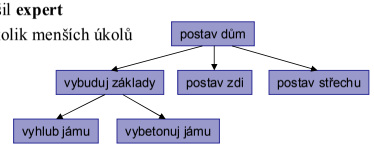
PLÁNOVÁNÍ A ROZVRHOVÁNÍ
Hierarchické plánování

Pavel Surynek, 2006

surynek@kti.mff.cuni.cz

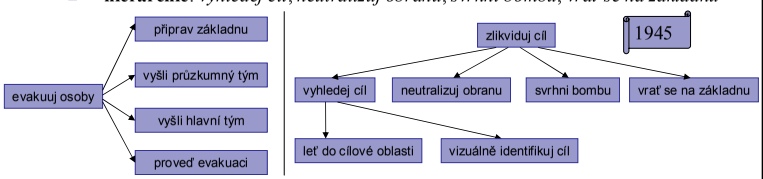
Základní motivace

- Stavba domu
  - Úkolem je **postavit dům**, chceme tedy určit **posloupnost činností**, jejichž postupným vykonáním na konci obdržíme hotový dům.
  - počáteční stav světa obsahuje různě **dislokovaný materiál**
    - je možné vykonávat běžné stavební činnosti pomocí běžných nástrojů
  - **klasicky:** akce typu *vezmi cihlu na místě A*, *polož cihlu na místo B* nebo *polož jeden kus materiálu do transportéru*, *přesuň transportér*
    - obrovský stavový prostor
    - ztráta znalostí o plánovací doméně (těžko poznat, že se staví dům)
  - **hierarchicky:** úkoly typu *vybuduj základy*, *postav zdi*, *postav střechu*
    - odpovídá postupu, jak by úlohu řešil **expert**
    - každý úkol se dále rozkládá na několik menších úkolů



Další aplikace: armáda a záchranáři

- Plánování činností bojových uskupení
  - přirozená **plánovací hierarchie** podle **hierarchie velení**
  - letka letadel má splnit jistý úkol = zlikvidovat cíl
  - **hierarchie**: *vyhledej cíl*, *neutralizuj obranu*, *svrhni bombu*, *vrať se na základnu*



- Záchranné operace - **evakuace osob** (systém HICAP)
  - silně **interaktivní**, dynamické doplňování znalostí, evakuačních úkolů, zdrojů
  - **hierarchie:** *připrav základnu*, *vyšli průzkumný tým*, *vyšli hlavní tým*, *proveď evakuaci*
- V **praxi** je hierarchické plánování **nejvíce používanou** plánovací technikou.
  - poskytuje účinný způsob pro včlenění **znalostí experta**

Formální systém hierarchického plánování

HTN Planning

(*Hierarchical Task Network Planning*)

Syntaxe HTN plánování - popis jazyka

- Formální systém hierarchického plánování používá jazyk prvního řádu (proměnné pro individua) L
- Jazyk L obsahuje množiny V,C,P,F,T a N, kde
  - V - nekonečná množina symbolů pro proměnné
  - C - konečná množina symbolů pro konstanty
  - P - konečná množina symbolů pro predikáty
  - F - konečná množina symbolů pro tzv. *primitivní úkoly* (primitive tasks)
  - T - konečná množina symbolů pro tzv. *složené úkoly* (compound tasks)
  - N - nekonečná množina pomocných symbolů pro úkoly

Poznámka: **žádné** funkční symboly (alespoň zatím).

- **Primitivní** úkol (primitive task)
  - syntaktická konstrukce $\boldsymbol{f}(x_1,x_2,...,x_n)$, kde $\boldsymbol{f} \in F$ (symbol pro primitivní úkol) a $x_1,x_2,...,x_n$ jsou termy (proměnné nebo konstanty)
  - například: ***move***$(co, odkud, kam)$, přičemž ***move*** $\in F$ a $co, odkud, kam \in V$ nebo ***move***$(robot1, místoA, místoB)$, kde $robot1, místoA, místoB \in C$
  - primitivní = jednoduchý, lze jej **přímo vykonat** (odpovídá **akci** v klasickém plánování), není rozložitelný na menší úkoly (*robot1* přejede *z místoA* na *místoB*)

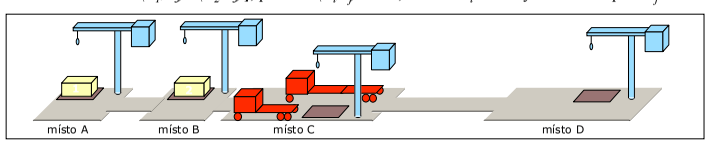
Syntaxe HTN plánování - pokračování

- **Složený** úkol (compound task)
  - syntaktická konstrukce $\boldsymbol{t}(x_1,x_2,...,x_n)$, kde $\boldsymbol{t} \in T$ (symbol pro složený úkol) a $x_1,x_2,...,x_n$ jsou termy (proměnné nebo konstanty)
  - například: ***transfer***$(co, odkud, kam)$, kde ***transfer*** $\in T$ a $co, odkud, kam \in V$ nebo ***transfer***$(kontejner1, místoA, místoB)$, kde $kontejner1, místoA, místoB \in C$
  - složený = nutno provést více kroků (podúkolů), **nelze vykonat** přímo (*kontejner1* je třeba nejdřív naložit na robota, pak přemístit robota místoA→místoB a vyložit)
- **Cílový** úkol (goal task)
  - syntaktická konstrukce *achieve[l]*, kde *l* literál (tedy atomická formule nebo její negace, atomická formule je tvaru $p(y_1,y_2,...,y_n)$, kde $p \in P$ (predikátový symbol) a $y_1,y_2,...,y_n$ jsou termy (proměnné nebo konstanty))
  - například: *achieve[****at****(co, kde)]*, přičemž ***at*** $\in P$ a $co, kde \in V$ nebo *achieve[****at****(kontejner1, místoB)]*, kde $kontejner1, místoB \in C$
  - určuje, co je třeba splnit, v klasickém plánování odpovídá **části cílového stavu**, podobný složenému úkolu - **nelze vykonat** přímo
- **Složené** a **cílové** úkoly = **neprimitivní** úkoly (non-primitive tasks)

Syntaxe HTN plánování - pokračování 2

- **Úkolová síť** (task network)
  - syntaktická konstrukce $\boldsymbol{[}(n_1{:}\alpha_1),(n_2{:}\alpha_2),...,(n_m{:}\alpha_m),\Phi\boldsymbol{]}$, kde α_i pro $i \in \{1...m\}$ jsou úkoly (primitivní, složené nebo i cílový), $n_i \in N$ pro $i \in \{1...m\}$ jsou odpovídající úkolové symboly a Φ je Boolovská formule říkající něco o $n_1,n_2,...,n_m$ (sestavena z určitých dovolených atomů - **ne úplně libovolná**)
  - v klasickém plánování odpovídá **cílovému stavu**
  - například:
    - $\boldsymbol{[}(n_1{:}achieve[$***at***$(kontejner1, kontejner2, místoD)]), \Phi{=}\emptyset\boldsymbol{]}$ nebo
    - $\boldsymbol{[}(n_1{:}$***transfer***$(kontejner1, místoA, místoC)), (n_2{:}$***transfer***$(kontejner2, místoB, místoC)), (n_3{:}$***transfer-two***$(kontejner1, kontejner2, místoC, místoD)), \Phi{=}(n_1,n_3)\&(n_2,n_3)\boldsymbol{]}$, přičemž (n_i,n_j) značí, že úkol n_i musí být dokončen před n_j

Primitivní úkolová síť (jen primitivní úkoly), **neprimitivní úkolová síť** (jinak)



- **Stav** je seznam platných základních atomů (jen konstanty)

HTN plánovací doména

- **Operátor** (= provedení **primitivního** úkolu)
  - syntaktická konstrukce **operator**[$\boldsymbol{f}(x_1,x_2,...,x_n)$, **precond**$(l_1,l_2,...,l_m)$, **effects**$(k_1,k_2,...,k_{m'})$], kde $\boldsymbol{f} \in F$, $l_1,l_2,...,l_m$ jsou literály, které musí splňovat **stav** než lze primitivní úkol $\boldsymbol{f}$ provést a $k_1,k_2,...,k_{m'}$ jsou literály popisující efekty primitivního úkolu
    - *termy* $x_1,x_2,...,x_n$ se mohou vyskytovat v rámci $l_1,l_2,...,l_m$ či $k_1,k_2,...,k_{m'}$
  - totéž jako akce v klasickém plánování
  - například: **operator**[***move****(robot1, místoA, místoB)*, **precond**(***at****(robot1, místoA)*, **effects**(¬***at****(robot1, místoA)*, ***at****(robot1, místoB))*]
- **Metoda** (= návod jak naložit s **neprimitivním** úkolem)
  - syntaktická konstrukce **(α:d)**, kde **α** je neprimitivní úkol a **d** je úkolová síť
  - neprimitivní úkol **α** lze splnit splněním úkolové sítě **d**
  - například: $(achieve[$***at***$(co1, co2, kam)]\boldsymbol{:}\boldsymbol{[}(n_1{:}$***transfer***$(co1, odkud1, místoC)), (n_2{:}$***transfer***$(co2, odkud2, místoC)), (n_3{:}$***transfer-two***$(co1, co2, místoC, kam)), \Phi{=}(n_1,n_3)\&(n_2,n_3)\boldsymbol{]})$
- **Plánovací doména** D je dvojice množin operátorů a metod <O,M>

Sémantika HTN

- **Použití operátoru** (provedení akce)
  - Operátor je aplikován **na stav**, aplikuje se **vždy základní** instance operátoru (jen konstanty), **výsledkem je stav**.
  - *Základní* **operator**[$f(x_1,x_2,...,x_n)$, **precond**($l_1,l_2,...,l_m$), **effects**($k_1,k_2,...,k_{m'}$)] je použitelný ke stavu **s**, když **s** splňuje $l_1,l_2,...,l_m$, výsledkem použití operátoru je nový stav (**s-effects**$^-$($k_1,k_2,...,k_{m'}$))$\cup$**effects**$^+$($k_1,k_2,...,k_{m'}$).
  - stejné jako akce v klasickém plánování
- **Použití metody** (rozklad úkolové sítě)
  - Metoda je aplikována **na úkolovou síť**, pro aplikaci **nemusí být základní** (zůstává **lifted** - liftovaná), **výsledkem je úkolová síť**.
  - **(α:d)** je použitelná k úkolové síti $[(n_1:\alpha_1),(n_2:\alpha_2),...,(n_m:\alpha_m),\Phi]$, když **α** je unifikovatelný s některým z $\alpha_1,\alpha_2,...,\alpha_m$
  - Nechť θ je **nejobecnější unifikace** (mgu) **α** a α_i, pak **výsledkem** aplikace metody **(α:d)** je úkolová síť $[(n_1:\alpha_1),(n_2:\alpha_2),...,(n_{i-1}:\alpha_{i-1}),\mathbf{d\theta},(n_{i+1}:\alpha_{i+1}),...,(n_m:\alpha_m),\Phi']$.
  - Φ' je upravená formule Φ
    - například: (n_1,n_j) je třeba nahradit **konjunkcí všech** (n_1,n) pro n probíhající všechny úkoly z **dθ**, podobně i jiné atomy ve Φ

Příklad rozkladu sítě pomocí metody

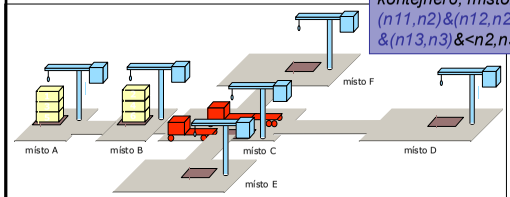
- Původní úkolová síť

[(n1:achieve[**at**(kontejner1, kontejner2, místoD))], (n2:achieve[**at**(kontejner3, kontejner4, místoE))], (n3:achieve[**at**(kontejner5, kontejner6, místoF))],Φ=<n1,n2,n3>**]**, přičemž <n1,n2,n3> je atom určující, že úkoly mají být provedeny přesně v pořadí n1,n2,n3

- Metoda

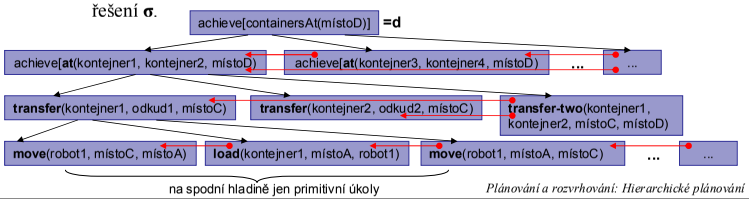
(achieve[**at**(co1, co2, kam)]:[(n1:**transfer**(co1, odkud1, místoC)), (n2:**transfer**(co2, odkud2, místoC)), (n3:**transfer-two**(co1, co2, místoC, kam)), Φ=(n1,n3)&(n2,n3)])

[(n11:**transfer**(kontejner1, odkud1, místoC)), (n12:**transfer**(kontejner2, odkud2, místoC)), (n13:**transfer-two**(kontejner1, kontejner2, místoC, místoD), (n2:achieve[**at**(kontejner3, kontejner4, místoE))], (n3:achieve[**at**(kontejner5, kontejner6, místoF))],Φ=(n11,n13)&(n12,n13)&(n11,n2)&(n12,n2)&(n13,n2)&(n11,n3)&(n12,n3)&(n13,n3)&<n2,n3>**]**



HTN plánovací problém

- **HTN plánovací problém** je trojice **<d, I, D>**, kde
  - **d** je úkolová síť, pro kterou vytváříme plán (ještě nemáme definován), **I** je počáteční stav světa (seznam atomů) a **D** je plánovací doména (<O,M>).
- **Plán řešící** problém **<d, I, D>** je posloupnost **σ** základních (jen konstanty) primitivních úkolů, která **splňuje podmínky**:
  - Když síť **d** je **primitivní**: pak **σ** je **uspořádání primitivních úkolů** v **d** tak, že **σ** je **proveditelná** z počátečního stavu **I** (souhlasí předpoklady operátorů) a jsou pro **σ** splněny všechny **podmínky** v síti **d** (je splněna formule Φ).
  - Když síť **d** je **neprimitivní**: existuje-li posloupnost rozkladů úkolové sítě **d** podle plánovací domény **D**, že výsledkem je primitivní síť **d'** a problém **<d', I, D>** má řešení **σ**.



Algoritmy hierarchického plánování

Obecný a uspořádaný algoritmus

UMCP: Plánovací procedura pro HTN

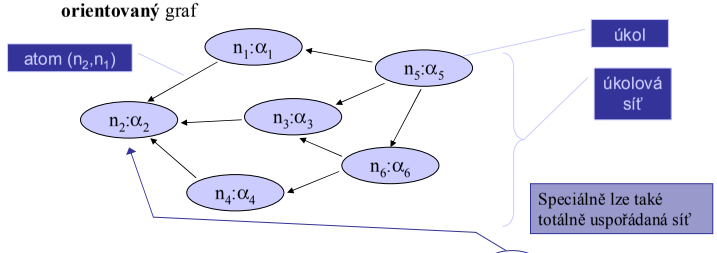
```
procedure UMCP
1. vstupní plánovací problém
   P = <d,I,<O,M>>
2. if d je primitivní then
     nedeterministicky zvol σ uspořádání
     a základní instance primitivních
     úkolů v d
     if σ je řešením problému P then
       return σ
     else
       return FAILURE
3. nedeterministicky zvol neprimitivní
   úkol (n:α) v síti d
4. nedeterministicky zvol metodu m∈M
   pro α
5. d ← redukce sítě d podle úkolu (n:α)
   a metody m
6. T ← τ(d,I,D)
7. nedeterministicky zvol d' ∈T, d ← d'
8. goto krok 2.
```

nepovinné

- **UMCP** - Universal Method Composition Planner
- Vytváří plán podle přesně definice
- Používá se tzv. kritická funkce τ (*critic function*) k odstraňování konfliktů a k prořezávání prohledávaného prostoru (místo pro zapojení vlastní heuristiky)
- Pro danou úkolovou síť **d**, počáteční stav **I** a plánovací doménu **D** funkce τ vrací množinu úkolových sítí, přičemž každá z nich odstraňuje *nějaký konflikt* v **d**
- Podmínky na τ:
  - když $d' \in \tau(d,I,D)$, pak $sol(d',I,D) \subseteq sol(d,I,D)$
  - když $\sigma \in sol_k(d,I,D)$, pak existuje $d' \in \tau(d,I,D)$, že $\sigma \in sol_k(d',I,D)$

Částečně uspořádané HTN plánování

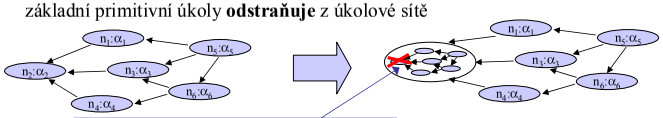
- Někdy (často) je podmínka v rámci formule Φ speciálního tvaru
  - Určuje částečné uspořádání úkolů v rámci úkolové sítě, tedy **acyklický orientovaný** graf



  - Toho lze využít, můžeme rozkládat (podle metod) vždy **uzel, který nemá žádné předchůdce** (zde se jedná o uzel s úkolem (n_2:α_2)).
  - Jaký by to mělo smysl ?
    - Lze **udržovat aktuální stav** světa, což pomůže k odstranění neurčitosti. Jak ?
    - **Metody** mohou mít **předpoklady** (stejně jako operátory), **redukce prostoru**, který je nutné prohledávat.

Uspořádané HTN: metody s předpoklady

- **Metoda** - nyní trochu jinak
  - syntaktická konstrukce **(α:precond(l_1,l_2,...,l_m):d)**, kde **α** je neprimitivní úkol, l_1,l_2,...,l_m jsou literály, které musí splňovat **aktuální stav**, aby bylo možné metodu na danou síť aplikovat a **d** je úkolová síť, na kterou se úkol **α** redukuje
  - například: (*achieve[**at**(co1, co2, kam)*]**:precond(*onTop(co1)*, *onTop(co2)*):** [(n_1:***transfer***(co1, odkud1, místoC)), (n_2:***transfer***(co2, odkud2, místoC)), (n_3:***transfer-two***(co1, co2, místoC, kam)), Φ=(n_1,n_3)&(n_2,n_3)])
- Použito v algoritmu SHOP2 (*Simple Hierarchical Ordered Planner 2*)
  - **metody s předpoklady**, nutno používat **základní instance metod**
  - **výsledný plán** (posloupnost primitivních úkolů) sestavuje v **pořadí**, jak bude nakonec prováděn (žádná nedeterministická volba uspořádání jako v UMCP)
  - udržuje **aktuální stav**, redukce podle metody odpovídá náhradě uzlu grafu sítě
  - základní primitivní úkoly **odstraňuje** z úkolové sítě



Základní primitivní bez předchůdce úkol je odstraněn

SHOP2: Algoritmus pro uspořádané HTN

```
procedure SHOP2
1. vstupní plánovací problém P = <d,I,<O,M>>
2. σ ← prázdný plán
3. s ← I
4. if d = ∅ return σ
5. nedeterministicky zvol (n:α) jeden z úkolů v d, který
   nemá předchůdce
6. if α je primitivní then
     active ← množina základních instancí operátorů z O
              aplikovatelných na stav s
     if active = ∅ then return FAILURE
     else
       nedeterministicky zvol jeden operátor o z množiny active
       s ← aplikace operátoru o na stav s
       d ← d po odstranění primitivního úkolu (n:α)
       σ ← σ∪{primitivní úkol příslušný operátoru o}
7. else {α je neprimitivní}
     active ← množina základních instancí metod z M
              aplikovatelných na stav s
     if active = ∅ then return FAILURE
     else
       nedeterministicky zvol jednu metodu m z množiny active
       d ← d po aplikaci metody m na neprimitivní úkol (n:α)
8. goto krok 4.
```

Různé dodatky k hierarchickému plánování

Vyjadřovací síla, složitost, rozšíření, srovnání

Vyjadřovací síla HTN

- **Věta:** HTN plánování má **alespoň stejnou** vyjadřovací sílu ve smyslu inkluze množin problémů, které lze modelovat, jako klasické plánování.
  - **Důkaz:** Stačí z formálního systému hierarchického plánování **vyřadit metody**, tím dostáváme systém ekvivalentní klasickému plánování.
- **Věta:** HTN plánování má **ostře větší** vyjadřovací sílu ve smyslu inkluze množin problémů, které lze modelovat, než klasické plánování.
  - **Důkaz:** Množina řešení klasického plánovacího problému tvoří regulární jazyk (rozpoznávaný konečným automatem), zatímco množina řešení HTN plánovacího problému tvoří bezkontextový jazyk (rozpoznávaný zásobníkovým automatem). Víme, že existují bezkontextové jazyky, které nejsou regulární.
    - například: **operator**[***o1()***, **precond**(), **effects**()], **operator**[***o2()***, **precond**(), **effects**()], **((t1():[**n_1**:*o1()*, n_2**:*t1()*, n_3**:*o2()*]):**<n_1,n_2,n_3>), **((t1():Ø)**
      řešení tvoří jazyk $o1^i\ o2^i()$, který je bezkontextový, ale není regulární
- **Věta:** Existence plánu v HTN plánování je **algoritmicky nerozhodnutelný** problém.
  - **Důkaz:** Bez důkazu. Idea - problém rozhodování o existenci plánu se převede na problém rozhodování, zda mají dva bezkontextové jazyky neprázdný průnik.

Složitost HTN plánování

Problém **existence plánu**

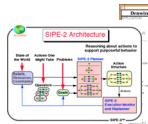
Omezení na neprimitivní úkoly	**Totální uspořádání úkolů**	**Bez proměnných**	**S proměnnými**
Bez omezení	**Ne**	Nerozhodnutelný	Nerozhodnutelný
	Ano	PSPACE-těžký EXPTIME	EXPSPACE-těžký double-EXPTIME
Regularita (neprimitivní pouze poslední)	**Nehraje roli**	PSPACE-úplný	EXPSPACE-úplný
Bez neprimitivních úkolů	**Ne**	NP-úplný	NP-úplný
	Ano	Polynomiální	NP-úplný

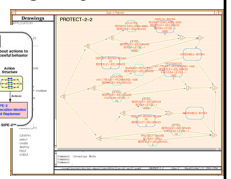
Rozšíření, srovnání

- **Předpoklady** metod a operátorů lze rozšířit na obecné formule
  - konjunkce, disjunkce, negace, obecná/existenční kvantifikace, numerické výpočty (operace spotřebuje jisté množství zdroje)
  - potřeba **axiomatické odvozování předpokladů**, které nejsou explicitně obsaženy v aktuálním stavu
  - může být **komplikované**, axiomy **Hornovské klauzule**
  - například: **operator**[**payDriver**(a,x,y), **precond**(cash(a)>1.5+0.5*distance(x,y)), **effects**(cash(a) ←cash(a)-1.5+0.5*distance(x,y))]
- Zavedení **funkčních symbolů**
  - nekonečně mnoho základních instancí metod a operátorů (řešení - *liftovat*)
- Další zajímavá rozšíření
  - zavedení **času** - operace mají trvání
  - využití **plánovacího grafu** (GraphHTN), plánovací graf určuje **dosažitelnost** stavů
- Klady a zápory
  - účinně zakódované **znalosti o doméně**, ale **potřeba experta**, který to udělá

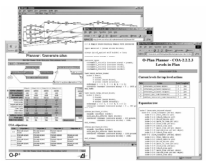
Konkrétní systémy

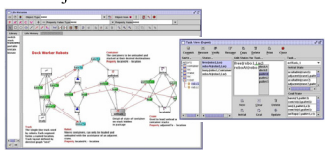
- Konkrétní systémy
  - **Nonlin** - jeden z prvních systémů HTN, ještě neexistoval formální systém HTN
  - **UMCP** - první systém založený na formálním systému, prokazatelně úplný
  - **SHOP2** - vítěz jedné ze 4 kategorií na International Planning Competition 2002
  - **SIPE-2** - armáda, letecká doprava, minimalizace škod při ropné havárii, průmyslová výroba





  - **O-Plan** - evakuační operace, záchrana rukojmích





  - **GIPO II (III)** - objektový návrh (HTN) plánovacích domén

propracované GUI, vizualizace

Literatura

- Malik Ghallab, Dana Nau, Paolo Traverso: ***Automated Planning:*** *theory and practice*. Elsevier, 2004.
- Kutluhan Erol, James Hendler, Dana Nau: ***Semantics for Hierarchical Task Network Planning***. Technická zpráva, University of Maryland, 1994.
- Kutluhan Erol, James Hendler, Dana Nau: ***Complexity Results for Hierarchical Task Network Planning***. Technická zpráva, University of Maryland, 1994.
- Dana Nau et al.: ***SHOP2: An HTN Planning System***. Journal of Artificial Intelligence Research, 379-404, 2003.
- Amnon Lotem, Dana Nau, James Hendler: ***Using planning graphs for solving HTN planning problems.*** Sborník AAAI/IAAI-99, 534-540, 1999.