Plánování a rozvrhování

Roman Barták, KTIML

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

Na úvod

Co nás čeká?

- Plánování, konečně!
- Klasické plánování
  - Konceptuální model
  - Reprezentace problému
  - Plánovací algoritmy
    - Stavový prostor
    - Prostor plánů

Formalizace

Konceptuální model plánování

Konceptuální model

- **Plánování** se zabývá **volbou a organizací akcí**, které mění stav systému.

Systém Σ modelující stavy a přechody:

- **množina stavů S** (rekurzivně spočetná)
- **množina akcí A** (rekurzivně spočetná)
  - plánovač kontroluje akce!
  - no-op (prázdná akce)
- **množina událostí E** (rekurzivně spočetná)
  - události jsou mimo kontrolu plánovače!
  - neutrální událost ε
- **přechodová funkce γ**: $S \times A \times E \rightarrow P(S)$
  - někdy se akce a události aplikují odděleně γ: $S \times (A \cup E) \rightarrow P(S)$

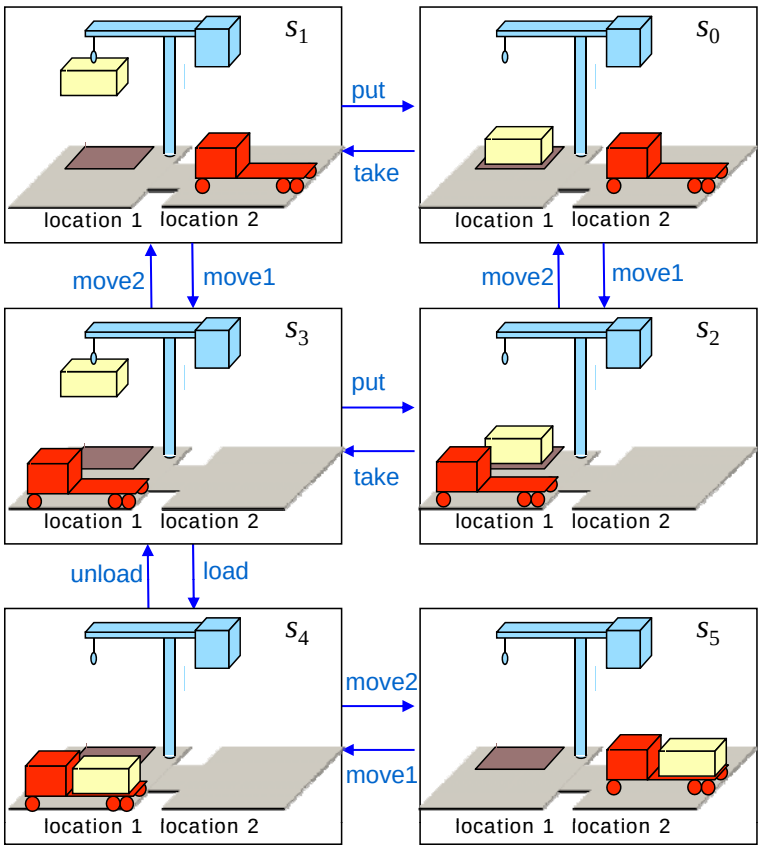
Plánování

- Cílem plánování je zjistit jaké akce a na které stavy se mají aplikovat, abychom z dané situace dosáhli požadovaných cílů.

Co jsou to cíle?

- **cílový stav** nebo množina cílových stavů
- **splnění dané podmínky** nad posloupností stavů, přes které systém přechází
  - např. stavy, kterým se vyhnout, nebo stavy, které se musí navštívit
- **optimalizace dané objektivní funkce** nad posloupností stavů
  - např. maximum nebo součet ohodnocení stavů

Příklad

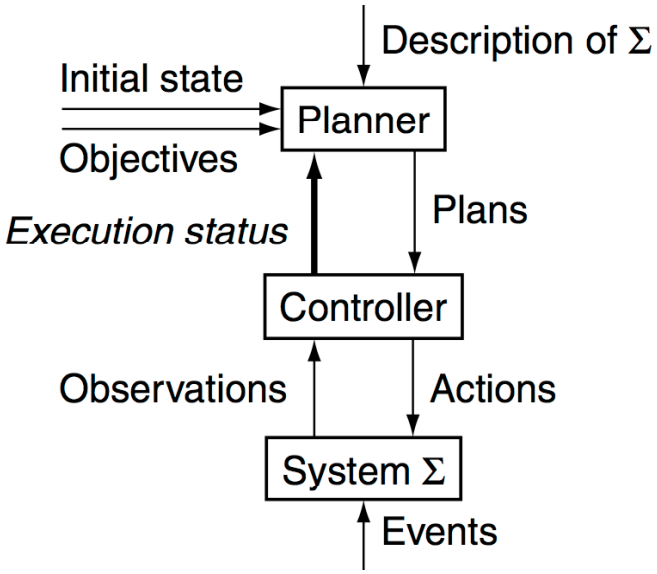


$\Sigma = (S,A,E,\gamma)$

- $S = \{s_0, \ldots, s_5\}$
- $E = \{\}$ resp. $\{\varepsilon\}$
- A = {move1, move2, put, take, load, unload}
- γ: obrázek

- počátek: s_0
- cíl: s_5

Jak to funguje?



- **Plánovač** generuje plány
- **Řadič** se stará o jejich realizaci
  - pro daný stav určí akci k provedení
  - pozorování převádějí reálný stav na modelovaný stav

Dynamické plánování umožňuje přeplánování na základě aktuálního stavu provádění plánu.

Zjednodušení modelu

- systém je **konečný**
- systém je „**plně pozorovatelný**“
  - Máme úplné informace o stavu systému.
- systém je **deterministický**
  - $\forall s \in S\ \forall u \in (A \cup E)$: $|\gamma(s,u)| \leq 1$
- systém je **statický**
  - Množina událostí je prázdná.

- **cíle** jsou **omezené**
  - Cílem je dosažení některého stavu z množiny cílových stavů.
- **plány** jsou **sekvenční**
  - Plánem je úplně uspořádaná posloupnost akcí.
- **čas** je **implicitní**
  - Akce i události jsou instantní (okamžité, tj. nemají žádné trvání).
- **plánujeme offline**
  - Stav systému se nemění v průběhu plánování.

Klasické plánování

- Pracujeme s deterministickým, statickým, konečným a plně pozorovatelným stavovým modelem s omezenými cíli a implicitním časem $\Sigma = (S,A,\gamma)$.

Plánovací problém $P = (\Sigma,s_0,g)$:

- s_0 je **počáteční stav**
- g charakterizuje **cílové stavy**

Řešením plánovacího problému P je posloupnost akcí $\langle a_1,a_2,\ldots,a_k\rangle$ odpovídající posloupnosti stavů $\langle s_0,s_1,\ldots,s_k\rangle$ takové, že $s_i=\gamma(s_{i-1},a_i)$ a s_k splňuje g

☞ **Klasické plánování (STRIPS plánování)** ☜

Zjednodušení?

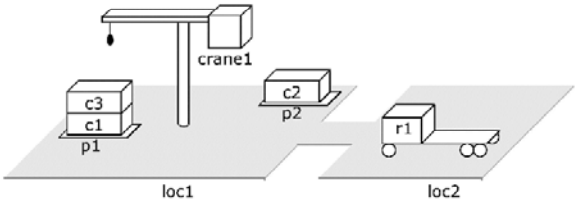
Plánování ve zjednodušeném modelu je „pouhé" hledání cesty v grafu.

Je to opravdu tak jednoduché?

5 míst, 3 hromady na místo, 100 kontejnerů, 3 roboti

↳ **10^{277} stavů**

tj. 10^{190} krát více než jsou největší odhady počtu částic ve vesmíru



Co nás bude zajímat?

- **Jak reprezentovat stavy a akce tak, aby nebylo potřeba vyjmenovat množiny S a A?**
  - připomeňme 10^{277} stavů vs. počet částic ve vesmíru
- **Jak efektivně hledat řešení plánovacího problému?**
  - Jak najít cestu v grafu s 10^{277} uzly?

Reprezentace

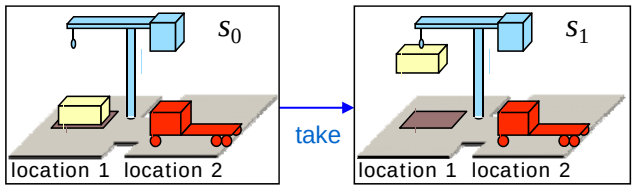
Reprezentace pro klasické plánování

Množinová reprezentace

Stav systému je popsán **množinou výroků**.
Př. {onground,at2}

Každá **akce** je syntaktický výraz specifikující:

- jaké výroky musí patřit do stavu, aby na něj byla akce aplikovatelná
  Př. take: {onground}
- jaké výroky akce přidá nebo smaže, aby vytvořila nový stav
  Př. take: {onground}$^-$, {holding}$^+$



Plánovací doména
množinová reprezentace

Nechť L= $\{p_1, ..., p_n\}$ je konečná množina výrokových symbolů (jazyk).

Plánovací doména Σ nad L je trojice (S,A,γ):

- S ⊆ P(L), tj. **stav** s je podmnožina L popisující jaké výroky platí
  - pokud p∈s, potom p ve stavu s platí
  - pokud p∉s, potom p ve stavu s neplatí
- **Akce** a∈A je trojice podmnožin L
  a = (precond(a),effects$^-$(a),effects$^+$(a))
  - effects$^-$(a) ∩ effects$^+$(a) = ∅
  - akce a je **použitelná** na stav s, pokud precond(a) ⊆ s
- **Přechodová funkce** γ:
  - γ(s,a) = (s – effects$^-$(a)) ∪ effects$^+$(a), je-li a použitelná na s

Plánovací problém

množinová reprezentace

Plánovací problém P je trojice (Σ, s_0, g):

- $\Sigma=(S,A,\gamma)$ je plánovací doména nad L
- s_0 je počáteční stav, $s_0 \in S$
- $g \subseteq L$ je množina cílových výroků
  - $S_g = \{s \in S \mid g \subseteq s\}$ množina cílových stavů

Plán π je posloupnost akcí $\langle a_1, a_2, \ldots, a_k \rangle$

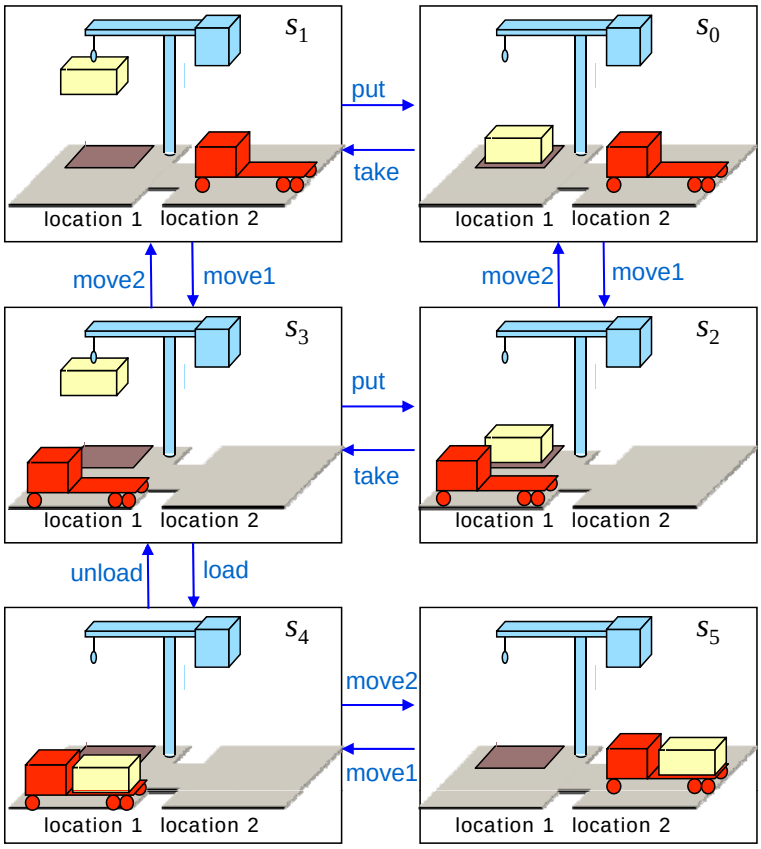
- **délka plánu** π je $k = |\pi|$
- **stav produkovaný plánem** π (zobecnění funkce γ)
  - $\gamma(s,\pi) = s$, je-li k=0 (plán π je prázdný)
  - $\gamma(s,\pi) = \gamma(\gamma(s,a_1), \langle a_2,\ldots,a_k \rangle)$, je-li k>0 a a_1 je použitelná na s
  - $\gamma(s,\pi)$ = nedefinováno v ostatních případech

Plán π je **řešením** P právě když $g \subseteq \gamma(s_0,\pi)$.

- **redundantní řešení**: obsahuje spojitou pod-posloupnost, která je také řešením P
- **minimální řešení**: neexistuje řešení P s kratší délkou

Příklad

množinové reprezentace



L = {onground, onrobot, holding, at1, at2}

s_0 = {onground, at2}

g = {onrobot}

load = (
{holding,at1},
{holding},
{onrobot})

⟨take,move1,load,move2⟩ je plán, ale není minimální

Dosažitelnost

množinová reprezentace

Přímí následníci stavu s:

$\Gamma(s) = \{\gamma(s,a) \mid a \in A \text{ je aplikovatelná na } s\}$

Dosažitelné stavy:

$\Gamma_\infty(s) = \Gamma(s) \cup \Gamma^2(s) \cup \ldots$

Plánovací problém má řešení právě když $S_g \cap \Gamma_\infty(s_0) \neq \emptyset$.

Akce a je relevantní pro cíl g právě když:

$g \cap effects^+(a) \neq \emptyset$

$g \cap effects^-(a) = \emptyset$

Regresní (zpětná) množina cíle g pro (relevantní) akci a:

$\gamma^{-1}(g,a) = (g - effects^+(a)) \cup precond(a)$

$\Gamma^{-1}(g) = \{\gamma^{-1}(g,a) \mid a \in A \text{ je relevantní pro } g\}$

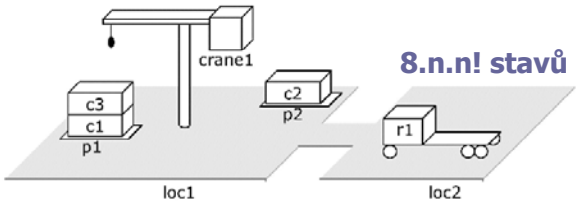
$\Gamma_\infty^{-1}(g) = \Gamma^{-1}(g) \cup \Gamma^{-2}(g) \cup \ldots$

Plánovací problém má řešení právě když s_0 je nadmnožinou nějakého prvku z $\Gamma_\infty^{-1}(g)$.

Vlastnosti

množinové reprezentace

- **Srozumitelnost**
  - přehlednější než výčet stavů
  - Kolik stavů pro n kontejnerů?



8.n.n! stavů

{nothing-on-c3, c3-on-c1,c1-on-pile1, nothing-on-c2, c2-on-pile2, crane-empty, robot-at-loc2}

- **Výpočty**
  - přechodová funkce se snadno realizuje pomocí množinových operací
  - pokud $precond(a) \subseteq s$, potom $\gamma(s,a) = (s - effects^-(a)) \cup effects^+(a)$,
- **Expresivita**
  - Některé množiny výroků neodpovídají žádnému stavu
    - {holding, onrobot, at2}
  - Některé stavové prostory stejně mají obrovskou množinovou reprezentaci.

Klasická reprezentace

- **Klasická reprezentace** zobecňuje množinovou reprezentaci směrem k **predikátové logice**:
  - **Stavy** jsou množiny logických atomů, které jsou v dané interpretaci buď pravda nebo nepravda.
  - **Akce** jsou reprezentovány plánovacími operátory, které mění pravdivostní hodnotu těchto atomů.

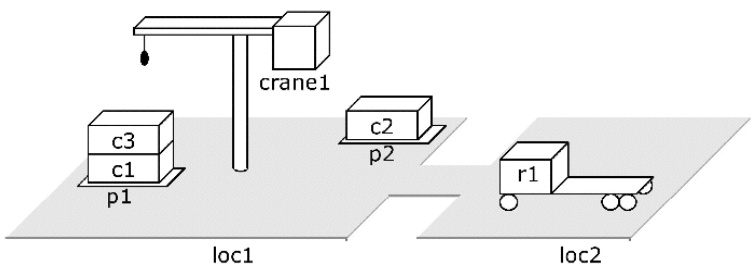
Přesněji:

- L (**jazyk**) je konečná množina predikátových symbolů a konstant (nemáme funkce!).
- **Atom** je predikátový symbol s argumenty, např. on(c3,c1).
- Můžeme používat **proměnné**, např. on(x,y).

Reprezentace stavů

klasická reprezentace

- **Stav je množina instanciovaných atomů** (bez proměnných). Opět jich je konečně mnoho!



{attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.

- Pravdivostní hodnota některých atomů se mění
  - **flexibilní atomy** (fluent)
  - např. at(r1,loc2)
- Některé atomy nemění svojí pravdivostní hodnotu s různými stavy
  - **neměnné atomy** (rigid)
  - např. adjacent(loc1,loc2)

- **Předpoklad uzavřeného světa** (closed world assumption) Atom, který není ve stavu explicitně uveden, neplatí!

Plánovací operátory
klasická reprezentace

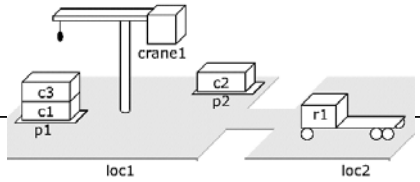
operátor o je trojice:
(name(o), precond(o), effects(o))

- **name(o): jméno operátoru** ve tvaru $n(x_1,...,x_k)$
  - n: symbol operátoru (jednoznačný pro každý operátor)
  - $x_1,...,x_k$: symboly proměnných (parametry operátoru)
    - musí obsahovat všechny symboly proměnných v operátoru!
- **precond(o): předpoklady**
  - literály, které musí být splnitelné, aby šlo operátor použít
- **effects(o): efekty**
  - literály, které se stanou pravdivými aplikací operátoru (nesmí to být neměnné atomy!)

take(k, l, c, d, p)
;; crane k at location l takes c off of d in pile p
precond: belong(k, l), attached(p, l), empty(k), top(c, p), on(c, d)
effects: holding(k, c), $\neg$ empty(k), $\neg$ in(c, p), $\neg$ top(c, p), $\neg$ on(c, d), top(d, p)

Akce
klasická reprezentace

Akce jsou plně instanciované operátory
– za proměnné jsou dosazeny konstanty

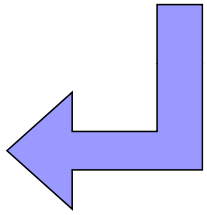


take(k, l, c, d, p)
;; crane k at location l takes c off of d in pile p
precond: belong(k, l), attached(p, l), empty(k), top(c, p), on(c, d)
effects: holding(k, c), $\neg$ empty(k), $\neg$ in(c, p), $\neg$ top(c, p), $\neg$ on(c, d), top(d, p)

operátor

take(crane1,loc1,c3,c1,p1)
;; crane crane1 at location loc1 takes c3 off c1 in pile p1
precond: belong(crane1,loc1), attached(p1,loc1),
empty(crane1), top(c3,p1), on(c3,c1)
effects: holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1),
¬top(c3,p1), ¬on(c3,c1), top(c1,p1)

akce



Aplikace akce
klasická reprezentace

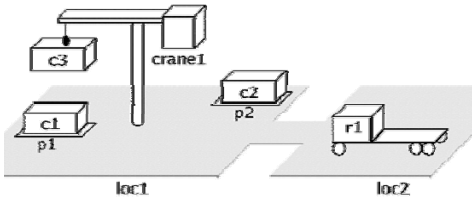
Notace:

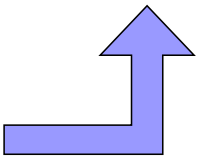
- S^+ = {pozitivní atomy v S}
- S^- = {atomy, jejichž negace je v S}

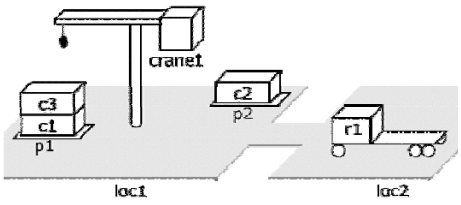
Akce a je **použitelná** na stav s právě když
$precond^+(a) \subseteq s \ \wedge \ precond^-(a) \cap s = \varnothing$

Výsledkem aplikace akce a na s je
$\gamma(s,a) = (s - effects^-(a)) \cup effects^+(a)$

take(crane1,loc1,c3,c1,p1)
;; crane crane1 at location loc1 takes c3 off c1 in pile p1
precond: belong(crane1,loc1), attached(p1,loc1), empty(crane1), top(c3,p1), on(c3,c1)
effects: holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1), ¬top(c3,p1), ¬on(c3,c1), top(c1,p1)







Plánovací doména
klasická reprezentace

Nechť L je jazyk a O je množina operátorů.

Plánovací doména Σ nad jazykem L a s operátory O je trojice (S,A,γ):

- **stavy** $S \subseteq P(\{\text{všechny instanciované atomy nad L}\})$
- **akce** A = {všechny instanciované operátory z O nad L}
  - akce a je **použitelná** na stav s, pokud $precond^+(a) \subseteq s \ \wedge \ precond^-(a) \cap s = \varnothing$
- **přechodová funkce** γ:
  - $\gamma(s,a) = (s - effects^-(a)) \cup effects^+(a)$, je-li a použitelná na s
  - S je uzavřená vzhledem ke γ (je-li $s \in S$, potom pro každou akci a aplikovatelnou na s platí $\gamma(s,a) \in S$)

Plánovací problém
klasická reprezentace

Plánovací problém P je trojice (Σ,s_0,g):

- $\Sigma=(S,A,\gamma)$ je plánovací doména
- s_0 je počáteční stav, $s_0 \in S$
- g je množina instanciovaných literálů
  - stav s splňuje g právě tehdy, když $g^+ \subseteq s \ \wedge \ g^- \cap s = \varnothing$
  - $S_g = \{s \in S \mid s \text{ splňuje } g\}$ - množina cílových stavů

Zápis plánovacího problému je trojice (O,s_0,g).

Plány a řešení
klasická reprezentace

Plán π je posloupnost akcí $\langle a_1,a_2,\ldots,a_k \rangle$.

Plán π je **řešením** P právě když $\gamma(s_0,\pi)$ splňuje g.

Plánovací problém má řešení právě když $S_g \cap \Gamma_\infty(s_0) \neq \varnothing$.

Plánovací problém má řešení právě když s_0 je nadmnožinou nějakého prvku z $\Gamma_\infty^{-1}(g)$ (ale trochu jiná definice γ^{-1}).

Akce a je relevantní pro cíl g právě když:

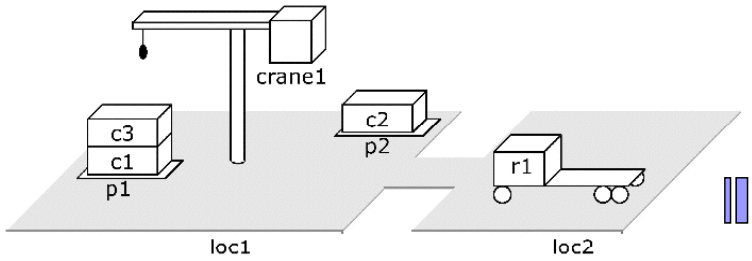
akce přispívá do g: $g \cap effects(a) \neq \varnothing$

efekty akce nejsou v konfliktu s g:

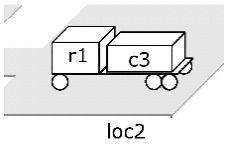
- $g^- \cap effects^+(a) = \varnothing$
- $g^+ \cap effects^-(a) = \varnothing$

Regresní (zpětná) množina cíle g pro (relevantní) akci a:
$\gamma^{-1}(g,a) = (g - effects(a)) \cup precond(a)$

Ukázka plánu
klasická reprezentace



náš cíl

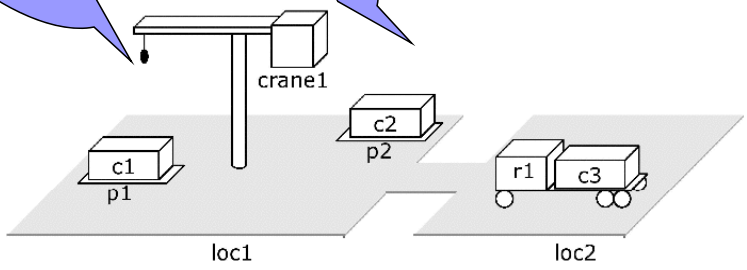


s_1= {attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.

g = {loaded(r1,c3), at(r1,loc2)}

⟨move(r1,loc2,loc1), take(crane1,loc1,c3,c1,p1), load(crane1,loc1,c3,r1), move(r1,loc1,loc2)⟩

⟨take(crane1,loc1,c3,c1,p1), move(r1,loc2,loc1), load(crane1,loc1,c3,r1), move(r1,loc1,loc2)⟩



Rozšíření
klasické reprezentace

- **Syntaktická rozšíření**
  - typované proměnné (konstanty z jazyka mají svůj typ)
    - *např. typ robot: rob1, rob2, rob3*
  - existenčně kvantifikované cíle (uzavřená formule!)
    - *např. $\exists x,y$ (on(x,c1) $\wedge$ on(y,c2))*
- **Podmíněné operátory**
  - pod jedním jménem je ukryto více „podobných" mini-operátorů, každý s vlastními předpoklady a efekty – aplikují se najednou všechny mini-operátory, jejichž předpoklady jsou splněny
  - *např. přepnutí vypínače vede ke zhasnutí, pokud bylo rozsvíceno, nebo k rozsvícení, pokud bylo zhasnuto*
- **Disjunktivní předpoklady**
  - předpokladem může být libovolná formule
  - *např. robot může přejet z A do B, pokud z A do B vede cesta nebo pokud má robot pohon na čtyři kola*

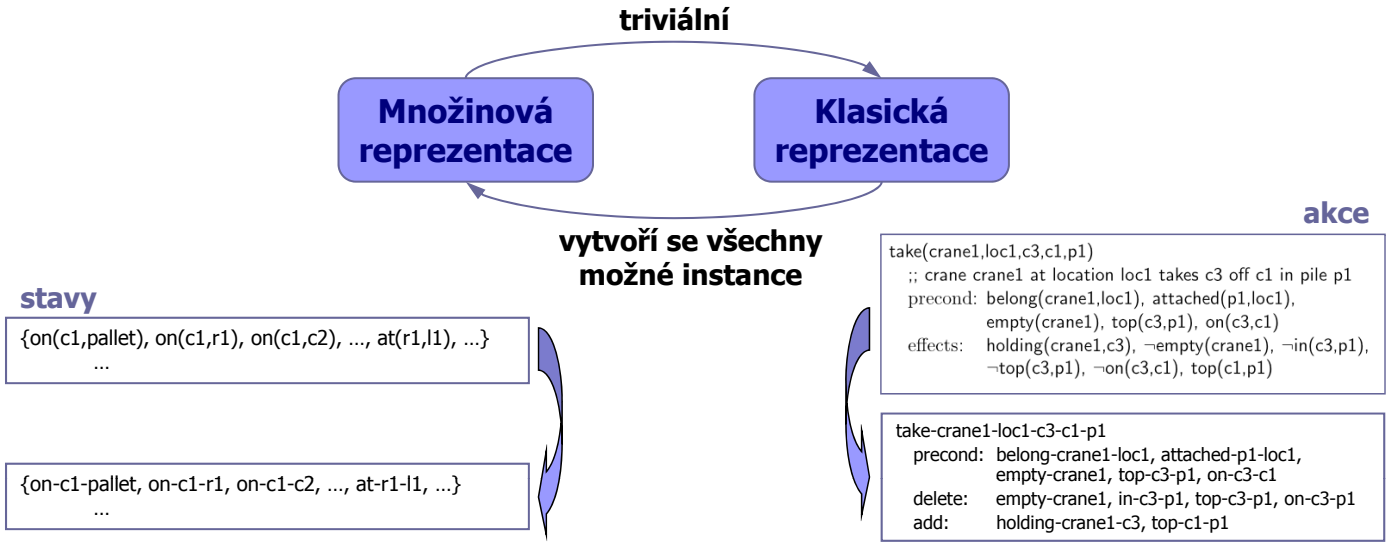
Další rozšíření
klasické reprezentace

- **Připojené procedury (k operátorům)**
  - umožňují testovat složitější (například numerické) předpoklady
  - *např. weight(c) ≤ maxweight(r)*
- **Axiomy**
  - pro automatické odvození některých faktů
  - *např. ∀ l,l‘ (adjacent(l,l‘) ⇔ adjacent(l‘,l))*
  - není problém pro neměnné atomy, ale musí se udělat opatrně pro flexibilní atomy
    - *∀k (¬∃x holding(k,x) ⇒ empty(k))*
    - *∀k (∃x holding(k,x) ⇒ ¬empty(k))*

    Flexibilní atomy jsou potom rozděleny na
    - **primární atomy**, které se mohou používat v předpokladech i efektech (holding)
    - **sekundární atomy**, které se mohou používat jen v předpokladech, tj. nesmí být v efektech (empty)

Srovnání reprezentací

- **Vyjadřovací síla** obou reprezentací **je stejná** (co lze reprezentovat množinově, lze i klasicky a naopak).
- Při převodu z klasické na množinovou reprezentaci ale může dojít k **exponenciálnímu nárůstu velikosti**.



Domácí úkol

- **Zopakovat prohledávací algoritmy.**
- **Navrhnout množinovou a klasickou reprezentaci pro svět kostek.**

Svět kostek (the blocks world)

- nekonečně velký stůl, konečný počet kostek
- poloha kostky na stole nás nezajímá
- kostka může ležet buď na stole nebo na jiné kostce
- při plánování chceme přesouvat kostky tak, že v dané chvíli můžeme držet maximálně jednu kostku

například