Plánování a rozvrhování

8

Roman Barták, KTIML

roman.bartak@mff.cuni.cz
http://ktiml.mff.cuni.cz/~bartak

Plánování s časem

Přístupy

- **Plánování s časovými operátory**
  - Při popisu akce říkáme, kdy mají platit předpoklady, kdy nastávají efekty a jaký je vztah mezi příslušnými časovými proměnnými.
- **Plánování s kronikami**
  - Akce je částečným popisem funkce charakterizující stavové proměnné v čase.
- **Plánovací graf s časem**
  - Akce je rozdělena na tři instance – start, střed, konec – a stavové vrstvy mají dobu trvání.

O co půjde?

Plánování, kde čas hraje důležitější roli.

Akce nejsou jen jednoduché změny stavu, ale **sady lokálních změn parametrů a trvajících podmínek**, které jsou rozprostřeny v čase:

- akce zabírají jistý **časový úsek**
  - cesta z A do B trvá určitý čas
- **předpoklady** akce platí v daných časových úsecích trvání akce
  - místo B musí být volné těsně před příjezdem
- podobně **efekty** akce nastávají v určitém čase trvání akce
  - místo A se uvolní okamžitě po odjezdu
- akce spolu **interferují** a mohou tak dosahovat **spojených efektů**
  - k otevření dveří je potřeba stisknout kliku a zatlačit na dveře
- **cíle** i **známé stavy** mohou být rozprostřeny v čase
  - v daném čase se dok opravuje a tedy je pro lodě nedostupný
  - nejprve je potřeba obsloužit zákazníka A potom B

Stavová proměnná a čas

- Stavové proměnné popisují vlastnost nějakého objektu v závislosti na stavu formou funkce.
  - rloc: robots $\times$ S $\rightarrow$ locations
- Nyní bude **stavová proměnná** funkce **závislá na čase**
  - rloc: robots $\times$ time $\rightarrow$ locations

Příklad:

- V čase t_1 přijel robot r1 na místo loc1, kde setrval do času t_2 a potom odjel.
- V čase t_3, $t_2<t_3$, robot r1 přijel na místo loc2, kde setrval do času t_4 a potom odjel.
- V čase t_5, $t_4<t_5$, robot r1 přijel na dosud nespecifikované místo l.

Události a trvající okolnosti

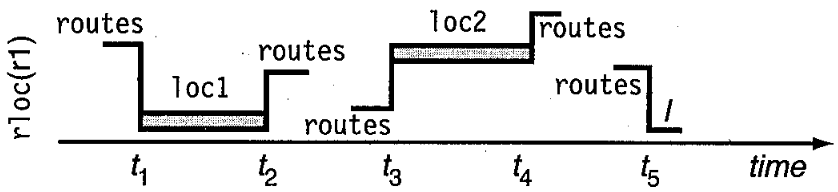
- Funkce stavové proměnné může být v plánu popsána částečně s „dírami", kde hodnotu stavové proměnné nepotřebujeme znát.
  - V průběhu plánování budeme tyto funkce postupně zpřesňovat.
- Jedná se o **po částech konstantní funkci**, kterou můžeme popsat pomocí dvou typů tzv. **časových výroků**:
  - **událost $x@t:(v_1,v_2)$** specifikující okamžitou změnu hodnoty proměnné x z v_1 na v_2 ($v_1 \neq v_2$) v čase t
    - $x@t:(v_1,v_2) \equiv (\exists t_0\ \forall t'\ (t_0<t'<t)\ x(t')=v_1) \wedge\ x(t)=v_2$
  - **trvající okolnost $x@[t_1,t_2):u$** specifikující, že proměnná x setrvává na hodnotě u v době $[t_1,t_2)$
    - $x@[t_1,t_2):u \equiv \forall t\ (t_1 \leq t<t_2)\ x(t)=u$

Vzájemný vztah události a trvající okolnosti lze popsat takto:

$x@t:(v_1,v_2) \equiv\ v_1 \neq v_2 \wedge \exists t_1, t_2\ (t_1<t<t_2)\ x@[t_1,t):v_1 \wedge x@[t,t_2):v_2$

Kronika

- **Kronika** pro množinu stavových proměnných je dvojice Φ=(F,C), kde
  - F je množina **časových výroků** nad stavovými proměnnými (tj. události a trvající okolnosti)
  - C je množina:
    - **objektových podmínek**, tj. podmínek svazujících objektové proměnné ve tvaru $x \in D$, $x=y$, $x \neq y$ a vztahy určené neměnnými relacemi
    - **časových podmínek**, tj. podmínek nad časovými proměnnými podle algebry okamžiků (<,=,>)
- **Časová osa** je kronika pro jednu stavovou proměnnou.



({ rloc(r1)@t_1: (l_1,loc1),
rloc(r1)@[t_1,t_2) : loc1,
rloc(r1)@t_2: (loc1,l_2),
rloc(r1)@t_3: (l_3,loc2),
rloc(r1)@[t_3,t_4) : loc2,
rloc(r1)@t_4: (loc2,l_4),
rloc(r1)@t_5: (l_5,l) }
{ adjacent(l_1,loc1),
adjacent(loc1,l_2),
adjacent(l_3,loc2),
adjacent(loc2,l_4),
adjacent(l_5,l),
$t_1<t_2<t_3<t_4<t_5$ })

Separace

- Aby **časová osa** specifikovala **alespoň jednu funkci** určující hodnotu **stavové proměnné** nesmí v ní být žádná dvojice **konfliktních časových výroků**, tj. výroků, které umožňují dvě různé hodnoty stavové proměnné ve stejném čase.
- Konfliktním výrokům se vyhneme, pokud pro ně časová osa obsahuje, přímo či nepřímo, nějakou separační podmínku.
- Pro dvojici časových výroků definujeme **separační podmínku** takto:
  - pro $x@[t_1,t_2):v_1$ a $x@[t_3,t_4):v_2$ máme tři možné separační podmínky:
    - $t_2 \leq t_3$, $t_4 \leq t_1$, $v_1=v_2$
  - pro $x@t:(v_1,v_2)$ a $x@[t_1,t_2):v$ máme čtyři možné separační podmínky:
    - $t<t_1$, $t_2<t$, $(t_1=t \wedge v=v_2)$, $(t_2=t \wedge v=v_1)$
  - pro $x@t:(v_1,v_2)$ a $x@t':(v'_1,v'_2)$ máme dvě možné separační podmínky:
    - $t \neq t'$, $(v_1=v'_1 \wedge v_2=v'_2)$

Poznámka:

- Separaci lze provést také požadavkem na různost objektové proměnné v popisu časových výroků (např. dvě události pro rloc(r) a rloc(r') lze separovat podmínkou $r \neq r'$).

Konzistence

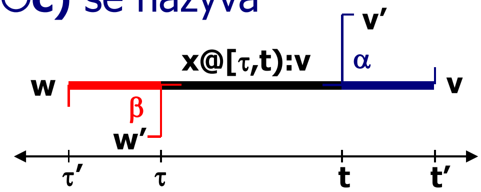
- **Časová osa** $\Phi=(F,C)$ pro proměnnou x je **konzistentní**, právě když C je konzistentní a pro každou dvojici časových výroků z F existuje separační podmínka, která je důsledkem C.
  - separační podmínka je buď přímo součástí C
  - nebo je přímým důsledkem C (tj. je splněna v každém řešení C)
- **Kronika je konzistentní**, právě když jsou všechny její časové osy konzistentní.

Poznámka:

- Pojem konzistence vyžaduje, aby separační podmínky byly důsledkem C, nestačí možnost jejich bezkonfliktního přidání do C.

Podpora výroku

- Konzistentní **kronika** $\Phi=(F,C)$ **podporuje časový výrok** α tvaru **x@t:(v,v′)** nebo **x@[t,t′):v**, právě když je v F výrok β nastavující hodnotu v pro α. β je buď tvaru **x@τ:(w′,w)** nebo **x@[τ′,τ):w** a existují separační podmínky c tak, že $\Phi \cup (\{\alpha, x@[\tau,t):v\}, \{w=v, \tau<t\} \cup c)$ je konzistentní kronika.
  - $\Phi \cup \Phi' = (F \cup F', C \cup C')$, $\Phi \subseteq \Phi' \equiv (F \subseteq F' \wedge C \subseteq C')$,
  - β se nazývá **podporou** pro α
  - dvojice δ = **($\{\alpha$, x@[τ,t):v$\}$, $\{$w=v,τ<t$\} \cup$c)** se nazývá **pomocníkem** (enabler) pro α v Φ



- **Poznámky:**
  - Kronika musí být konzistentní než může někoho podporovat.
  - Pomocník je také kronikou.
  - Podporu pro α hledáme pouze zpětně, tj. pro čas před t. Tato podpora se použije jako kauzální zdůvodnění α.
  - Podpor a pomocníků pro α může být více.

Podpora kroniky

Konzistentní **kronika** Φ=(F,C) **podporuje množinu časových výroků** ε, právě když je každý výrok $\alpha_i \in \varepsilon$ podporován kronikou ($F \cup \varepsilon - \{\alpha_i\}$, C) pomocníkem δ_i tak, že $\Phi \cup \phi$ je konzistentní kronika ($\phi = \cup_i \delta_i$).

Poznámky:

- Definice umožňuje aby jeden výrok $\alpha_i \in \varepsilon$ podporoval jiný výrok $\alpha_j \in \varepsilon$ vzhledem k Φ, za předpokladu že sjednocení pomocníků je konzistentní s Φ. To vede k možnosti překryvu akcí s **interferujícími efekty**.
- ϕ je pomocník pro ε (a nemusí být určen jednoznačně)

Nechť Φ'=(F',C') je kronika taková, že Φ podporuje F' a nechť **$\theta(\Phi'/\Phi) = \{\phi \cup (\emptyset, C') \mid \phi$ je pomocník pro F'$\}$**.
Potom konzistentní **kronika** Φ=(F,C) **podporuje kroniku** Φ'=(F',C'), právě když Φ podporuje F' a existuje pomocník $\phi \in \theta(\Phi'/\Phi)$ tak, že $\Phi \cup \phi$ je konzistentní kronika.

Φ' **je důsledkem** Φ, právě když Φ podporuje Φ' a existuje pomocník $\phi \in \theta(\Phi'/\Phi)$ tak, že $\phi \subseteq \Phi$.

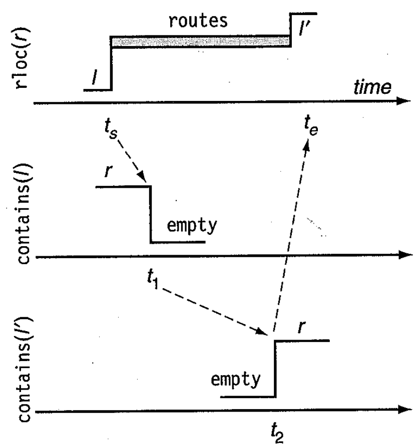
Plánovací operátor

Plánovací operátor je dvojice o = (name(o), (F(o),C(o))):

- name(o) je výraz ve tvaru o(t_s,t_e,t_1,…,v_1,v_2,…) obsahující všechny časové a objektové proměnné operátoru (o je symbol operátoru)
- (F(o),C(o)) je kronika

Příklad *(zjednodušený zápis)*:

move(t_s,t_e,t_1,t_2,r,l,l') =
{rloc(r)@t_s : (l,routes),
rloc(r)@[t_s,t_e) : routes,
rloc(r)@t_e : (routes,l'),
contains(l)@t_1 : (r,empty),
contains(l')@t_2 : (empty,r),
$t_s < t_1 < t_2 < t_e$,
adjacent(l,l') }



Rozdíl od klasických plánovacích operátorů

- **nejsou** nijak odlišeny **předpoklady a efekty**
- **operátor** neaplikujeme na stav ale **na kroniku**
- výsledek **aplikace** operátoru **není jednoznačný**

Aplikace akce

- **Akce** je částečně instanciovaný operátor.
- **Akce** a=(F(a),C(a)) je **aplikovatelná** na kroniku Φ, právě když Φ podporuje kroniku (F(a),C(a)).

  Výsledkem aplikace akce a na Φ je množina kronik $\gamma(\Phi,a) = \{\Phi\cup\phi \mid \phi\in\theta(a/\Phi)\}$.
- **Množina akcí** $\pi=\{a_1,\ldots,a_n\}$ je **aplikovatelná** na kroniku Φ, právě když Φ podporuje kroniku $\Phi_\pi = \cup_i (F(a_i),C(a_i))$.

  Výsledkem aplikace π na Φ je množina kronik $\gamma(\Phi,\pi) = \{\Phi\cup\phi \mid \phi\in\theta(\Phi_\pi/\Phi)\}$.

Plánovací problém

- **Plánovací problém** je trojice $P=(O,\Phi_0,\Phi_g)$, kde
  - O je množina operátorů
  - Φ_0 je konzistentní kronika popisující neměnné relace, počáteční stav a očekávaný vývoj nezávislý na plánovaných akcích
  - Φ_g je konzistentní kronika reprezentující cíle
- **Řešící plán** pro problém P je množina akcí $\pi=\{a_1,\ldots,a_n\}$, kde každá akce je instancí operátoru z O a Φ_g je důsledkem nějaké kroniky z $\gamma(\Phi_0,\pi)$.

Plánování s kronikami

- Plánovací procedura je odvozena od **plánování v prostoru plánů**.
- Pro plánovací problém $P=(O,\Phi_0,\Phi_g)$ začneme s kronikou $\Phi=(F_0,C_0\cup C_g)$, množinou otevřených cílů $G=F_g$, prázdným plánem $\pi=\emptyset$ a prázdnou množinou hrozeb $K=\emptyset$.

```
CP(Φ, G, K, π)
    if G = K = ∅ then return(π)
    perform the two following steps in any order
        if G ≠ ∅ then do
            select any α ∈ G
            if θ(α/Φ) ≠ ∅ then return(CP(Φ, G − {α}, K ∪ {θ(α/Φ)}, π))
            else do
                relevant ← {a | a contains a support for α}
                if relevant = ∅ then return(failure)
                nondeterministically choose a ∈ relevant
                return(CP(Φ ∪ (F(a), C(a)), G ∪ F(a), K ∪ {θ(a/Φ)}, π ∪ {a}))
        if K ≠ ∅ then do
            select any C ∈ K
            threat-resolvers ← {φ ∈ C | φ consistent with Φ}
            if threat-resolvers = ∅ then return(failure)
            nondeterministically choose φ ∈ threat-resolvers
            return(CP(Φ ∪ φ, G, K − C, π))
end
```

Otevřený cíl

- je buď podporován Φ, potom jeho pomocníky zařadíme mezi hrozby K
- nebo pro něj najdeme akci, která ho podporuje, a tu zařadíme do systému

Hrozby jsou dosud nevyřízené množiny pomocníků.

- Z každé množiny pomocníků jednoho vybereme tak, aby byl konzistentní s Φ a přidáme ho do Φ.

Plánování se zdroji

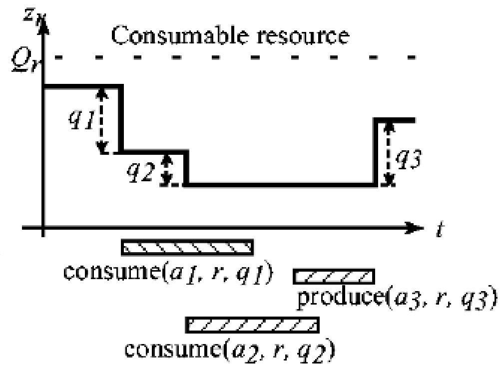
Na úvod

- Umíme používat **čas v plánování**
  - plánování s kronikami
- **Zdroje** jsme v plánování také měli
  - např. ruka, jeřáb
- **Stavová proměnná** s hodnotami obsazený/volný nemusí být efektivním modelem více identických zdrojů – je jedno, jaký jeřáb daný kontejner zvedne (symetrie).
- Množinu identických zdrojů můžeme modelovat jednou proměnnou udávající **počet dostupných zdrojů**.
  - proměnná má **numerickou doménu** (počet zdrojů)
  - proměnná se **mění relativně** (zdroje se půjčují a vrací)

Kapacitní proměnná

- **Stavová proměnná** popisuje vlastnost objektu v závislosti na čase.
  - změny jsou **absolutní** (např. pozice se změnila z loc1 na loc2)
- Podobně ale můžeme popisovat i profil zdroje, tj. změnu jeho kapacity v čase, pomocí **kapacitní proměnné**
  - resources × time → {0,1,…,Q}, kde Q je maximální kapacita zdrojů
  - obor hodnot je numerický
  - změny této proměnné budou **relativní** (kapacita se změní o danou hodnotu)

  Poznámka:
  předpokládáme **okamžité změny**



Časové výroky a kapacita

- Podobně jako změnu stavové proměnné můžeme popisovat změnu kapacitní proměnné pomocí **časových výroků pro zdroje**.
  - **pokles** kapacity z@t:-q
  - **navýšení** kapacity z@t:+q
  - **půjčení** kapacity z@[t,t′):q

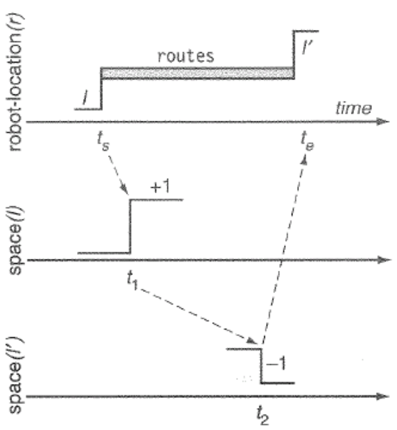
Poznámky:

- jedná se o popis **relativních** změn
- z@[t,t′):q ≡ z@t:-q ∧ z@t′:+q
- z@t:-q ≡ z@[t,∞):q
- z@t:+q ≡ z@0:+q ∧ z@[0,t):q
  - na úvod navýšíme kapacitu z Q na Q+q a až do času t si ji půjčíme
- v popisu domény je potřeba specifikovat maximální kapacitu každého zdroje

Operátory a zdroje

- **Plánovací operátor** je popsaný kronikou, kronika je množina časových výroků a podmínek.
- Pro popis práce se zdroji stačí do **kroniky** přidat odpovídající **časové výroky pro zdroje**.

$$\text{move}(t_s, t_e, t_1, t_2, r, l, l') = \{ \; \begin{aligned} &\text{robot-location}(r)@t_s : (l, \text{routes}),\\ &\text{robot-location}(r)@[t_s, t_e) : \text{routes},\\ &\text{robot-location}(r)@t_e : (\text{routes}, l'),\\ &\text{space}(l)@t_1 : +1,\\ &\text{space}(l')@t_2 : -1,\\ &t_s < t_1 < t_2 < t_e,\\ &\text{adjacent}(l, l') \; \} \end{aligned}$$



Zdrojový konflikt

- Budeme pracovat s akcemi, které si půjčují kapacitu, tj. časové výroky mají tvar z@[t,t′):q.
- Musíme rozšířit pojem konzistence kroniky na časové výroky pro zdroje, tj. vzít v úvahu kapacitu zdroje.
- Říkáme, že **množina časových výroků** R_z pro zdroj z je **konfliktní**, právě když existuje její podmnožina $\{z@[t_i,t_i'):q_i \mid i\in I\} \subseteq R_z$ taková, že:
  - výroky z této množiny se mohou překrývat, tj. je možné přiřadit časy t_i tž. $\cap_{i\in I} [t_i,t_i')\neq\emptyset$
  - $\Sigma_{i\in I}\, q_i > Q$

Poznámky:

- Konflikt znamená možné **překročení kapacity zdroje**.
- Konflikt může vzniknout pouze pro časové výroky pro stejnou kapacitní proměnnou.

- **Kronika je konzistentní**, právě když všechny časové výroky nad stavovými proměnnými jsou konzistentní a žádná množina časových výroků nad kapacitními proměnnými není konfliktní.

Kritická množina

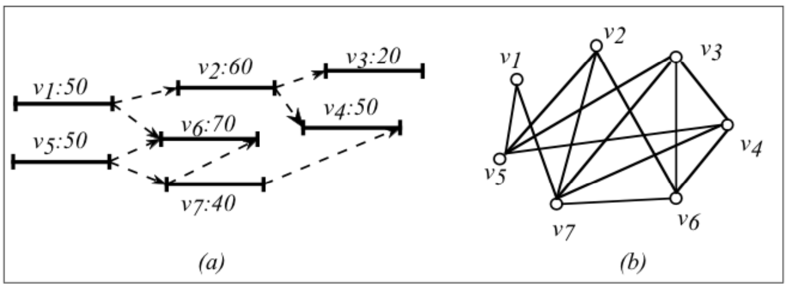
Jak hledat zdrojové konflikty?

Tvrzení:

Intervaly z množiny I se mohou překrývat, právě když se mohou překrývat po dvojicích.
$(\cap_{i\in I} [t_i,t_i')\neq\emptyset \Leftrightarrow \forall i,j\in I: [t_i,t_i')\cap[t_j,t_j')\neq\emptyset)$

Množinu intervalů/časových výroků můžeme **reprezentovat grafem**:

- vrcholy odpovídají intervalům/výrokům
- hrana spojuje vrcholy, jejichž odpovídající intervaly se mohou protínat



- V grafu budeme hledat kliku U takovou, že $\Sigma_{i\in U}\, q_i > Q$. Přesněji budeme hledat nejmenší takové kliky vzhledem k inkluzi – **minimální kritické množiny** (MCS)

Detekce konfliktů

Jak hledat minimální kritické množiny?

- vrcholy očíslujeme
- pro každý vrchol prohledáváním do hloubky konstruujeme kliky obsahující vrcholy s menším indexem
- kliky, které překročí kapacitu, si zapamatujeme a dále nerozšiřujeme

dosud nalezená část kliky

kandidáti na zařazení do kliky (mají hranu do každého vrcholu z p)

abychom nehledali stejné kliky

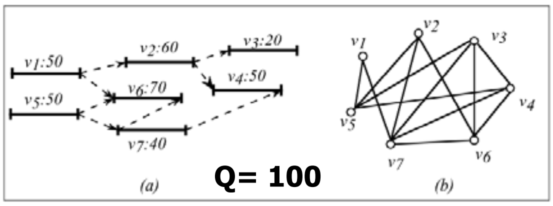
```
MCS-expand(p)
    for each v_i ∈ pending(p) do
        add a new node m_i successor of p
        pending(m_i) ← {v_j ∈ pending(p) | j < i and (v_i, v_j) ∈ E}
        clique(m_i) ← clique(p) ∪ {v_i}
        if clique(m_i) is over-consuming then MCS ← MCS ∪ clique(m_i)
        else if pending(m_i) ≠ ∅ then MCS-expand(m_i)
end
```

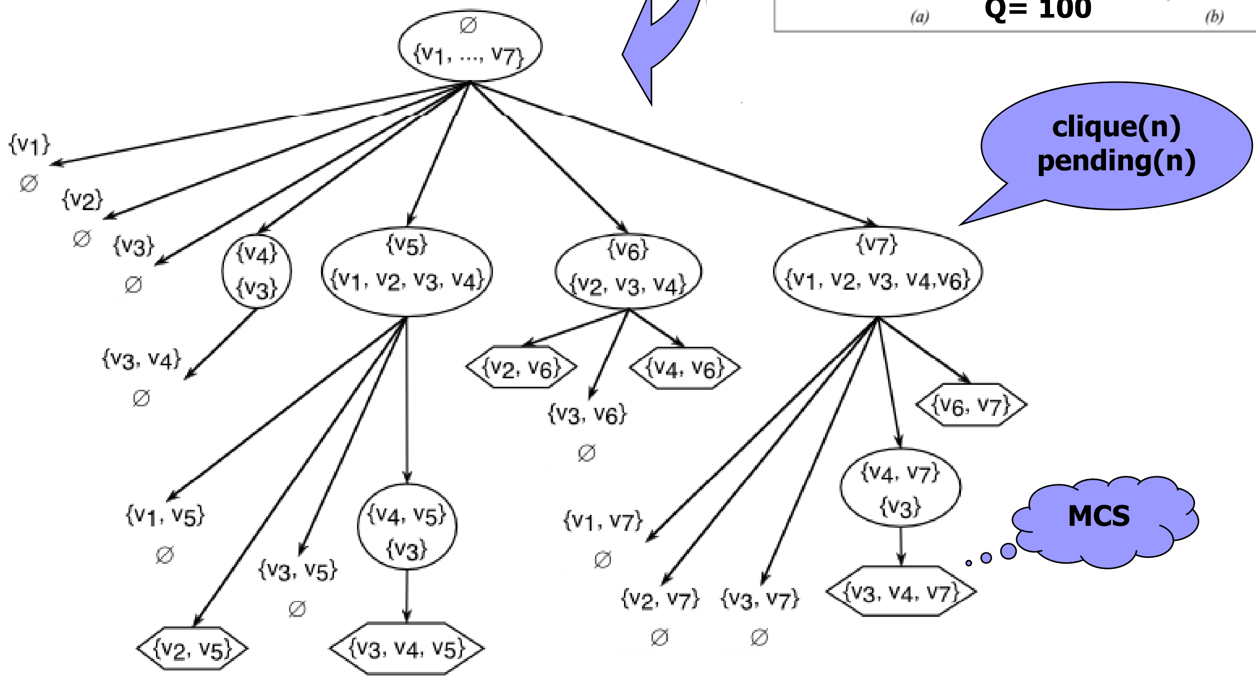
- Vstupem algoritmu je nalezená část kliky (clique), na začátku prázdná, a kandidáti na zařazení do kliky (pending), na začátku všechny vrcholy.
- Hledají se možná rozšíření kliky o uzel v_i (a uzly s indexem <i).

Detekce konfliktů

příklad

```
MCS-expand(p)
    for each v_i ∈ pending(p) do
        add a new node m_i successor of p
        pending(m_i) ← {v_j ∈ pending(p) | j < i and (v_i, v_j) ∈ E}
        clique(m_i) ← clique(p) ∪ {v_i}
        if clique(m_i) is over-consuming than MCS ← MCS ∪ clique(m_i)
        else if pending(m_i) ≠ ∅ than MCS-expand(m_i)
end
```

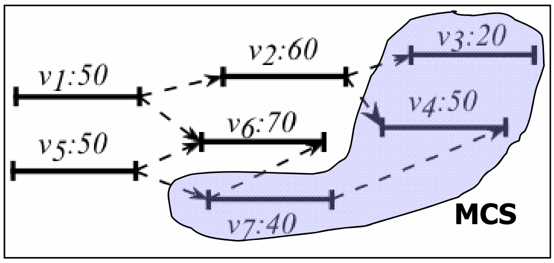




Oprava konfliktů

Jak odstranit zdrojový konflikt?

- Máme-li minimální kritickou množinu $U = \{z@[t_i,t_i'):q_i \mid i \in I\}$, potom libovolná podmínka $t_i' < t_j$ pro $i,j \in I$ odstraňuje zdrojový konflikt.
  - tato podmínka vyřadí z grafu hranu (i,j), tj. U přestane být klikou
  - každá klika $U' \supseteq U$ přestane být klikou
  - žádná klika $U' \subseteq U$ nebyla konfliktní
- Některé z těchto **podmínek** jsou **v konfliktu** s ostatními podmínkami.
  - Příklad: $t_4'<t_7$ je v konfliktu s $t_7'<t_4'$ a $t_7<t_7'$
  - Tyto opravy nelze použít!
- Některé podmínky jsou **příliš silné** (vynutí odstranění další hrany z kliky).
  - Příklad: $t_4'<t_3$ je příliš silná, protože vynucuje $t_7'<t_3$ (přes $t_7'<t_4'$)
  - Protože plánovací algoritmus bude větvit podle různých oprav, je dobré použít jen nejnutnější opravy (ostatní opravy jsou jejich důsledkem)



Plánování a zdroje

```
CPR(Φ, G, K, M, π)
  if G = K = M = ∅ then return(π)
  perform the three following steps in any order
    if G ≠ ∅ then do
      select any α ∈ G
      if θ(α/Φ) ≠ ∅ then return(CPR(Φ, G − {α}, K ∪ θ(α/Φ), M, π))
      else do
        relevant ← {a | a applicable to Φ and has a provider for α}
        if relevant = ∅ then return(failure)
        nondeterministically choose a ∈ relevant
        M' ← the update of M with respect to Φ ∪ (F(a), C(a))
        return(CPR(Φ∪(F(a),C(a)), G∪F(a), K∪{θ(a/Φ)}, M', π∪{a}))
    if K ≠ ∅ then do
      select any C ∈ K
      threat-resolvers ← {φ ∈ C | φ consistent with Φ}
      if threat-resolvers = ∅ then return(failure)
      nondeterministically choose φ ∈ threat-resolvers
      return(CPR(Φ ∪ φ, G, K − C, M, π))
    if M ≠ ∅ then do
      select U ∈ M
      resource-resolvers ← {φ resolver of U | φ is consistent with Φ}
      if resource-resolvers = ∅ then return(failure)
      nondeterministically choose φ ∈ resource-resolvers
      M' ← the update of M with respect to Φ ∪ φ
      return(CPR(Φ ∪ φ, G, K, M', π))
end
```

Algoritmus pro **plánování s kronikami** rozšíříme o práci s minimálními konfliktními množinami (uloženy v M) a odstraňování zdrojových konfliktů