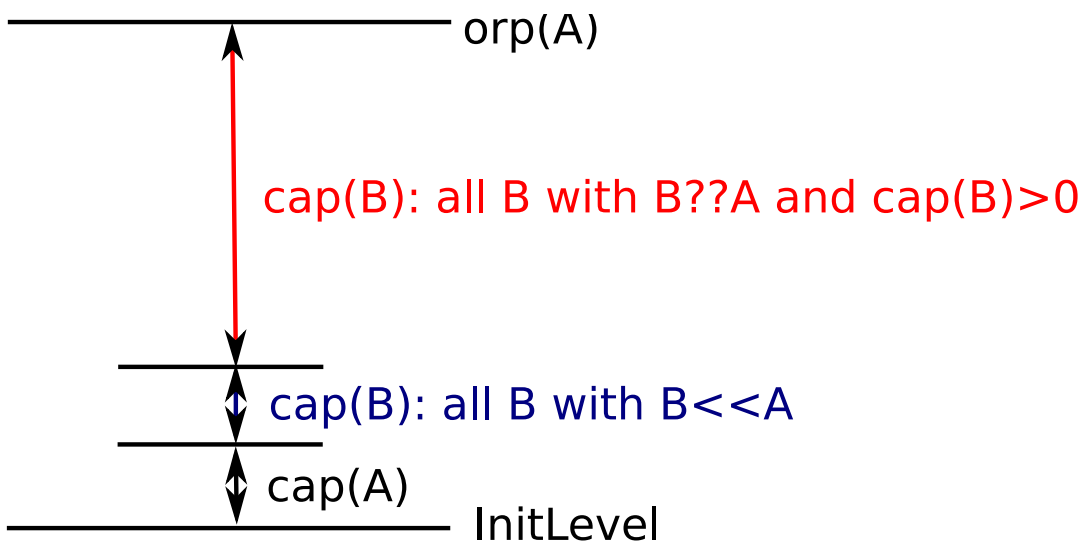


B??A znamená, že pořadí A a B ještě není známé

orp odvozovací pravidla I.

$orp(A) < MinLevel \Rightarrow fail$

- i když je veškerá produkce plánována před A
  není dosažena minimální požadovaná úroveň zdroje



$$orp(A) = InitLevel + cap(A) + \sum_{B<<A} cap(B) + \sum_{B??A \,\&\, cap(B)>0} cap(B)$$

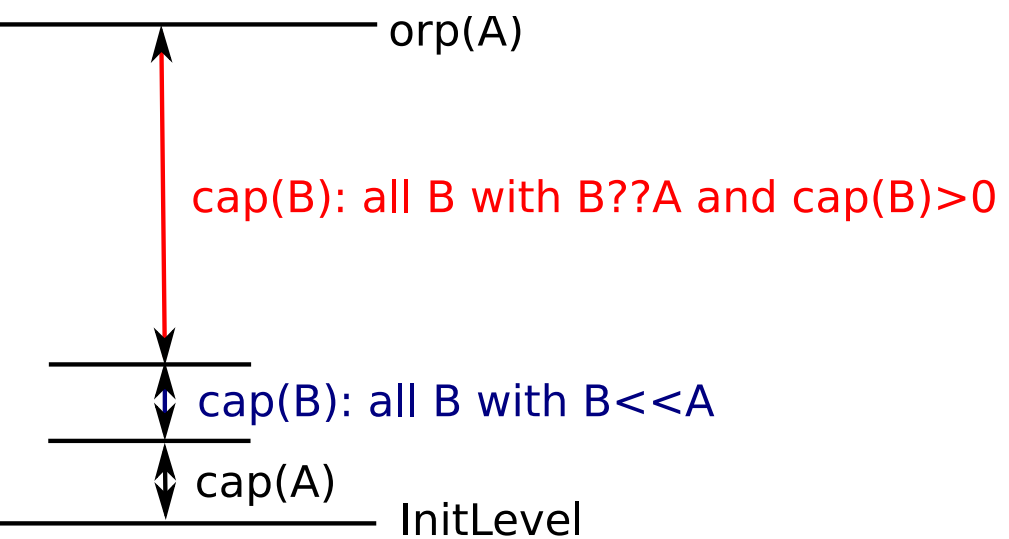
orp odvozovací pravidla II.

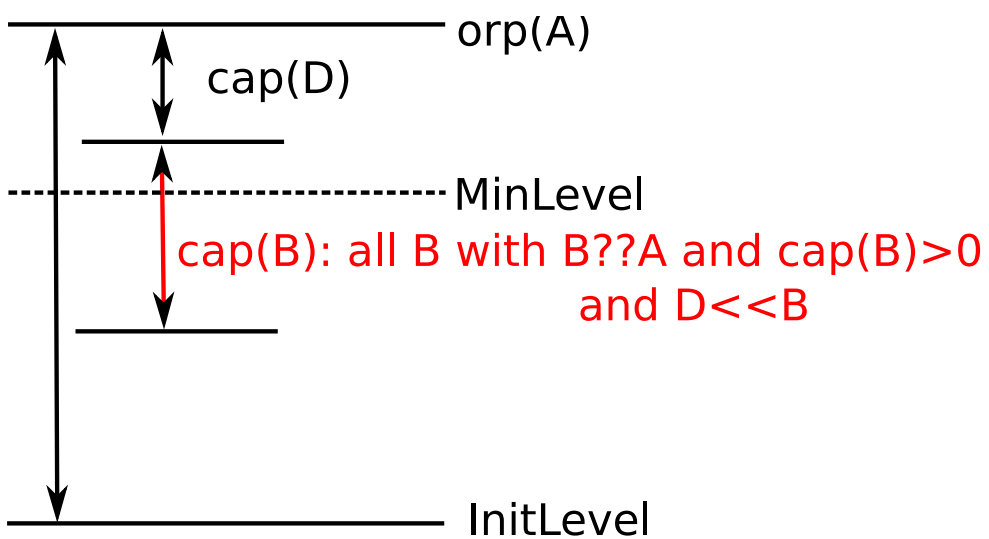
$orp(A) - cap(D) - \sum_{D<<B \,\&\, B??A \,\&\, cap(B)>0} cap(B) < MinLevel$
$\Rightarrow D << A$

Uvažujme časový okamžik, kdy začne aktivita A

- pro libovolné D takové, že $D??A$ a $cap(D) > 0$:
  pokud je produkce v D plánována za A a minimální požadovaná úroveň zdroje není dosažena, pak musí být D před A

Příklad:





Odvozovací pravidla pro *orp*

- $orp(A) = InitLevel + prod(A) + \sum_{B<<A} prod(B) + \sum_{B??A \ \&\ prod(B)>0} prod(B)$
- $orp(A) < MinLevel \Rightarrow fail$
  - přestože veškerá produkce je plánována před A, pořád ještě není dosažena požadovaná minimální úroveň zdroje
- $orp(A) - prod(D) - \sum_{D<<C \ \&\ C??A \ \&\ prod(C)>0} prod(C) < MinLevel$

  $\Rightarrow D << A$

  pro libovolné D takové, že $D??A$ a $prod(D) > 0$
  - pokud je produkce v D plánována za A a minimální požadovaná úroveň zdroje není dosažena ani když všechny ostatní produkční aktivity jsou před A (které tam být mohou), potom D musí být před A
  - tedy odečteme produkci C, které musí být až po D (tudíž tato produkce není k dispozici pro A)

  $\Rightarrow$ v orp(A) je zahrnuta produkce D a produkce aktivit C, které musí být po D. Proto tyto produkce odečítáme od orp(A) a dostáváme se k novému požadavku na MinLevel.

Optimistický zdrojový profil: cvičení

$$orp(A) = InitLevel + cap(A) + \sum_{B<<<A} cap(B) + \sum_{B??A \,\&\, cap(B)>0} cap(B)$$

$$orp(A) - cap(D) - \sum_{D<<<B \,\&\, B??A \,\&\, cap(B)>0} cap(B) < MinLevel$$
$$\Rightarrow D << A$$

- Máme dány aktivity A, B, C, X, Y, Z
  a jejich požadované kapacity jsou po řadě $1, 2, 3, 4, 5, -1$
- Jsou dány precedence: $A << B << C, X << Y$
- $InitLevel = MinLevel = 0$
- Jaká je hodnota $orp(A)$?
  $orp(A) = 0 + cap(A) + 0 + (cap(X) + cap(Y)) = 0 + 1 + 0 + (4 + 5) = 10$
- Může být X naplánováno po A? $10 - 4 - 5 = 1$, tj. ano
- Co se změní, pokud bychom přidali aktivitu D s následujícími vlastnostmi?
  - $cap(D) = -2$
  - $D << A$

  $orp(A) = 0 + 1 + (\underline{-2}) + (4 + 5) = 8$ a dále $8 - 4 - 5 = -1$, tj. X nesmí být po A

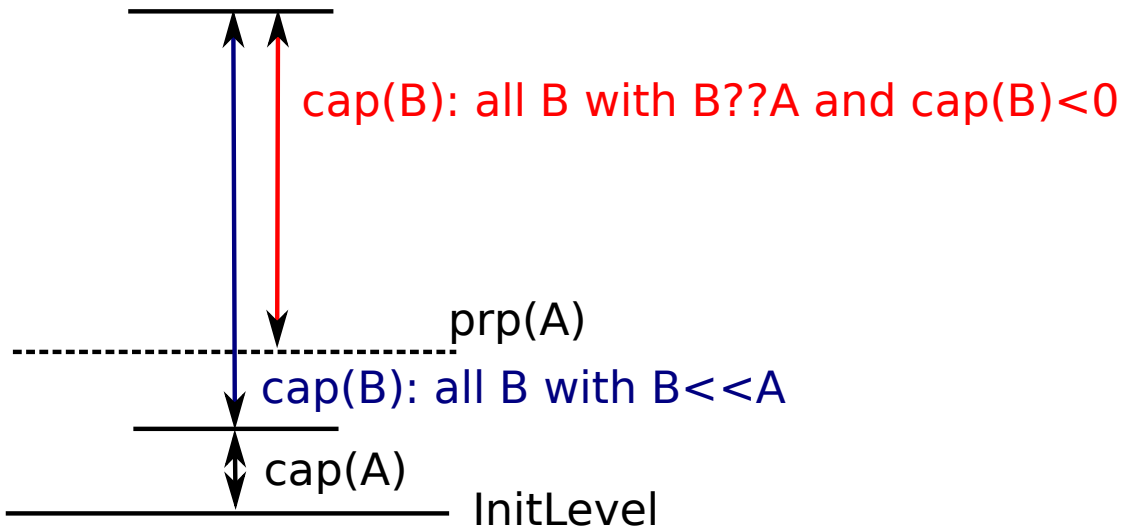
Pesimistický zdrojový profil (*prp*) Cesta&Stella 1997

prp(*A*): minimální možná úroveň zdroje v čase, kdy se A začne zpracovávat

Aktivity, které musí být před A, se vezmou dohromady s konzumačními aktivitami, které mohou být před A

$$prp(A) = InitLevel + cap(A) + \sum_{B<<A} cap(B) + \sum_{B??A \,\&\, cap(B)<0} cap(B)$$

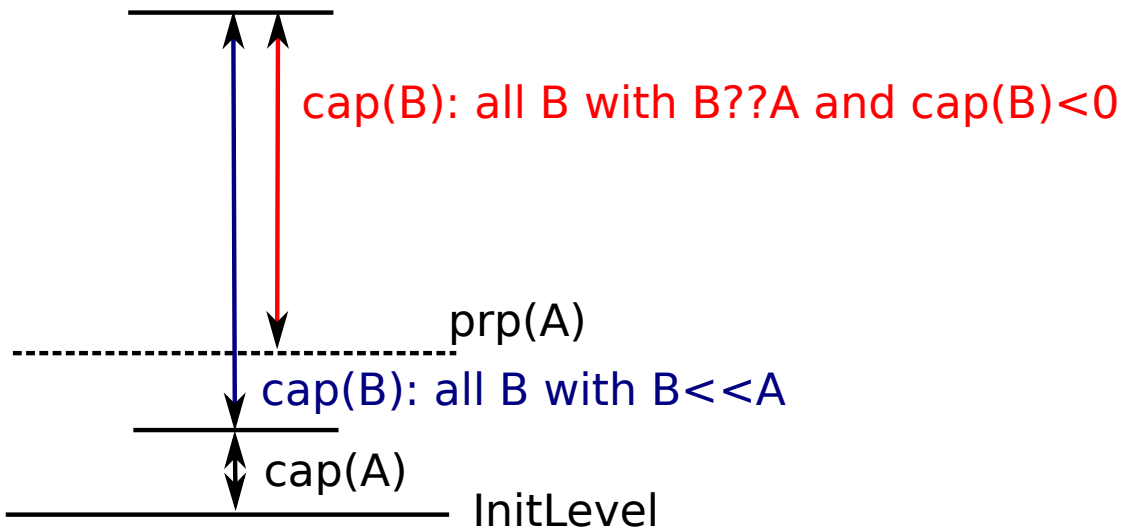
Příklad:



prp odvozovací pravidla I.

$prp(A) > MaxLevel \Rightarrow fail$

- i když je veškerá konzumace plánována před A, maximální povolená kapacita zdroje je překročena



$$prp(A) = InitLevel + cap(A) + \sum_{B<<A} cap(B) + \sum_{B??A \,\&\, cap(B)<0} cap(B)$$

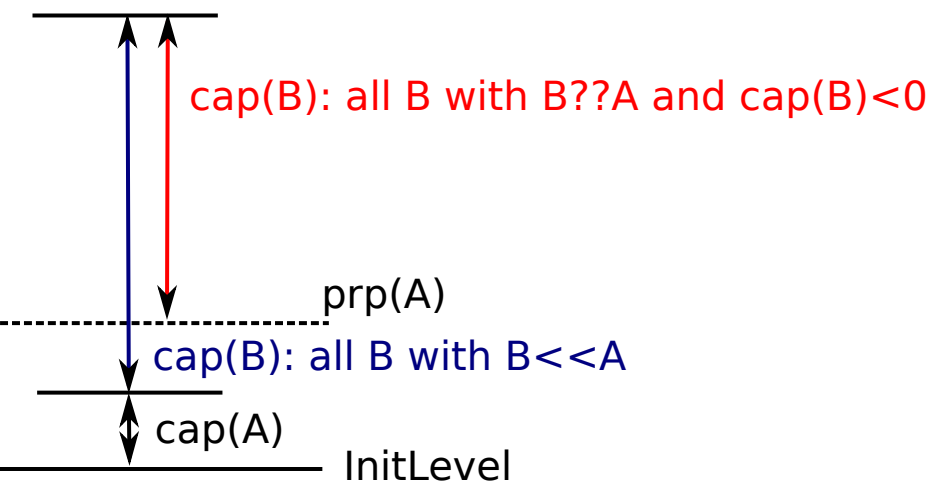
prp odvozovací pravidla II.

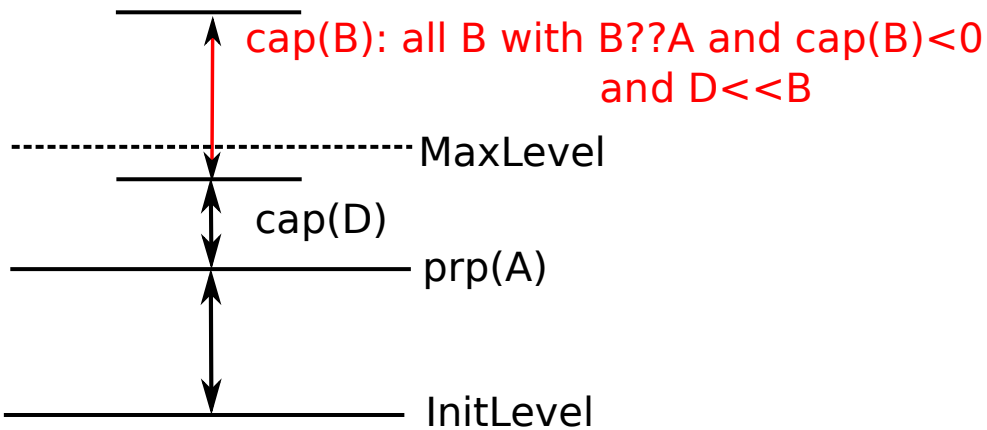
$prp(A) - cap(D) - \sum_{D<<B \;\&\; B??A \;\&\; cap(B)<0} cap(B) > MaxLevel$
$\Rightarrow D << A$

Uvažujme časový okamžik, kdy začne aktivita A:

- pro libovolné D takové, že $D??A$ a $cap(D) < 0$:
  jestliže je konzumace v D plánována po A a je překročena maximální povolená úroveň zdroje, pak musí být D před A

Příklad:





Odvozovací pravidla pro *prp*

- $prp(A) = InitLevel + prod(A) + \sum_{B<<A} prod(B) + \sum_{B??A \ \&\ prod(B)<0} prod(B)$
- $prp(A) > MaxLevel \Rightarrow fail$
  - přestože veškerá konzumace je plánována před A, je maximální úroveň zdroje (kapacita) překročena
- $prp(A) - prod(D) - \sum_{D<<C \ \&\ C??A \ \&\ prod(C)<0} prod(C) > MaxLevel$

  $\Rightarrow D << A$

  pro libovolné D takové, že $D??A$ a $prod(D) < 0$
  - pokud je konzumace v D plánována za A a maximální úroveň zdroje je překročena i když všechny ostatní konzumační aktivity jsou před A (které tam být mohou), potom D musí být před A
  - tedy přičteme konzumaci C, které musí být až po D (tudíž se tato konzumace neodehraje před A)

  $\Rightarrow$ v prp(A) je zahrnuta konzumace D a konzumace aktivit C, které musí být po D. Proto tyto konzumace přičítáme k prp(A) a dostáváme se k novému požadavku na MaxLevel.

Globální podmínky

Pro reprezentaci zdrojů využívány v programovacích jazycích
tzv. globální podmínky

- definované pro libovolný konečný počet proměnných
- komplexní podmínky s vlastním propagačním algoritmem

Základní globální podmínky (pro rozvrhování)

- příklady z IBM ILOG OPL (Optimization Programming Language)
- všechny proměnné různé
  - `allDifferent`
- disjunktivní zdroj
  - `dvar interval`, `dvar sequence`
  - `noOverlap`
- kumulativní zdroj
  - `cumuFunction`, `pulse`

Všechny proměnné různé

Proměnné v poli `Array` jsou různé

- reprezentace unárního zdroje s jednotkovou dobou trvání všech aktivit
- `dvar int Array[Interval];`
- globální podmínka: `allDifferent(Array)`

Příklad: učitelé musí učit v různé hodiny

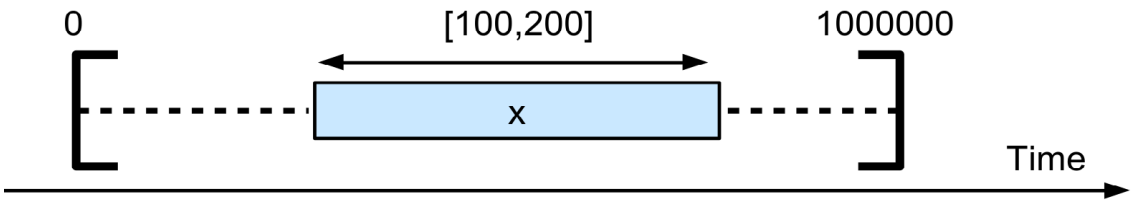
- `Jan = 6, Ota = 2, Anna = 5,`
  `Marie = 1, Petr` $\in \{3,4\}$`, Eva` $\in \{3,4\}$

učitel	min	max
Jan	3	6
Petr	3	4
Anna	2	5
Ota	2	4
Eva	3	4
Marie	1	6

Intervalové proměnné

Intervalová proměnná: pro modelování časového intervalu (úlohy, aktivity)

- hodnotou intervalové proměnné je celočíselný interval $[start, end)$
- příklad: `dvar interval x in 0..1000000 size 100..200;`



Disjunktní rozvrhování

Sekvenční proměnná `p`

- definována na množině intervalových proměnných `x`

```
dvar interval x[i in 1..n] ...;
dvar sequence p in x;
```

- hodnota intervalové proměnné `p` je permutace přítomných intervalů
  - pozor, permutace `t` ještě neimplikuje žádné uspořádání v čase

Omezení `noOverlap(p)`

- vyjadřuje, že sekvenční proměnná `p` reprezentuje řetězec nepřekrývajících se intervalových proměnných
- pro vyjádření rozvrhování na unárním/disjunktivním zdroji, kde se intervaly/úlohy nepřekrývají

Precedence, účelová funkce

Mezi intervalovými proměnnými můžeme definovat precedenční podmínky:

```
dvar interval i;
dvar interval j;


endBeforeStart(i, j);
startBeforeStart(i, j);
startAtStart(i, j);
...
```

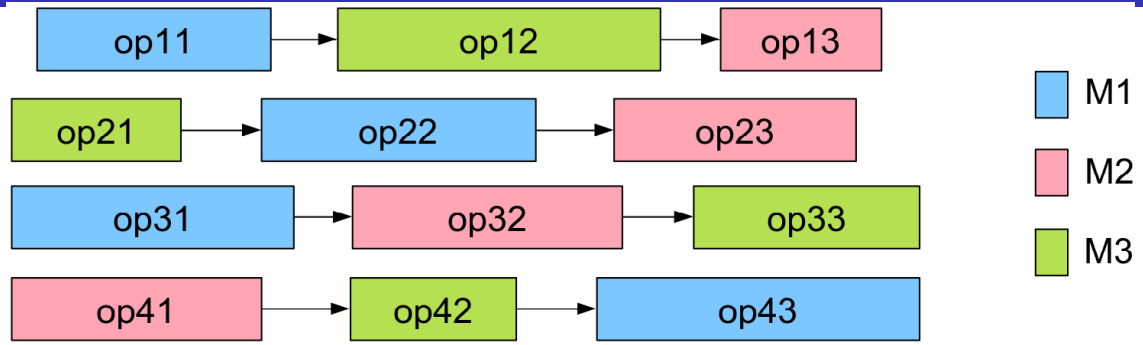
Pro vytváření účelových funkcí nebo definici omezení

```
startOf(x)
endOf(x)
sizeOf(x, V)
```

Příklad: minimalizace makespanu

```
minimize max(i in 1..n) endOf(x[i])
```

Příklad: rozvrhování problému job-shop




```
tuple Operation {
  int mch; // machine
  int pt;  // processing time
};
Operation Ops[j in Jobs][p in Pos] = ...;
```

```
1  dvar interval op[j in Jobs][p in Pos] size Ops[j][p].pt;
2  dvar sequence mchs[m in Mchs] in
3    all(j in Jobs, p in Pos: Ops[j][p].mch == m) op[j][p];
4
5  minimize max(j in Jobs) endOf(op[j][nbPos]);
6  subject to {
7    forall(m in Mchs)
8      noOverlap(mchs[m]);
9    forall(j in Jobs, p in 2..nbPos)
10     endBeforeStart(op[j][p-1],op[j][p]);
11 }
```

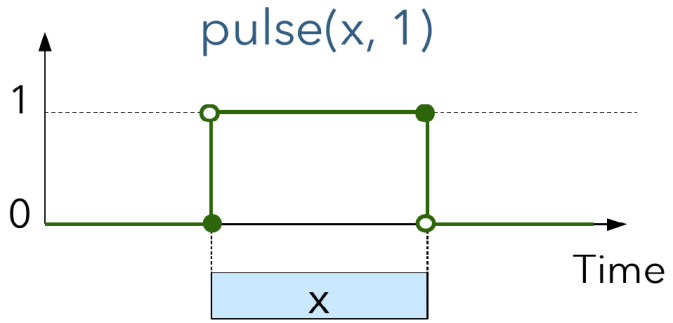
op[j][p] odkazuje operaci úlohy j, která je zpracovávána v rámci úlohy jako p-tá

Kumulativní zdroj pomocí kumulativní funkce

Hodnota výrazu kumulativní funkce reprezentuje vývoj kvantity v čase, která může být inkrementálně změněna (snížena nebo navýšena) intervalovými proměnnými.

Intervalové proměnné `x[i]` přispívají do kumul. funkce po dobu svého provádění

```
int capacity[1..5] = [1,3,2,4,1];
cumulFunction y = sum(i in 1..5) pulse(x[i],capacity[i]);
```



Omezení na výrazech kumulativní funkce: pro omezení kapacity zdroje

```
int h = ...
cumulFunction f= ...
f<=h
```

Příklad: job-shop a omezení celkového počtu strojů

```
cumulFunction allMachines = sum(j in Jobs, p in Pos) pulse(op[j][p],1);
allMachines <= m;
```

Prohledávání/přiřazování (opakování)

Konzistenční techniky jsou (obvykle) neúplné

$\Rightarrow$ potřeba prohledávací algoritmus, který vyřeší "zbytek"

Přiřazovaní *(labeling)*

- prohledávání do hloubky (DFS/BT)
  - přiřaď hodnotu do proměnné
  - propaguj = uděl̆ej
    problém lokálně konzistentní
  - vrať se v případě neúspěchu

Způsoby větvení (opakování)

Jaká proměnná má být ohodnocena první?

- princip prvotního neúspěchu *(first-fail)*
  - preferuj proměnnou, jejíž přiřazení je nejobtížnější (hrozí u ní nebezpečí failu)

Jaká hodnota má být vyzkoušena první?

- princip prvotního úspěchu *(succeed-first)*
  - preferuj hodnoty, které nejspíše patří do řešení

Schémata větvení

Větvení = řešení disjunkcí

Tradiční rozvrhovací přístupy

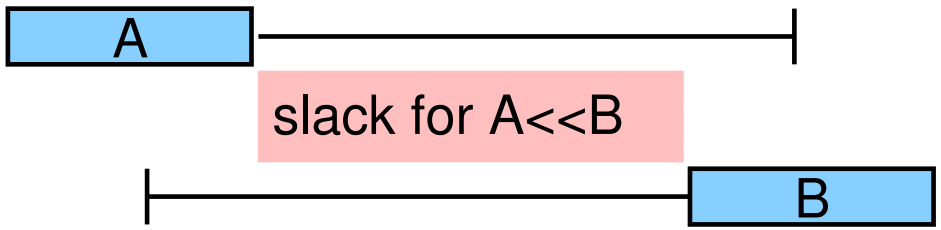
- kritická rozhodnutí se dělají první
  - vyřeš kritická místa *(bottlenecks)*, ...
  - definuje tvar prohledávacího stromu
  - podobně jako princip prvního neúspěchu *(first-fail)*
- preferuj alternativy s větší flexibilitou
  - definuje pořadí větví pro prozkoumání
  - podobně jako princip prvního úspěchu *(succeed-first)*

Jak popsat, co je kritické a co je flexibilní?

Rezerva *(slack)* Smith&Cheng 1993

- Rezerva *(slack)* je formální popis flexibility
- Rezerva pro dané pořadí dvou aktivit

  „volný čas pro posunování aktivit"



$$slack(A << B) = \max(end(B)) - min(start(A)) - p(A) - p(B))$$

- Rezerva pro dvě aktivity (bez určení pořadí)

  $slack(\{A, B\}) = \max(slack(A << B), slack(B << A))$

- Rezerva pro skupinu aktivit

  $slack(\Omega) = \max(end(\Omega)) - min(start(\Omega)) - p(\Omega)$

Větvení uspořadáním dvojic aktivit

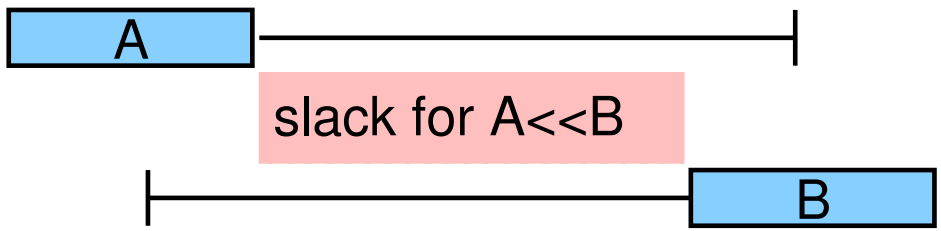
$A << B \quad \vee \quad \neg A << B$

Jaké aktivity mají být uspořádány první?

- nejkritičtější pár (first-fail)
- pár s minimální rezervou $slack(\{A, B\})$

Jaké pořadí aktivit má být zvoleno?

- nejflexibilnější pořadí (succeed-first)
- pořadí maximalizující $slack(A??B)$
- Příklad: vybereme pořadí $A << B$



Bodů volby při n aktivitách: $O(n^2)$

Větvení první/poslední — Baptiste at al. 1995

$$(A << \Omega \vee \neg A << \Omega) \quad \vee \quad (\Omega << A \vee \neg \Omega << A)$$

Máme hledat první nebo poslední aktivitu?

- díváme se na množinu možných kandidátů na první aktivitu a na množinu možných kandidátu na poslední aktivitu
- výbereme menší z těchto dvou množin (first-fail)
  - menší počet kandidátů znamená, že je těžší najít vhodného kandidáta

Jaká aktivita má být vybrána?

- pokud se hledá první aktivita, potom preferuj aktivitu, která má nejmenší $min(start(A))$
- pokud se hledá poslední aktivita, potom preferuj aktivitu, která má největší $max(end(A))$

Bodů volby: $O(n)$

Zdrojová rezerva

Zdrojová rezerva je definovaná jako

rezerva množiny aktivit zpracovávaných daným zdrojem

Jak používat zdrojovou rezervu?

- pokud volíme zdroj, na kterém budou aktivity uspořádány jako první
  - vyber zdroj s minimální rezervou (kritické místo)
- pokud volíme zdroj, na který alokovat danou aktivitu
  - vyber zdroj s maximální rezervou (flexibilita)

Omezující podmínky: shrnutí

Problém splňování podmínek

- popis problému pomocí doménových proměnných a omezení
- konzistence a propagace
- prohledávání

Rozvrhování jako problém splňování podmínek

- doménové proměnné pro čas a zdroje
- propagační algoritmy pro
  - unární zdroje
  - alternativní zdroje
  - kumulativní zdroje
  - produkovatelné a spotřebovatelné zdroje
- globálních podmínky pro zdroje
- prohledávání a rozvrhovací strategie
  - pojem rezervy
  - přístupy k větvení