Poznámky z přednášek
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# Umělá inteligence I
# BONUS

Peter Černo, 2010
petercerno@gmail.com

**Garant**: doc. RNDr. Roman Barták, Ph.D.
**E-mail**: Roman.Bartak@mff.cuni.cz
**Domácí stránka**: `http://ktiml.ms.mff.cuni.cz/~bartak/`

**Anotace**: Úvodní přednáška představující základní pojmy a metody různých oblastí umělé inteligence.

**Sylabus**:

1. Inteligentní agenti, jejich prostředí a základní struktury.
2. Řešení úloh prohledáváním (DFS, BFS, ID, A*, IDA*, lokální a on-line prohledávaní, heuristiky).
3. Splňování omezujících podmínek.
4. Hry (minimax, alfa-beta prořezávání).
5. Reprezentace znalostí v logice, logické odvozovací techniky (dopředné a zpětné řetězení, rezoluční metoda).
6. Automatické plánování.

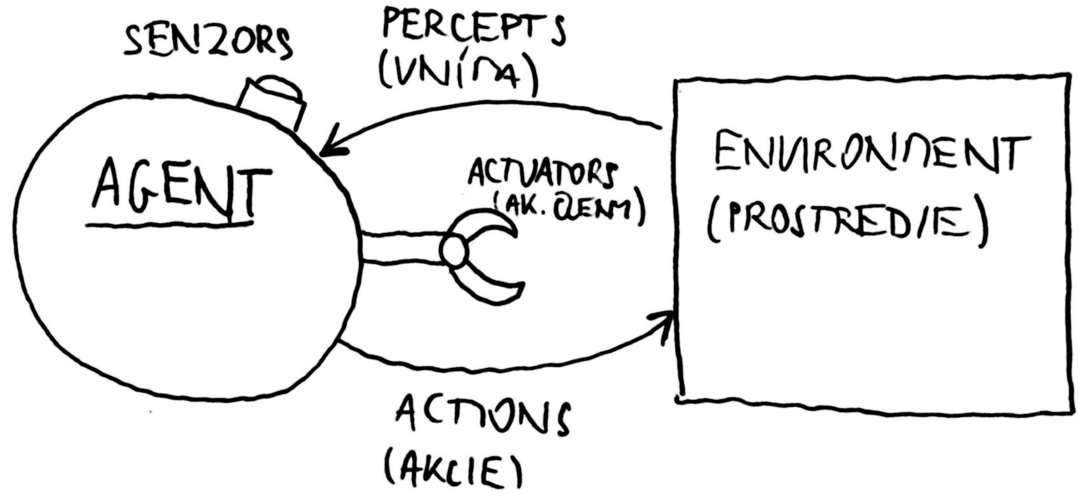
**Cíl předmětu**: Představit základní pojmy a metody různých oblastí umělé inteligence.

**Literatura**:


1. S. Russell, P. Norvig: Artificial Intelligence; A Modern Approach, 2003
2. V. Mařík, O. Štepánková, J. Lažanský a kol.: Umělá Inteligence (1). Academia, Praha
3. V. Mařík, O. Štepánková, J. Lažanský a kol.: Umělá Inteligence (2). Academia, Praha
4. Z. Renc: Vybrané partie z umělé inteligence. Skriptum MFF UK Praha, 1987

26.01.2009 Mo
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I




AGENTOVÁ FUNKCIA (ABSTR. MAT. POPIS, POMOCOU TABUĽKY)
$V^* \rightarrow A$, V = MN. VNEMOV, A = MN. AKCIÍ

AGENTOVÝ PROGRAM = KONKRÉTNA IMPLEM. AG. FCIE.

MIERA VÝKONU AGENTA - URČENÉ PODĽA TOHO, AKO CHCEME, ABY MERALO PROSTREDIE

RACIONÁLNY AGENT PRE KAŽDÚ POSTUPNOSŤ VNEMOV ZVOLÍ AKCIU, MAXIMALIZUJÚCU OČAKÁVANÚ MIERU VÝKONU (NA ZÁKLADE VNEMOV + VNÚT. ZNALOSTÍ AGENTA)

## VLASTNOSTI PROSTREDIA

1. PLNÁ / ČIASTOČNÁ POZOROVATEĽNOSŤ
2. DETERMINISTICKÉ / STOCHASTICKÉ / STRATEGICKÉ (INÝ AGENT)
3. EPIZODICKÉ / SEKVENČNÉ
4. STATICKÉ / DYNAMICKÉ / SEMI-DYNAMICKÉ (MIERA VÝKONU)
5. DISKRÉTNE / SPOJITÉ
6. JEDEN AGENT / MULTI-AGENT

AGENT = ARCHITEKTÚRA + PROGRAM

VÝPOČTOVÉ ZARIADENIE S FYZ. SENZORMI + AKČ. ČLENMI

IMPL. AGEN. FCIE

## TABUĽKOVÝ AGENT TABLE-DRIVEN-AGENT

function TABLE-DRIVEN-AGENT (percept) returns an action
  APPEND percept TO THE END OF percepts
  return LOOKUP (percepts, table)

## AGENT S REFLEXAMI SIMPLE-REFLEX-AGENT

AKCIA VYBRANÁ IBA NA ZÁKLADE SÚČASNÉHO VNEMU
IMPLEM.: FORMOU PRAVIDIEL PODMIENKA → AKCIA
FUNGUJE IBA PRE PLNE POZOROVATEĽNÉ PROSTREDIE

## AGENT S MODELOM REFLEX-AGENT-WITH-STATE

PROBLÉM S ČIASTOČNOU POZOROV. → VNÚTORNÝ STAV
(AKO SA SVET VYVÍJA + AKO VPLÝVAJÚ AG. AKCIE) → MODEL SVETA

## AGENT S CIEĽOM

UVAŽOVANIE O BUDÚCNOSTI, PLÁNOVANIE

## AGENT SO ŽIADOSTIAMI

MAPOVANIE STAVOV NA REÁLNE ČÍSLA : MIERA ŽIADANOSTI
AGENT SA PRI VÝBERE AKCIE RIADI AJ TÝM, AKÝ JE PRÍPADNÝ DOSIAHNUTEĽNÝ STAV ŽIADÚCI

## AGENT UČIACI SA

(1) VÝKONNÝ PRVOK - PÔVODNÝ AGENT, KT. VYBERÁ AKCIE
(2) UČIACI SA PRVOK - VYLEPŠUJE PROGRAM PÔVODNÉHO AGENTA
(3) KRITIKA - SPÄTNÁ VÄZBA, AKO DOBRE SA AGENT CHOVÁ
(4) GENERÁTOR PROBLÉMOV - NAVRHUJE AKCIE, KTORÉ BY MOHLI VIESŤ K NOVEJ SKÚSENOSTI

26.01.2009 n.
PETER ČERNO

# NAIL 069 UMELÁ INTELIGENCIA I

## AGENT S CIEĽOM → AGENT RIEŠIACI ÚLOHY

```
function SIMPLE-PROBLEM-SOLVING-AGENT (percept) returns an action
  state ← UPDATE-STATE (state, percept)
  if sequence is empty then
      goal ← FORMULATE-GOAL (state)
      problem ← FORMULATE-PROBLEM (state, goal)
      sequence ← SEARCH (problem)
  action ← FIRST(sequence)
  sequence ← REST(sequence)
  return action
```

## FORMULÁCIA CIEĽA (FORMULATE-GOAL)

AKÝ JE ŽIADÚCI STAV SVETA ?

## FORMULÁCIA ÚLOHY (FORMULATE-PROBLEM)

(1) POČIATOČNÝ STAV
(2) POPIS MOŽNÝCH AKCIÍ → STAVOVÝ PRIESTOR
SUCCESSOR-FN (state) ⇝ (action, new-state)
(3) TEST CIEĽA
(4) CENA CIEST

RIEŠENIE ÚLOHY : CESTA Z POČ. STAVU DO CIEĽOVÉHO
OPTIMÁLNE RIEŠENIE ÚLOHY : CESTA -II- S MINI. CENOU

→ JE NUTNÁ SPRÁVNA ABSTRAKCIA

## PREHĽADÁVANIE STAVOVÉHO PRIESTORU

TREE-SEARCH (problem, strategy) return < solution / failure
    initialize ST using the initial state of problem
    loop
        if no candidates for expansion then return failure
        choose a leaf node (by strategy)
        if the node contains a goal state then return solution
        else expand node

$\boxed{\text{node} \neq \text{state}}$

UZOL (node) PREHĽADÁVAJÚCEHO STROMU (ST) OBSAHUJE:

(1) AKTUÁLNY STAV (state)
(2) ODKAZ NA RODIČA
(3) AKCIU, KTORÁ VIEDLA OD RODIČA
(4) CENU $g(n)$ CESTY Z KOREŇA
(5) HĹBKU (POČET KROKOV OD KOREŇA)

OKRAJ (FRINGE) = SÚBOR UZLOV PRE EXPANDOVANIE (= LISTY)
MOŽNO REPREZENTOVAŤ POMOCOU FRONTY

## VÝKON ALGORITMU:

(1) ÚPLNOSŤ
(2) OPTIMALITA
(3) ČASOVÁ ZLOŽITOSŤ
(4) PRIESTOROVÁ ZLOŽITOSŤ

VETVIACI FAKTOR $b$ = MAXIMÁLNY POČET NÁSLEDNÍKOV
HĹBKA $d$ = HĹBKA NAJMENEJ ZANORENÉHO CIEĽ. UZLU
CESTA $m$ = DĹŽKA MAX. CESTY

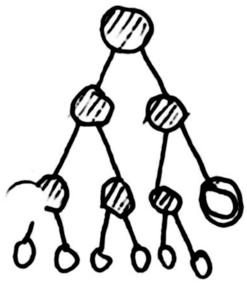
CENA PREHĽADÁVANIA = KOĽKO ČASU & PRIESTORU POTREBUJEM
CELKOVÁ CENA = KOMBINUJE CENU PREHĽ. & CENU NÁJ. RIEŠENIA

26.01.2009 Mo
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## BREADTH-FIRST SEARCH (BFS)

Z OKRAJA BERIEME UZOL V NAJMENŠEJ HĹBKE (FIFO)
ÚPLNÁ, NEMUSÍ BYŤ OPTIMÁLNA ! (IBA AK CENA NEKLESÁ S HĹBKOU)
ČASOVÁ ZLOŽITOSŤ : CIEĽ V HĹBKE $d$ NA KONCI VRSTVY



$$1 + b + b^2 + \ldots + b^d + b(b^d - 1) = O(b^{d+1})$$

PAMÄŤOVÁ ZLOŽITOSŤ = $O(b^{d+1})$
(PREDSTAVUJE VÄČŠÍ PROBLÉM)

## UNIFORM-COST SEARCH DO ŠÍRKY PODĽA CENY

AKO PRVÝ EXPANDUJEME UZOL S NAJMENŠOU CENOU (HALDA)
POZOR NA KROKY S NULOVOU CENOU ! (CYKLY)
ÚPLNOSŤ A OPTIMÁLNOSŤ MOŽNO GARANTOVAŤ AK JE CENA KAŽDEJ AKCIE $\geq \varepsilon > 0$.
ČASOVÁ A PAMÄŤOVÁ ZLOŽITOSŤ : $O(b^{1 + \lfloor C^*/\varepsilon \rfloor})$

## DEPTH-FIRST SEARCH (DFS)

VŽDY EXPANDUJEME NAJHLBŠÍ UZOL (LIFO, REKURZÍVNE)
NIE JE ÚPLNÝ ANI OPTIMÁLNY !!!
ČASOVÁ ZLOŽITOSŤ : $O(b^m)$ (ČASTO $d << m$)
PAMÄŤOVÁ ZLOŽITOSŤ : $O(bm)$
STAČÍ SI PAMÄTAŤ JEDNU CESTU Z KOREŇA SPOLOČNE SO SUSEDNÝMI UZLAMI
EXPANDOVANÝ UZOL S PRESKÚMANÝMI POTOMKAMI MOŽNO ZAHODIŤ.
MOŽNO ZLEPŠIŤ BACKTRACKINGOM NA $O(m)$

ÚPLNOSŤ AK $l \geq d$ !
OPTIMALITA: NIE

# DEPTH LIMITED SEARCH (DLS)

function DEPTH-LIMITED-SEARCH (problem, limit) returns { solution, failure, cutoff }
    RECURSIVE-DLS (MAKE-NODE (INITIAL-STATE[problem]), problem, limit )

function RECURSIVE-DLS (node, problem, limit) returns { solution, failure, cutoff }

```
1   cutoff-occured? ← false
2   if GOAL-TEST[problem] (STATE [node]) then return SOLUTION (node)
3   else if DEPTH [node] = limit then return cutoff
4   else foreach successor in EXPAND (node, problem) do
5       result ← RECURSIVE-DLS(successor, problem, limit)
6       if result = cutoff then cutoff-occured ? ← true
7       else if result ≠ failure then return result
8   if cutoff-occured? then return cutoff
9   else return failure
```

AK ALG. NENÁJDE solution, POTOM VRÁTI NA node:
(1) cutoff $\Leftrightarrow$ ASPOŇ JEDEN UZOL MAL HĹBKU limit
(2) failure $\Leftrightarrow$ VŠETKY UZLY MALI HĹBKU < limit

ČASOVÁ ZLOŽITOSŤ : $O(b^l)$
PRIESTOROVÁ ZLOŽITOSŤ : $O(bl)$

## AKO FUNGUJE RECURSIVE-DLS

MÁME UZOL node. ROZLÍŠIME DVA PRÍPADY

(1) HĹBKA UZLU = limit
AK V 2. GOAL-TEST = true → MÁME RIEŠENIE → NÁVRAT NA RIADOK 5, NA RIADKU 7 SA ZNOVU VYNÁRAME S RIEŠ.
AK V 2. GOAL-TEST = false → CUTOFF (T.J. MOŽNO PREHĹBOVAŤ) → NÁVRAT NA RIADOK 5, V 6. RIADKU cutoff-occured ← true
⇒ AK V TEJTO VETVE NENÁJDEME RIEŠ., VRACIAME cutoff V 8

(2) HĹBKA UZLU < limit
AK V 2. GOAL-TEST = false → EXPANDUJEME. AK SA NENÁJDE RIEŠENIE A VYSKYTNE SA CUTOFF, POTOM MÔŽME VETVU PREHĹBOVAŤ → VRACIAME CUTOFF. INAK failure

26.01.2009 Mô
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

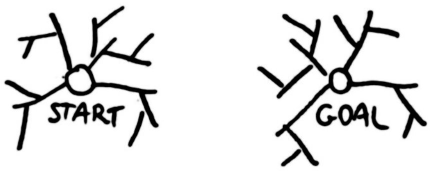
## ITERATIVE DEEPENING DFS (IDS)

function ITERATIVE-DEEPENING-SEARCH (problem) returns {solution / failure}
  for depth ← 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH (problem, depth)
    if result ≠ cutoff then return result

- ÚPLNOSŤ, OPTIMALITA (AK CENA NEKLESÁ S HĹBKOU)
  MALÉ PAMÄŤOVÉ NÁROKY $O(bd)$
  ČASOVÁ ZLOŽITOSŤ:
  $$d \cdot b^1 + (d-1) b^2 + \ldots + 1 \cdot b^d = O(b^d)$$

IDS JE PREFEROVANÁ METÓDA SLEPÉHO PREHĽADÁVANIA VO VEĽKOM PRIESTORE S NEZNÁMOU HĹBKOU RIEŠENIA !

## BIDIRECTIONAL SEARCH (BiS)




- PREČO? $b^{d/2} + b^{d/2} \ll b^d$

⇒ PRED EXPANZIOU UZLA VŽDY SKONTROLUJEME, ČI NIE JE V OKRAJI DRUHÉHO PREHĽ. STROMU

⇒ POTREBUJEME MAŤ V PAMÄTI ASPOŇ JEDEN ZO STROMOV $O(b^{d/2})$

⇒ PRI POUŽITÍ BFS NA OBOCH STRANÁCH MÁME GARANCIU ÚPLNOSTI AJ OPTIMALITY !!!

PROBLÉMY: VIAC CIEĽOVÝCH STAVOV,
  IMPLICITNE DANÉ CIEĽOVÉ STAVY.
  AKO DEFINOVAŤ PREDCHODCU STAVU

# GRAPH - SEARCH

PAMÄTÁME SI UŽ EXPANDOVANÉ STAVY → TZV. <u>UZAVRENÉ STAVY</u>
AK SA DO UZAVRENÉHO STAVU DOSTANEME ZNOVA,
ĎALEJ HO V PREHĽ. NEEXPANDUJEME

```
function GRAPH-SEARCH (problem, fringe (strategy)) returns { solution / failure
  closed ← ∅
  insert MAKE-NODE (INITIAL-STATE [problem]) to fringe
  loop
    if fringe is empty then return failure
    node ← remove from fringe
    if GOAL-TEST [problem] (STATE [node]) then return SOLUTION (node)
    if STATE [node] is not in closed then
      add STATE [node] to closed
      insert EXPAND (node, problem) to fringe
```

ČO AK AGENTOVI CHÝBAJÚ INFORMÁCIE?

<u>CONFORMANT PROBLEMS</u> - PRACUJEME S DOMNELÝM (BELIEF) STAVOM = MNOŽINA REÁLNYCH STAVOV A HĽADÁME BELIEF STAV OBSAHUJÚCI IBA CIEĽOVÉ STAVY

<u>CONTINGENCY PROBLEMS</u> - PLÁNY S ALTERNATÍVNYMI VETVAMI

<u>EXPLORATION PROBLEMS</u> - RIEŠENIE PRIESKUMOM

| ZHRNUTIE (SKRATKY) | ÚPLNOSŤ | OPTIMALITA | ČAS | PRIESTOR |
|---|---|---|---|---|
| BFS (BREADTH-FIRST) | ÁNO | ÁNO, AK CENA NEKLESÁ S HĹBKOU!! | $b^{d+1}$ | $b^{d+1}$ |
| UNIFORM-COST | ÁNO | ÁNO, AK $\exists \varepsilon > 0$ | $b^{C^*/\varepsilon}$ | $b^{C^*/\varepsilon}$ |
| DFS (DEPTH-FIRST) | NIE | NIE | $b^m$ | $bm$ |
| DLS (DEPTH-LIMITED) | ÁNO, AK $l \geq d$ | NIE | $b^l$ | $bl$ |
| IDS (ITERATIVE-DEEPENING) | ÁNO | ÁNO, AK CENA NEKLESÁ S HĹBKOU!! | $b^d$ | $bd$ |
| BiS (BIDIRECTIONAL) | ÁNO, AK BFS | ÁNO, AK BFS | $b^{d/2}$ | $b^{d/2}$ |

26.01.2009 r.
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## INFORMOVANÉ PREHĽADÁVANIE, HEURISTIKY (PROBLÉMOVO ZÁVISLÉ INFO)

EVALUAČNÁ FUNKCIA $f(n)$ .. OHODNOCUJE UZOL $n$ NA ZÁKLADE DĹŽKY CESTY DO CIEĽA PRECHÁDZAJÚCEJ CEZ $n$

HEURISTICKÁ FUNKCIA $h(n)$ .. ODHADUJE DĹŽKU NAJKRATŠEJ CESTY Z $n$ DO CIEĽA

$h(n) = 0 \Leftrightarrow n$ JE CIEĽOVÝ STAV !

## GREEDY BEST-FIRST SEARCH BFS (HLADOVÝ ALGORITMUS PREHĽADÁVANIA S CENOU)

$\boxed{f(n) = h(n)}$, PREFERUJEME UZOL, KT. JE NAJBLIŽŠIE K CIEĽU

NEGARANTUJE OPTIMÁLNE RIEŠ. ANI NÁJDENIE RIEŠENIA !

ČAS: $O(b^m)$, PAMÄŤ: $O(b^m)$

## ALGORITMUS A*

$\boxed{f(n) = g(n) + h(n)}$

ASI NAJZNÁMEJŠÍ ALGORITMUS PREHĽADÁVANIA S CENOU

PRÍPUSTNÁ HEURISTIKA: $h(n) \leq$ NAJMENŠIA CESTA Z $n$ DO CIEĽA

MONOTÓNNA (KONZISTENTNÁ) HEURISTIKA:

$h(n) \leq c(n, a, n') + h(n')$

MONOTÓNNA HEURISTIKA JE PRÍPUSTNÁ + HODNOTY $f(n)$ NEKLESAJÚ !

AK $g(n') = g(n) + c(n, a, n')$

$\Rightarrow f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n)$

(1) AK JE $h(n)$ PRÍPUSTNÁ, POTOM JE A* VRÁMCI TREE-SEARCH OPTIMÁLNY

(2) AK JE $h(n)$ MONOTÓNNA, POTOM JE A* VRÁMCI GRAPH-SEARCH OPTIMÁLNY !

DŌKAZ (1): PRVÝ EXPANDOVANÝ CIEĽOVÝ UZOL JE OPTIMÁLNY

NECH $G_2$ JE SUBOPTIMÁLNY UZOL Z OKRAJU, $C^*$ OPT. CENA

$$f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$$

NECH $n$ JE UZOL Z OKRAJU, KTORÝ JE NA OPT. CESTE

$$f(n) = g(n) + h(n) \leq C^* \quad \boxed{\text{PODĽA PRÍPUSTNOSTI } h}$$

$$\Rightarrow f(n) \leq C^* < f(G_2) \quad \square$$

DŌKAZ (2):

PRE MON. HEURISTIKU SÚ $f(n)$ POZDĹŽ ĽUB. CESTY NEKLESAJÚCE

$A^*$ UZAVRE UZOL S NAJMENŠÍM $f(n)$

$\Rightarrow$ NEEXISTUJE ŽIADNA KRATŠIA CESTA DO $n$ NEŽ PRÁVE TÁ NÁJDENÁ...

$\Rightarrow$ PRVÝ ZAVRENÝ CIEĽOVÝ UZOL MUSÍ BYŤ OPTIMÁLNY $\square$

$A^*$ EXPANDUJE VŠETKY UZLY, KDE $f(n) < C^*$, PO VRSTEVNICIACH

MÔŽE EXPANDOVAŤ NIEKTORÉ UZLY, KDE $f(n) = C^*$

UZLY, KDE $f(n) > C^*$ NIE SÚ EXP.

$\Rightarrow$ VŽDY NÁJDE OPTIMÁLNE RIEŠENIE

AK PLATÍ $|h(n) - h^*(n)| \leq O(\log h^*(n))$

$\Rightarrow$ POLYNOMIÁLNE VEĽA EXPANZIÍ

$A^*$ MÁ PROBLÉM S PAMÄŤOU !

27.01.2009 Tu
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## ITERATIVE-DEEPENING A* (IDA*)

JEDNODUCHÝ SPÔSOB AKO ZMENŠIŤ PAMÄŤOVÉ NÁROKY

```
function IDA* (problem) returns { solution
                                  failure
   root ← MAKE-NODE (INITIAL-STATE[problem])
   f-limit ← f-COST (root)
   loop
      solution, f-limit ← DFS-CONTOUR (root, f-limit)
      if solution ≠ ∅ then return solution
      if f-limit = ∞ then return failure

function DFS-CONTOUR (node, f-limit)
   if f-COST [node] > f-limit then return null, f-COST[node]
   if GOAL-TEST.. = true then return SOLUTION (node), f-limit
   next-f ← ∞
   foreach s in SUCCESSORS (node)
      solution, new-f ← DFS-CONTOUR (s, f-limit)
      if solution ≠ ∅ then return solution, f-limit
      next-f ← MIN (next-f, new-f)
   return null, next-f
```

PRE NASLEDUJÚCU ITERÁCIU SA POUŽIJE LIMIT NAJMENŠIE f(n) UZLU n, KTORÝ V PREDCH. ITERÁCII PREKROČIL LIMIT

# RECURSIVE BEST-FIRST SEARCH (RBFS)

function RECURSIVE-BEST-FIRST-SEARCH (problem) $\left\{ \begin{array}{l} \text{solution} \\ \text{failure} \end{array} \right.$

RBFS (problem, MAKE-NODE (INITIAL-STATE [problem]), $+\infty$)

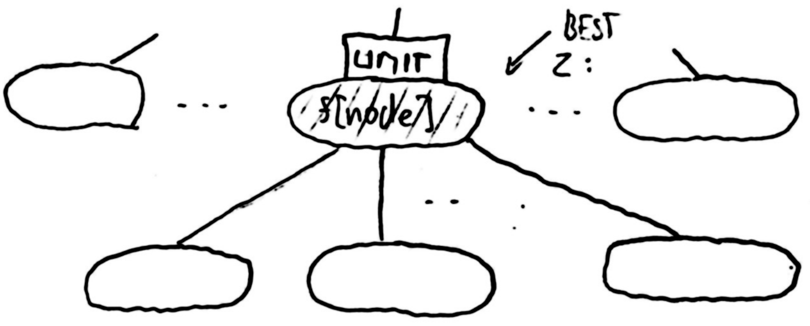
function RBFS (problem, node, f-limit) $\left\{ \begin{array}{l} \text{solution} \\ \text{failure} \end{array} \right.$, new f-cost

```
1   if GOAL-TEST ... then return SOLUTION (node), f[node] = g(node)
2   successors ← EXPAND (node, problem)
3   if successors is empty then return failure, +∞
4   foreach s in successors
5       f[s] ← max (g(s) + h(s), f[node])          !!
6   loop
7       best ← lowest f-value node in successors
8       if f[best] > f_limit then return failure, f[best]
9       alternative ← second-lowest f-value
10      result, f[best] ← RBFS (problem, best, min (f-limit, alternative))
11      if result ≠ failure then return result, f[best]
```

INVARIANT: PRED VSTUPOM DO 1 PLATÍ f[node] ≤ f-limit

DK: PLATÍ NA ZAČIATKU + V KROKU 10 JE f[best] ≤ f-limit, alternative

CYKLUS 6-11 JE KONEČNÝ, PRETOŽE BUĎ V 10 NÁJDE solution, ALEBO NENÁJDE A ZVÝŠI f[best], PRIČOM PRI VNORENÍ NA RIADKU 10 JE f[best] > min (f_limit, alternative)




KEĎ JE h PRÍPUSTNÁ
⟹ OPTIMÁLNY

PAMÄŤ: $O(bd)$

ČAS: EXPONENCIÁLNY

27.01.2009 Tu
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## SIMPLIFIED MEMORY-BOUNDED A* (SMA*)

KEĎ VYČERPÁME PAMÄŤ, VYHODÍME NAJMENEJ SĽUBNÝ LIST (TJ. S NAJVÄČŠOU CENOU $f(n)$)
TÚTO CENU SI ZAPAMÄTÁME U RODIČA

0 VYTVORÍM FRONTU Q POZOSTÁVAJÚCU Z POČ. UZLU
V CYKLE:

1 AK JE FRONTA PRÁZDNA → RETURN FAILURE
2 NÁJDEME V Q NAJHLBŠÍ UZOL S NAJMENŠOU CENOU: n
3 AK JE n CIEĽOVÝ → RETURN SOLUTION
4 VYTVORÍME ĎALŠIEHO NÁSLEDNÍKA s UZLU n
5 AK s NIE JE CIEĽ A JE V NAJVÄČŠEJ HĹBKE ⇒ $f(s) \leftarrow \infty$
INAK $f(s) \leftarrow \max(f(n), g(s)+h(s))$
6 AK SME UŽ VYGENEROVALI ∀ NÁSLEDNÍKOV n,
POTOM UPDATEUJEME $f(n)$, PRÍP. PREDCHODCOV
7 KEĎ SÚ VŠETCI NÁSLEDNÍCI n V PAMÄTI, VYHODÍME n Z Q
8 AK JE PAMÄŤ PLNÁ, POTOM
NÁJDEME NAJPLYTŠÍ UZOL S NAJVYŠŠOU CENOU
ODSTRÁNIME HO AKO POTOMKA SVOJHO RODIČA
RODIČA VLOŽÍME DO FRONTY
(U RODIČA SI POZNAMENÁME JEHO f-HODNOTU)
9 VLOŽÍME s DO FRONTY Q

AK JE PAMÄŤ DOSTATOČNÁ PRE OPTIMÁLNU CESTU,
SMA* NÁJDE OPT. RIEŠENIE
INAK NÁJDE NAJLEPŠIE MOŽNÉ RIEŠ. S DOSTUPNOU PAM.

## EFEKTÍVNY VETVIACI FAKTOR $b^*$ (VÝKON HEURISTIKY)

NECH ALG. POTREBUJE N UZLOV, RIEŠENIE V HĹBKE $d$
$\Rightarrow N+1 = 1 + b^* + \dots (b^*)^d$

$\forall n \quad h_2(n) \geq h_1(n) \quad h_2$ DOMINUJE NAD $h_1$
$A^*$ EXPANDUJE $\forall$ UZLY KDE $f(n) < C^*$, T.J. $h(n) < C^* - g(n)$
$\Rightarrow h_2$ JE LEPŠIA HEURISTIKA

## RELAXÁCIA PROBLÉMU

ZJEDNODUŠENIE, RIEŠENIE PÔVODNÉHO PROBLÉMU JE AJ RIEŠENÍM RELAX. PROBLÉMU

CENA OPT. RIEŠ. RELAX. PROBLÉMU JE DOLNÝ ODHAD

## DATABÁZA VZOROV (PATTERN DATABASE)

AK MÁME VIAC HEURISTÍK $\rightarrow$ MAXIMUM

## LOKÁLNE PREHĽADÁVANIE

NIE JE DÔLEŽITÁ CESTA, IBA CIEĽ
V KAŽDOM KROKU TROCHU POZMENÍME STAV
OBJEKTÍVNA FUNKCIA

27.01.2009 TH
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## HILL-CLIMBING

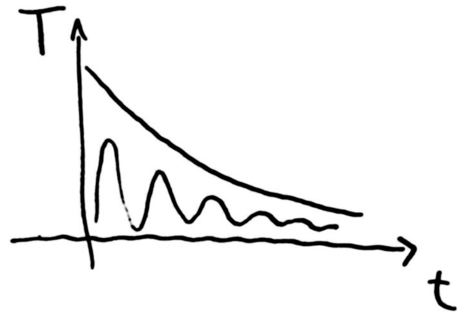
Z OKOLIA DANÉHO STAVU VŽDY VYBERIEME STAV S NAJLEPŠOU OBJEKTÍVNOU FUNKCIOU

PROBLÉMY - LOKÁLNE OPTIMÁ, HREBENE, PLOŠINY (PLATEAU)

STOCHASTICKÝ HC (RULETA MEDZI ZLEPŠUJÚCIMI)
HC PRVEJ VOĽBY (NÁHODNE GEN. SUSEDOV, PRVÝ ZLEPŠUJÚCI)
HC S NÁHODNÝMI REŠTARTMI (ÚNIK Z LOK. OPTÍM)

## SIMULATED ANNEALING (SIMULOVANÉ ŽÍHANIE)

KOMBINUJE VÝHODY HC A RANDOM WALK




```
loop t++
  T ← schedule[t]
  if T=0 then return current
  next ← RANDOM SUCCESSOR !!
  ΔE ← F[next] - F[current]
  if ΔE > 0 then current ← next
  else current ← next WITH PROBABILITY
```
$\boxed{e^{\Delta E / T}}$

## LOCAL BEAM SEARCH (PAPRSKY)

k NÁHODNÝCH STAVOV
NÁJDE ∀ SUSEDOV A VYBERIE Z NICH k NAJLEPŠÍCH

## GENETICKÉ ALGORITMY

FITNESS, SELECTION, CROSS-OVER, MUTATION

## ZHRNUTIE (SKRATKY)

| | |
|---|---|
| BFS (BEST-FIRST SEARCH) | $f(n) := h(n)$, HLADOVÝ ALG. PREHĽADÁVANIA S CENOU $O(b^m)/O(b^m)$, NIE JE ÚPLNÝ ANI OPTIMÁLNY |
| A* | $f(n) := g(n) + h(n)$. $h(n)$ PRÍP. ⇒ TREE-SEARCH OPT., $h(n)$ KONZ. ⇒ GRAPH-SEARCH OPT. PROBLÉM S PAMÄŤOU |
| IDA* (ITERATIVE-DEEPENING A*) | ITEROVANÉ PREHĽBOVANIE - f-limit NA $f(n)$ ZMENŠENIE PAM. NÁROKOV |
| RBFS (RECURSIVE BEST-FIRST SEARCH) | REKURZÍVNA VERZIA PREHĽ. S CENOU $h(n)$ PRÍP. ⇒ OPTIM., PAMÄŤ: $O(bd)$ |
| SMA* (SIMPLIFIED MEMORY-BOUNDED A*) | AK JE PAMÄŤ DOSTATOČNÁ PRE OPT. CESTU ⇒ NÁJDE OPT. |
| HC (HILL-CLIMBING) | LOK. PREHĽ., METÓDA NAJVÄČ. STÚPANIA STOCHASTICKÝ HC / HC 1. VOĽBY / HC S NÁH. REŠTARTMI |
| SA (SIMULATED ANNEALING) | SIMULOVANÉ ŽÍHANIE ZHORŠENIE S PRST. $e^{\Delta E/T}$  |
| (L)BS (LOCAL BEAM SEARCH) | LOKÁLNE PAPRSKY |
| GA (GENETIC ALGORITHMS) | GENETICKÉ ALGORITMY |

## OFFLINE PREHĽADÁVANIE

NAJSKÔR NÁJDE KOMPLETNÉ RIEŠENIE, POTOM HO APLIKUJE

## ONLINE PREHĽADÁVANIE

VYBERIE AKCIU → VYKONÁ AKCIU → ZISTÍ AKO VYZERÁ SVET

DYNAMICKÉ / SEMI-DYN. PROSTREDIE

ACTIONS(s) = ZOZNAM AKCIÍ APLIK. NA STAV s

$c(s, a, s')$ = CENA 1 KROKU (AŽ KEĎ AGENT POZNÁ s')

GOAL-TEST(s)

PREDPOKLADÁME, ŽE: AGENT ROZPOZNÁVA UŽ NAVŠTÍVENÉ STAVY

POUŽÍVA DETERM. AKCIE

MÁ PRÍP. HEURISTIKU $h(n)$

KOMPETITÍVNY POMER, SLEPÉ ULIČKY, BEZPEČNE PREZKÚM. SVETY

! ONLINE ALG. NEMÔŽU PRESKOČIŤ DO ÚPLNE INÉHO STAVU BEZ ABSOLVOVANIA CELEJ CESTY

27.01.2009 Tu
PETER CERNO

# NAIL069 UMELÁ INTELIGENCIA I

## ONLINE DFS

```
function ONLINE-DFS-AGENT ( s ) returns an action
  [r, a = PREVIOUS STATE AND ACTION, INITIALLY NULL]
  if GOAL-TEST ( s ) then stop
  if s is a new state then unexplored[s] ← ACTIONS (s)
  if r ≠ NULL then
      result [a, r] ← s
      ADD r TO unbacktracked [s]
  if unexplored[s] is empty then
      if unbacktracked [s] is empty then stop
      else a ← AN ACTION b S.T. result [b, s] =
                   = POP ( unbacktracked [s] )
  else a ← POP (unexplored [s] )
  r ← s
  return a
```

$r \xrightarrow{a} s$

# ONLINE LOK PREHĽADÁVANIE

NAPR. HC. POZOR! NEMÔŽEME POUŽIŤ RESTART !!
RIEŠENIE : NÁHODNÁ PRECHÁDZKA (NEEFEKT.)

# LEARNING REAL-TIME A*

UČÍ SA, KAM HO DANÁ AKCIA DOVEDIE (result)
A AKÁ JE VZDIALENOSŤ DO CIEĽA (H)

function LRTA*-AGENT (s) returns an action
  [r, a = PREVIOUS STATE AND ACTION, INITIALLY NULL]
  if GOAL-TEST (s) then stop
  if s IS A NEW STATE (NOT IN H) then $H[s] \leftarrow h(s)$
  if $r \neq$ NULL then
    result $[a, r] \leftarrow s$
    $H[r] \leftarrow \min_{b \in \text{ACTIONS}(r)}$ LRTA*-COST (r, b, result[b, r], H)
  $a \leftarrow$ AN ACTION b IN ACTIONS(s) THAT
    MINIMIZES LRTA*-COST (s, b, result [b, s], H)

function LRTA*-COST (r, a, s, H) returns a cost estimate
  if s is undefined then return h(r)
  else return $c(r, a, s) + H[s]$

28.01.2009 We
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## CSP (CONSTRAINT SATISFACTION PROBLEM)

KONEČNÁ MNOŽINA PREMENNÝCH
DOMÉNY - KONEČNÉ MNOŽINY PRE PREMENNÉ
KONEČNÉ MNOŽINY PODMIENOK

STAV = PRIRADENIE HODNÔT DO (NIEKT.) PODM.
KONZISTENTNÝ STAV
ÚPLNÝ STAV
OBJEKTÍVNA FCIA.

## BACKTRACKING

ZÁKLADNÝ NEINFORMOVANÝ ALG. PRE RIEŠENIE CSP
AKO ZLEPŠIŤ EFEKTIVITU?

(1) VOĽBOU PREMENNEJ - VARIABLE ORDERING
(a) NAJVIAC OBMEDZENÁ = DOM HEURISTIKA
(b) S NAJVIAC PODMIENKAMI = DEG HEURISTIKA
→ PRAVIDLÁ PRVÉHO NEÚSPECHU (FAIL-FIRST)

(2) VÝBER HODNOTY - VALUE ORDERING
PRAVIDLO PRVÉHO ÚSPECHU (SUCCEED-FIRST)
(POUŽÍVAJÚ SA PROBLÉMOVO ZÁVISLÉ HEURISTIKY)

(3) VYUŽITIE ŠTRUKTÚRY PROBLÉMU

## FORWARD CHECKING

PO PRIRADENÍ → MRADIŤ NEKOREKTNÉ HODNOTY Z DOMÉN VOĽNÝCH PREMENNÝCH, SPOJENÝCH PODMIENKOU S AKT. PREMENNOU

## CONSTRAINT PROPAGATION (=LOOK AHEAD)

REALIZUJE SA METÓDOU UDRŽOVANIA KONZISTENCIE PODM.

## ARC CONSISTENCY (HRANOVÁ KONZISTENCIA)

SMEROVÉ ODSTRÁNENIE HODNÔT Z DOMÉNY PREM. X, KTORÉ NEMAJÚ PODPORU V DOMÉNE PREM. Y

FILTRÁCIU JE TREBA ZAVOLAŤ VŽDY KEĎ SA ZMENÍ DOMÉNA Y

## ALGORITMUS AC-3

function AC-3 (csp)

- MOŽNO VOLAŤ INKREMENTÁLNE: AK JE INŠTANCIOVANÁ $X_i$, PRIDÁME DO Q $\forall (X_k, X_i)$
- $\forall (X_i, X_j)$ DÁME DO Q

while queue $\neq \emptyset$ do
    $(X_i, X_j) \leftarrow$ REMOVE-FIRST (queue)
    if RM-INCONSISTENT-VALUES $(X_i, X_j)$
        add $(X_k, X_i)$ to queue (for all such constraints except $(X_j, X_i)$ !!!)

ČASOVÁ ZLOŽITOSŤ: $O(ed^3)$
  e .. POČET PODMIENOK
  d .. VEĽKOSŤ DOMÉNY

NIE JE OPTIMÁLNY (OPAKOVANE SA TESTUJÚ ROVNAKÉ DVOJICE HODNÔT). MOŽNO UROBIŤ OPT. V $O(ed^2)$ AC-4 ...

28.01.2009 We
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

**k-KONZISTENCIA**: PRE KONZISTENTNÉ HODNOTY (k-1) PREMENNÝCH MOŽNO NÁJSŤ KONZ. HODNOTU ĽUBOVOĽNEJ ĎALŠEJ PREMENNEJ

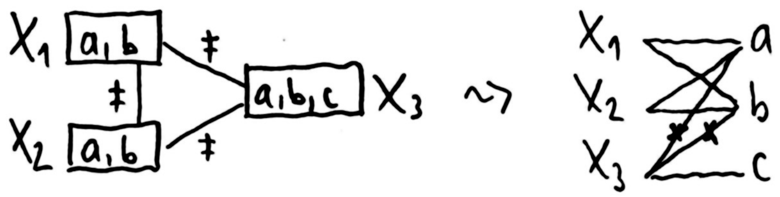
AC = 2-KONZ., PC (PATH CONS.) = 3-KONZ.

ZLOŽITOSŤ: EXPONENCIÁLNA V k

## GLOBÁLNE PODMIENKY

ŠPECIÁLNY FILTRAČNÝ ALGORITMUS NA ZAISTENIE GLOBÁLNEJ KONZISTENCIE

NAPR. all-different




ODSTRÁNI HRANY, KT. NIE SÚ SÚČASŤOU ŽIADNEHO MAX. PÁROVANIA → ODSTR. PRÍSLUŠNÉ HODNOTY

## DEKLARATÍVNY PRÍSTUP RIEŠENIA PROBLÉMOV

VYTVORÍME MODEL → POUŽIJEME RIEŠIČ

OPTIMALIZAČNÉ PROBLÉMY (BRANCH & BOUND)

MÄKKÉ PODMIENKY (PREFERENCIE)

ĎALŠIE:

LOKÁLNE PREHĽ.

CELOČÍSELNÉ PROGR.

# KOMPETITÍVNE MULTI-AGENTNÉ PROSTREDIE

(1) HRY DVOCH HRÁČOV S NULOVÝM SÚČTOM
A ÚPLNOU INFORMÁCIOU

(2) HRY S NÁHODNÝM VSTUPOM

# MINIMAX

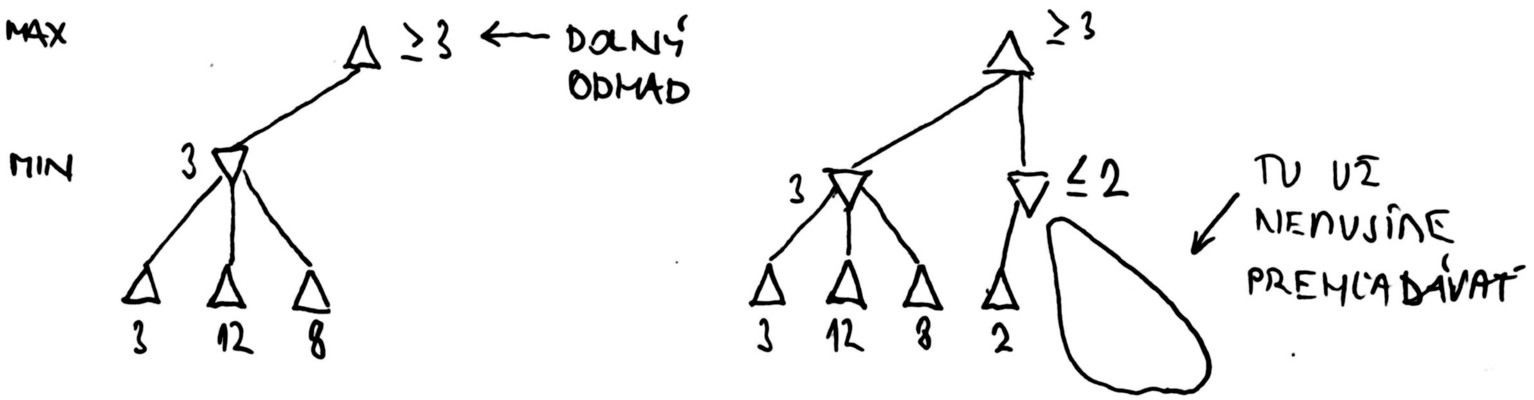
```
function MAX-VALUE (state)
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← -∞
    foreach (a,s) in SUCCESSORS (state) do
        v ← MAX (v, MIN-VALUE(s))
    return v
```

PODOBNE MIN-VALUE ...

ČAS: $O(b^m)$, PAMÄŤ: $O(bm)$

PRE VIAC HRÁČOV: VEKTOR HODNÔT,
KAŽDÝ HRÁČ NA SVOJOM ŤAHU OPTIMALIZUJE SVOJU POLOŽKU

# $\alpha$-$\beta$ ODSEKÁVANIE



28.01.2009 We
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## ALGORITMUS α-β

function MAX-VALUE (state, α, β)

α .. NAJLEPŠIA ALTERNATÍVA PRE MAX POZDĹŽ CESTY
β .. NAJLEPŠIA ALTERNATÍVA PRE MIN POZDĹŽ CESTY

```
    if TERMINAL-TEST (state) then return UTILITY (state)
    v ← -∞
    foreach (a, s) in SUCCESSORS (state) do
        v ← MAX (v, MIN-VALUE (s, α, β))
        if v ≥ β then return v        MIN MÁ LEPŠIU ALTER
        α ← MAX (α, v)
```

PODOBNE MIN-VALUE

PRI PERFEKTNOM USPORIADANÍ ČAS: $O(b^{m/2})$

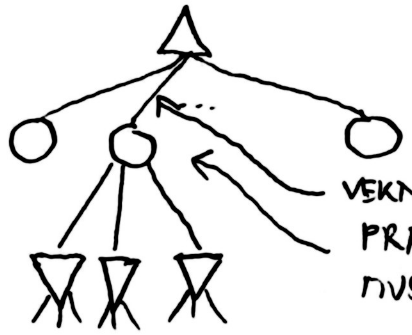
PREHĽADÁVANIE VETIEV MÔŽEME UKONČIŤ SKÔR
→ HEURISTICKÁ EVALUAČNÁ FUNKCIA EVAL
KONCOVÉ UZLY MUSÍ USP. ROVNAKO AKO FCIA. ZISKU
NEKON. UZLY → SILNO ZVIAZANÁ SO SKUTOČNOU ŠANCOU VYHRAŤ

(a) OČAKÁVANÁ HODNOTA
(b) MATERIÁLNA HODNOTA

PROBLÉMY S OREZANÍM:
NESTABILNÝ ODHAD → UKĽUDNENIE (QUIESCENT)
EFEKT HORIZONTU

VYLEPŠENIA:
SINGULÁRNE PREDĹŽENIE
DOPREDNÉ PREREZÁVANIE
TRANSPOZIČNÉ TABUĽKY

# HRY S NÁHODOU



PRIDÁME

UZLY NÁHODY

VEKTOR AKCII:
PRE KAŽDÝ VÝSLEDOK NÁHODY
MUSÍME MAŤ AKCIU

EMINIMAX-VALUE(n) =

| | |
|---|---|
| UTILITY(n) | n JE CIEĽOVÝ |
| $\max_{s \in succ(n)}$ EMINIMAX-VALUE(s) | n HRÁ MAX |
| $\min_{s \in succ(n)}$ EMINIMAX-VALUE(s) | n HRÁ MIN |
| $\sum_{s \in succ(n)} P(s) \cdot$ EMINIMAX-VALUE(s) | n JE UZOL NÁHODY |

POZOR! ÚLOHU ZOHRÁVAJÚ AJ ABSOLÚTNE HODNOTY, NIELEN PORADIE UZLOV

ČAS $O(b^m n^m)$ — $n$ POČET NÁH. VÝSLEDKOV

$\alpha$-$\beta$ AK MÁME OHRANIČ. FCIU. ZISKU

HRY S ČIAST. INFORMÁCIOU (KARTY)

CESTA A .. ZLATO
CESTA B .. ROZCESTIE < DRAHOKAMY / SMRŤ

29.01.2009 Th
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## LOGICKÍ (ZNALOSTNÍ) AGENTI

BÁZA ZNALOSTÍ, ASK, TELL

$\alpha$ JE DÔSLEDOK KB $\Leftrightarrow$ KAŽDÝ MODEL KB JE MODELOM $\alpha$

### POUŽITIE VÝR. LOGIKY

(1) VÝROKOVÉ PREMENNÉ - POPISUJÚ "MAPU SVETA"
(2) VÝROKOVÉ FORMULE - POPISUJÚ
  (a) ZNÁME INFORMÁCIE O SVETE
  (b) OBECNÉ ZNALOSTI O VLASTNOSTIACH SVETA
  (c) ZÍSKANÉ POZOROVANIA
(3) ODVODZOVACIE POSTUPY - ALGORITMUS $\vdash_i$

SYNTAX, SÉMANTIKA (MODEL, $M(\alpha)$, $KB \models \alpha \Leftrightarrow M(KB) \subseteq M(\alpha)$)
  KOREKTNÝ ALG.: $KB \vdash_i \alpha \Rightarrow KB \models \alpha$
  ÚPLNÝ ALG.: $KB \models \alpha \Rightarrow KB \vdash_i \alpha$

### ODVODZOVACIE ALGORITMY

(1) OVEROVANIE MODELOV (MODEL CHECKING)
  (a) ENUMERÁCIA PRAVD. TAB: TT-CHECK-ALL
  (b) DAVIS-PUTNAM-LOGEMANN-LOVELAND (DPLL)
  (c) LOK. PREHL. WALKSAT
(2) ODVODZOVACIE PRAVIDLÁ (INFERENCE RULES)
  (a) HĽADANIE DÔKAZU
  (b) REZOLUČNÁ METÓDA

$KB \models \alpha \Leftrightarrow KB \wedge \neg\alpha$ JE NESPLNITEĽNÁ
$\leadsto$ TESTUJEME SPLNITEĽNOSŤ FORMULÍ V CNF

## DAVIS-PUTNAM-LOGEMANN-LOVELAND DPLL

ÚPLNÝ, KOREKTNÝ, TEST. SPLNIT. CNF (NÁJDE MODEL)

```
function DPLL(clauses, symbols, model)
  if every clause is true in model then return true
  if some clause is false in model then return false
  P, value ← FIND-PURE-SYMBOL (HEURIST. RÝDZEJ PREMENNEJ)
  if P ≠ ∅ then return DPLL(clauses, symbols - P, [P=value|model])
  P, value ← FIND-UNIT-CLAUSE (HEURIST. JEDN. KLAUZULE)
  if P ≠ ∅ then return DPLL(clauses, symbols - P, [P=value|model])
  P ← FIRST(symbols); rest ← REST(symbols)
  return DPLL(clauses, rest, [P=true|model]) or
         DPLL(clauses, rest, [P=false|model]) BACKTRACK.
```

## WALKSAT

HOROLEZ. METÓDA + NÁH. PRECHÁDZKA

```
function WALKSAT(clauses, p, max-flips)
  model ← RANDOM ASSIGNMENT
  for i ← 1 to max-flips do
     if model SATISFIES clauses then return model
     clause ← RANDOM UNSATISF. CLAUSE FROM clauses
     WITH PROB. p FLIP RANDOMLY SEL. SYMBOL IN clause
     else FLIP SYMBOL IN clause THAT MAXIMIZES
        THE NUMBER OF SATISFIED CLAUSES
```

29.01.2009 Th
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## OBECNÁ REZOLÚCIA

DOKAZUJE NESPLNITEĽNOSŤ $KB \wedge \neg\alpha$ V CNF APLIKÁCIOU <u>REZOLUČNÉHO PRAVIDLA</u>

AK $l_1$ A $m_1$ SÚ KOMPLEMENTÁRNE $\Rightarrow$

$$\boxed{\frac{l_1 \vee \ldots \vee l_k \qquad m_1 \vee \ldots \vee m_k}{l_2 \vee \ldots \vee l_k \vee m_2 \vee \ldots \vee m_k}}$$

ALGORITMUS KONČÍ AK:

(a) NEMOŽNO PRIDAŤ ŽIADNU KLAUZULU (POTOM $\neg KB \models \alpha$)
(b) VZNIKLA PRÁZDNA KLAUZULA (POTOM $KB \models \alpha$)

ÚPLNÝ + KOREKTNÝ

### AKO NÁJSŤ MODEL: (b)

POSTUPNE BERIEME PREMENNÉ $P_i$
AK MÁME KLAUZULU S $\neg P_i$ T. Ž. OSTATNÉ LITERÁLY SÚ FALSE PRI OHODN. $P_1, \ldots, P_{i-1}$ $\Rightarrow$ $P_i :=$ false
INAK $P_i :=$ true

```
function PL-RESOLUTION (KB, α) returns true / false
   clauses ← CNF OF KB ∧ ¬α
   new ← ∅
   loop
      foreach Ci, Cj in clauses
         resolvents ← PL-RESOLVE (Ci, Cj)
         if resolvents CONTAINS THE EMPTY CL. → return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false
      clauses ← clauses ∪ new
```

# HORNOVSKÉ KLAUZULE

MAX 1 POZITÍVNY LITERÁL

DOTAZ - ČI JE DANÁ VÝR. PREMENNÁ ODVODITEĽNÁ Z KB

DOPREDNÉ REŤAZENIE / SPÄTNÉ REŤAZENIE
(METÓDA RIADENÁ DATAMI) (METÓDA RIADENÁ DOTAZOM)

DOPR. ŠÍRENIE:

function PL-FC-ENTAILS (KB, q) returns < true / false

```
[ count [clause]      .. INITIALLY COUNT OF PREMISES
  inferred [symbol]   .. INITIALLY FALSE
  agenda              .. INITIALLY LIST OF SYMBOLS KNOWN TO BE TRUE

while agenda ≠ Ø do
    p ← POP (agenda)
    if ¬ inferred [p] then    inferred [p] ← true;
      foreach H-CLAUSE c IN WHOSE PREMISE p APPEARS do
          -- count [c]
          if count [c] = 0 then
              if HEAD [c] = q then return true
              PUSH (HEAD [c], agenda)
```

SPÄTNÉ ŠÍRENIE:

PRE DANÝ DOTAZ HĽADÁME ROZKLAD POMOCOU HORNOVSKEJ KLAUZULE NA PODDOTAZY, KTORÉ RIEŠIME ROVNAKÝM SPÔSOBOM

29.01.2009 Th
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

(ODVODZOVANIE)
FORMÁLNY JAZYK : REPREZENTÁCIA ZNAL. / PRÁCA SO ZNAL.
PROCEDURÁLNY PRÍSTUP / DEKLARATÍVNY PR.

CHCEME DEKL. PRÍSTUP, KOMPOZICIÁLNU SÉMANTIKU (NEZÁLEŽÍ NA KONTEXTE), JEDNOZNAČNOSŤ

VÝR. LOGIKA - FAKTA, KT. PLATIA ALEBO NEPLATIA
PREDIKÁTOVÁ LOGIKA - OBJEKTY A RELÁCIE MEDZI NIMI
TEMPORÁLNA LOGIKA - FAKTA A ČASY, KEDY PLATIA
LOG. VYŠŠÍCH RÁDOV - RELÁCIE MEDZI OBJ. SÚ TAKTIEŽ OBJ.

## PREDIKÁTOVÁ LOGIKA

KONŠTANTY, FCIE., PREDIKÁTY, PREMENNÉ, ... → JAZYK

TELL - PRIDÁVANÉ AXIÓMY (FAKTA AKO ATOMICKÉ TVRDENIA)
DEFINÍCIE (S POUŽITÍM ⇔)
TEORÉMY (ZRÝCHLUJÚ ODVODZOVANIE)
ASK

## REDUKCIA PL NA VL

SKOLEMIZÁCIA !

(GROUNDING)
UZEMNENIE : ZA VŠ. PREMENNÉ DOSADÍME VŠ. MOŽNÉ TERMY
ATOMICKÉ TVRDENIA POTOM ZODPOVEDAJÚ VÝR. PREMENNÝM

KVANTIFIKÁTORY : ∀ → PREMENNÁ NAHRADENÁ TERMOM
∃ → SKOLEMIZÁCIA (PREM. NAHRADENÁ NOVOU KONŠTANTOU)

PROBLÉM : FUNKČNÉ SYMBOLY VEDÚ K ∞ VEĽA TERMOV
HERBRAND : ODVODENIE V PL Z DANEJ KB ∃ ⇔
∃ ODVODENIE W VL V KONEČNEJ PODMN. INŠTANCIOU. KB
KEĎ ALE ODVODENIE ¬∃ → NESKONČÍME ! :(

# ODVODZOVANIE V PL

**LIFTOVANIE** - IBA SUBST., KT. POTREBUJEME PRE ODVODZOVANIE

**LIFTOVANÉ PR. MODUS PONENS** PRE DEFINITNÉ KLAUZULE S 1 POZ. LITERÁLOM

$$\frac{p_1, \ldots, p_n, \quad q_1 \wedge \ldots \wedge q_n \Rightarrow q}{Subst(\theta, q)}$$

KDE $\forall i \; Subst(\theta, p_i) = Subst(\theta, q_i)$

**UNIFIKÁCIA** MGU (MOST GENERAL UNIFIER)

```
function UNIFY(x, y, Θ) returns substitution
   if Θ = failure then return failure
   else if x = y then return Θ
   else if VARIABLE?(x) then return UNIFY-VAR(x, y, Θ)
   else if VARIABLE?(y) then return UNIFY-VAR(y, x, Θ)
   else if COMPOUND?(x) & COMPOUND?(y) then
      return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], Θ))
   else if LIST?(x) & LIST?(y) then
      return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], Θ))
   else return failure

function UNIFY-VAR(var, t, Θ)
   if (var/value) ∈ Θ then return UNIFY(value, t, Θ)
   else if VARIABLE?(t) & (t, value) ∈ Θ then
      return UNIFY(var, value, Θ)
   else if OCCUR-CHECK?(var, t) then return failure
   else return add (var/t) to Θ
```

29.01.2009 Th
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

STANDARDIZÁCIA STRANOU – PRED KAŽDÝM POUŽITÍM TVRDENIA Z BÁZE ZNALOSTÍ V ŇOM PREMENUJEME ∀ PREM. NA ÚPLNE NOVÉ PREM.

ODVODZOVACIE TECHNIKY (IBA PRE DEF. KLAUZULE)

(1) DOPREDNÉ REŤAZENIE
POMOCOU MODUS PONENS ⇝ VŠETKY MOŽNÉ TVRDENIA
DEDUKTÍVNE DATABÁZY (DATALOG), PRODUKČNÉ SYSTÉMY

(2) SPÄTNÉ REŤAZENIE
TYPICKY V LOG. PROGRAMOVANÍ (PROLOG)

(1) function FOL-FC-ASK (KB, $\alpha$) returns substitution / failure

```
loop
  new ← ∅
  foreach sentence r in KB do
    (p1 ∧ ... ∧ pn ⇒ q) ← STANDARDIZE-APART (r)
*   foreach (p1 ∧ ... ∧ pn)θ = (r1 ∧ ... ∧ rn)θ
      r1, ..., rn ∈ KB (STANDARDIZED)
      q' ← SUBST(θ, q)
      if q' is not renaming of sent. in KB ∪ new then
          add q' to new
          if φ ← UNIFY(q', α) ≠ fail then return φ
  if new ≠ ∅ then add new to KB
  else return failure
```

KOREKTNÝ A ÚPLNÝ ALGORITMUS
AK ODVODENIE NEEX., NEMUSÍ SKONČIŤ (AK JAZYK OBSAHUJE FCIE.)

* HĽADANIE VZOROV (PATTERN MATCHING)

MÔŽEME INDEXOVAŤ MNOŽINU FAKTOV PODĽA MENA PREDIK.

P1(...) ∧ P2(...).. ALEBO P2(...) ∧ P1(...)..

ZAČNI TAM, KDE JE MENEJ MOŽNOSTÍ
(PRAVIDLO PRVÉHO NEÚSPECHU)

INKREMENTÁLNE DOPREDNÉ REŤAZENIE

PRAVIDLO (SENTENCE) SA POUŽIJE V ITERÁCII t
AK V ITER. t-1 BOL ODVODENÝ FAKT UNIFIKOVATEĽNÝ
S FAKTOM V TELE PRAVIDLA
(KEĎ DO KB PRIDÁME NOVÝ FAKT ⇝ OZNAČÍME
PRAVIDLÁ, KT. OBSAHUJÚ S NÍM UNIFIK. VÝRAZ)

RETE ALGORITMUS

PREDSPRACUJE PRAVIDLÁ DO PODOBY SIETE ZÁVISLOSTÍ

MAGICKÁ MNOŽINA - RELEVANTNÉ NAVIAZANIE PREM.
MOŽNO ZÍSKAŤ SPÄTNÝM PRIECHODOM CEZ PRAVIDLÁ

29.01.2009Th
PETER CERNO

# NAIL069 UMELÁ INTELIGENCIA I

(2) function FOL-BC-ASK (KB, goals, $\theta$) returns a set of substitutions

```
    if goals = ∅ then return {θ}
    q' ← SUBST(θ, FIRST(goals))
    foreach (p1 ∧ ... ∧ pn ⇒ q) IN KB (STANDARDIZED APART)
        if θ' ← UNIFY(q, q') ≠ failure then
            ans ← ans ∪ FOL-BC-ASK(KB,
                [p1, ..., pn | REST(goals)], COMPOSE(θ, θ'))
    return ans
```

ZODPOVEDÁ LOG. PROGRAMOVANIU

POZOR ! NEGÁCIA = NEÚSPECH DÔKAZU, ODVODENIA

## REZOLÚČNA METÓDA PRE CNF

PRENEXNÍ OP. PREVEDIEME NA TVAR

$(\square_1 x_1) \dots (\square_n x_n)\ O$ ← OTVORENÉ JADRO V CNF

ZAVEDIEME SKOLEMOVSKÉ FCIE. / KONŠTANTY → ODSTR. $\exists$ KV.

A ODSTRÁNIME $\forall$ KVANT.

$$\frac{l_1 \vee \dots \vee l_k \qquad m_1 \vee \dots \vee m_n}{(l_2 \vee \dots \vee l_k \vee m_2 \vee \dots \vee m_n)\,\theta}$$

KDE UNIFY $(l_1, \neg m_1) = \theta$

V KLAUZULÍ PREDPOKLADÁME ŠTANDARDIZÁCIU STRANOU

PRE <u>ÚPLNOSŤ</u>:

(a) ROZŠÍRENIE REZOLÚCIE NA VIAC LITERÁLOV

(b) <u>FACTORING</u> - ODSTRÁNENIE REDUNDANTNÝCH LIT. (T.J. UNIFIKOVATEĽNÝCH)

DOTAZ $\alpha$ NAD KB RIEŠIME APLIKÁCIOU REZOL. KROKOV NA CNF(KB $\wedge \neg\alpha$)
AK ODVODÍME PRÁZDNU FORMULU $\Rightarrow$ KB $\models \alpha$

JE TO <u>ÚPLNÁ</u> A <u>KOREKTNÁ</u> ODV. METÓDA

REZOLÚCNA METÓDA APLIKOVANÁ NA DEFINITNÉ KLAUZULE $\equiv$ SPÄTNÉ REŤAZENIE

LITERÁL PRE ZÍSKANIE ODPOVEDI
$\neg\alpha \vee Answer(w)$

## EFEKTÍVNA REZOLÚCIA

(1) JEDNOTKOVÁ REZOLÚCIA - PREFERUJEME REZ. KROK, KDE JE JEDNA Z KLAUZULÍ JEDNOTKOVÁ

(2) MNOŽINA PODPOR - VYBRANÉ KLAUZULE, Z KT. VŽDY ASPOŇ JEDNA SA ÚČASTNÍ REZOLÚCIE

(3) VSTUPNÁ REZOLÚCIA - KAŽDÉHO REZ. KROKU SA ÚČASTNÍ ASPOŇ JEDNA KLAUZULA ZO VSTUPU (NIE JE ÚPLNÁ)

(4) SUBSUMCIA - ELIMINÁCIA KLAUZULÍ, KT. SÚ ZAHRNUTÉ INÝMI KLAUZULAMI

31.01.2009 So
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## KNOWLEDGE ENGINEERING (ZNALOSTNÉ INŽINIERSTVO)

PROCES, AKO BUDOVAŤ ZNALOSTNÉ BÁZY

(1) POCHOPIŤ PRÍSLUŠNÚ PROBLÉMOVÚ OBLASŤ (DOMÉNU)
(2) URČIŤ AKÉ KONCEPTY SÚ DÔLEŽITÉ
(3) NAVRHNÚŤ FORMÁLNU REPREZENTÁCIU OBJ. A RELÁCIÍ

PODROBNE:

1. IDENTIFIKÁCIA ÚLOHY: AKÉ OTÁZKY KLADIEME KB?
2. KNOWLEDGE ACQUISITION: AKO DANÁ DOMÉNA NAOZAJ FUNGUJE?
3. SLOVNÍK PREDIKÁTOV, FCIÍ, KONŠT.: ⤳ ONTOLÓGIA DOMÉNY
4. ZAKÓDOVANIE OBECNÝCH INFO: AXIÓMY V DOMÉNE
5. ZAKÓDOVANIE ŠPECIFICKEJ PROBLÉMOVEJ INŠTANCIE
6. KLADENIE OTÁZOK INFERENČNÉMU MECHANIZMU: AKO FUNGUJE I.M.
7. LADENIE KB: AKÉ AXIÓMY SME ZABUDLI ATĎ.

## KATEGÓRIA

= MNOŽINA SVOJICH ČLENOV
= KOMPLEXNÝ OBJEKT S RELÁCIAMI MemberOf, SubsetOf

## TAXONÓMIA

KATEGÓRIE ORGANIZUJÚ A ZJEDNODUŠUJÚ BÁZU ZNALOSTÍ PROSTREDNÍCTVOM DEDENIA VLASTNOSTÍ

PODTRIEDY ORGANIZUJÚ KATEGÓRIE DO TAXONÓMIE
SubsetOf (Basketballs, Balls)

# AKCIE A SITUÁCIE

VO VÝROKOVEJ LOGIKE POTREBUJEME KÓPIU KAŽDEJ AKCIE PRE KAŽDÝ ČAS (SITUÁCIU)

$$L^t_{x,y} \wedge FacingRight^t \wedge Forward^t \Rightarrow L^{t+1}_{x+1,y}$$

V PREDIKÁTOVEJ LOGIKE SA KÓPIÁM AXIÓMOV MÔŽEME VYHNÚŤ UNIVERZÁLNOU KVANTIFIKÁCIOU CEZ ČAS (SITUÁCIE)

(1) AKCIE - LOGICKÉ TERMY

GO(x,y), GRAB(G), RELEASE(G)

(2) SITUÁCIE - LOG. TERMY

POČIATOČNÁ SITUÁCIA : $S_0$

SITUÁCIA PO APLIKÁCII a NA s : RESULT(a, s)

(3) FLEXIBILNÉ PREDIKÁTY A FUNKCIE (FLUENTS)

MAJÚ SITUÁCIU V POSL. ARGUMENTE

AT ( object, position, situation )

(4) NEMENNÉ (RIGID, ETERNAL) PR. A FCIE.

ADJACENT (x, y)

PLÁNY : RESULT([], s) = s

RESULT([a|seq], s) = RESULT(seq, RESULT(a, s))

PROJEKČNÁ ÚLOHA : AKÁ JE VÝSL. SITUÁCIA PO APLIKOVANÍ POST. AKCIÍ ?

PLÁNOVACIA ÚLOHA : AKÁ POSTUPNOSŤ AKCIÍ VEDIE K DANEJ SITUÁCII ?

31.01.2009 Sa
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

**AXIÓM POUŽITEĽNOSTI** PRECONDITIONS $\Rightarrow$ POSS (a, s)

$At(Agent, x, s) \land Adjacent(x, y) \Rightarrow Poss(Go(x, y), s)$

**AXIÓM EFEKTU** Poss (a, s) $\Rightarrow$ CHANGES

$Poss(Go(x,y), s) \Rightarrow At(Agent, y, Result(Go(x,y), s))$

POZOR! MUSÍME REPREZENTOVAŤ, ČO SA PO VYKONANÍ AKCIE NEMENÍ!

**AXIÓM RÁMCA** (ČO SA NEMENÍ)

$At(o, x, s) \land o \neq Agent \land \neg Holding(o, s) \Rightarrow At(o, x, Result(Go(y, z), s))$ $\quad O(FA)$

**AXIÓM NASLEDUJÚCEHO STAVU**

$Poss(a, s) \Rightarrow$

$(F_i(\dots, Result(a, s)) \Leftrightarrow$

$(a = A_1 \lor a = A_2 \lor \dots) \lor$ AKCIE, KTORÉ MAJÚ $F_i$ AKO SVOJ EFEKT

$(F_i(\dots, s) \land a \neq B_1 \land a \neq B_2 \land \dots)$ AKCIE, KTORÉ SPÔSOBIA, ŽE $F_i$ NEBUDE PRAVDA

F AXIÓMOV S CELKOVO $O(AE)$ LITERÁLMI

F .. POČET FLEX. PREDIKÁTOV
A .. POČET AKCIÍ
E .. POČET EFEKTOV NA AKCIU

POZOR NA IMPLICITNÉ EFEKTY!

**PROBLÉM DÔSLEDKU** (RAMIFICATION PROBLEM)

AK AGENT NIEČO DRŽÍ A PRESUNIE SA INAM

## KLASICKÝ AXIÓM NASLEDUJÚCEHO STAVU

ČAS RIEŠENIA PROJEKČNEJ ÚLOHY: $O(AEt)$

## UPRAVENÝ AXIÓM NASLEDUJÚCEHO STAVU

$$\begin{aligned}
&Poss(a,s) \Rightarrow (F_i(\ldots, Result(a,s)) \Leftrightarrow \\
&\quad PosEffect(a, F_i) \lor \\
&\quad (F_i(\ldots, s) \land \neg NegEffect(a, F_i)) )
\end{aligned}$$

$PosEffect(A_1, F_i)$
$PosEffect(A_2, F_i)$
...
$NegEffect(B_1, F_i)$
$NegEffect(B_2, F_i)$
...

## SKRYTÉ PREDPOKLADY

PREDPOKLAD UZAVRENÉHO SVETA:
(CWA = CLOSED WORLD ASSUMPTION)

PREDPOKLAD JEDNOZNAČNOSTI MIEN:
(UNA = UNIQUE NAME ASSUMPTION)

01.02.2009 Su
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## PLÁNOVANIE (KLASICKÁ REPREZENTÁCIA)

(NEMÁME FUNKCIE !!!)

L (JAZYK) = KONEČNÁ MNOŽINA PRED. SYMBOLOV A KONŠTANT

ATÓM = PREDIKÁTOVÝ SYMBOL (MÔŽU BYŤ PREMENNÉ)

STAV = MNOŽINA INŠTANCIOVANÝCH ATÓMOV

PREDPOKLAD UZAV. SVETA (UWCA)

- FLEXIBILNÉ ATÓMY (FLUENT)
- NEMENNÉ ATÓMY (RIGID)

OPERÁTOR $o$ = (name($o$), precond($o$), effects($o$))

name($o$) : name($x_1, \dots, x_k$)

precond($o$) : LITERÁLY, KT. MUSIA BYŤ SPLNENÉ

effects($o$): LITERÁLY, KT. SA STANÚ PRAVDIVÉ (IBA FLUENT)

AKCIA = PLNE INŠTANCIOVANÝ OPERÁTOR

AKCIA $a$ JE POUŽITEĽNÁ NA STAV $s$ $\Leftrightarrow$

$\text{precond}^+(a) \subseteq s \quad \wedge \quad \text{precond}^-(a) \cap s = \emptyset$

VÝSLEDOK APLIKÁCIE $s$ NA $a$ JE :

$\gamma(s, a) = s - \text{effects}^-(a) \cup \text{effects}^+(a)$ (IMPLICITNE RIEŠI PROBLÉM RÁMCA)

NECH L JE JAZYK A O MNOŽINA OPERÁTOROV

PLÁNOVACIA DOMÉNA $\Sigma$ NAD L A O JE: $(S, A, \gamma)$

STAVY $S \subseteq \mathcal{P}$({VŠETKY INŠTANCIOVANÉ ATÓMY NAD L})

AKCIE $A$ = {VŠETKY INŠTANCIOVANÉ OPERÁTORY Z O NAD L}

PRECHODOVÁ FUNKCIA $\gamma$ (UZAVRETÁ NA $S$, $A$)

PLÁNOVACÍ PROBLÉM $P$ JE $(\Sigma, s_0, g)$

$\Sigma = (S, A, \delta)$ JE PLÁNOVACIA DOMÉNA

$s_0 \in S$ POČIATOČNÝ STAV

$g$ = MN. INŠTANCIOVANÝCH LITERÁLOV

$s$ SPLŇUJE $g \Leftrightarrow$
$g^+ \subseteq s$, $g^- \cap s = \emptyset$
$S_g = \{ s \mid s \text{ SPLŇUJE } g \}$

PLÁN $\pi = (a_1, \dots, a_k)$ JE POSTUPNOSŤ AKCIÍ.

$\pi$ JE RIEŠENÍM $P \Leftrightarrow \delta(s_0, \pi) \in S_g$

DOPREDNÉ OVERENIE $\Gamma(s) = \{ \delta(s,a) \mid a \text{ JE APL. NA } s \}$

$P$ MÁ RIEŠ. $\Leftrightarrow S_g \cap \Gamma_\infty(s_0) \neq \emptyset$

SPÄTNÉ OVERENIE

AKCIA $a$ JE RELEVANTNÁ PRE $g \Leftrightarrow$

(1) $g \cap \text{effects}(a) \neq \emptyset$

(2) $g^- \cap \text{effects}^+(a) = \emptyset \wedge g^+ \cap \text{effects}^-(a) = \emptyset$

REGRESNÁ (SPÄTNÁ) MNOŽINA CIEĽA $g$ PRE RELEVANTNÚ $a$:

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

$\Gamma^{-1}(g) = \{ \delta^{-1}(g,a) \mid a \in A \text{ JE RELEVANTNÁ PRE } g \}$

$P$ MÁ RIEŠ. $\Leftrightarrow s_0$ JE NADMNOŽINOU NEJAKÉHO $X \in \Gamma_\infty^{-1}(g)$

01.02.2009 Su
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## PLÁNOVANIE SO STAVMI

UZLY = STAVY, HRANY = APLIK. AKCIE, CIEĽ = CESTA $s_0 \rightsquigarrow g$

(1) <u>DOPREDNÉ</u> (FORWARD SEARCH, PROGRESSION)

(2) <u>SPÄTNÉ</u> (BACKWARD SEARCH, REGRESION) (LIFTOVANÁ VERZIA)

FORWARD-SEARCH $(O, s_0, g)$

- $s \leftarrow s_0$ ; $\pi \leftarrow \emptyset$
- loop
  - if $s$ SATISFIES $g$ then return $\pi$
  - $E \leftarrow \{a \mid a$ IS USABLE IN $s\}$ if $E = \emptyset$ then failure
  - NONDETERMINISTICALLY CHOOSE $a \in E$
  - $s \leftarrow \gamma(s, a)$
  - $\pi \leftarrow \pi, a$

BACKWARD-SEARCH $(O, s_0, g)$

- $\pi \leftarrow \emptyset$
- loop
  - if $s_0$ SATISFIES $g$ then return $\pi$
  - $A = \{a \mid a$ IS RELEVANT FOR $g\}$ if $A = \emptyset$ then failure
  - NONDETERMINISTICALLY CHOOSE $a \in A$
  - $\pi \leftarrow a, \pi$
  - $g \leftarrow \gamma^{-1}(g, a)$

LIFTED-BACKWARD-SEARCH $(O, s_0, g)$

$\pi \leftarrow \emptyset$

loop

if $s_0$ SATISFIES $g$ then return $\pi$

$A \leftarrow \{ (o, \theta) \mid$ $o \in O$ STANDARDIZED APART

$\theta$ IS AN mgu FOR AN ATOM OF $g$ AND

AN ATOM OF effects $(o)$,

$\gamma^{-1}(\theta(g), \theta(o))$ IS DEFINED $\}$ if $A = \emptyset$ then failure

$\pi \leftarrow \theta(o).\ \theta(\pi)$

$g \leftarrow \gamma^{-1}(\theta(g), \theta(o))$

POZOR! KOMPLIKÁCIA DETEKCIE CYKLOV!

## PLÁNOVANIE V PRIESTORE PLÁNOV

ZAČÍNAME Z PRÁZDNEHO PLÁNU, PRIDÁVAME ĎALŠIE AKCIE A VÄZBY MEDZI NIMI

(= OPRAVOVANIE KAZOV V ČIASTOČNOM PLÁNE)

MOŽNÉ ÚPRAVY PLÁNU:

(1) PRIDANIE AKCIE

(2) ZVIAZANIE PREMENNÝCH

(3) PRIDANIE PODMIENKY USPORIADANIA

(4) PRIDANIE KAUZÁLNEJ (PRÍČINNEJ VÄZBY)

POČIATOČNÝ STAV A CIEĽ ZAKÓDUJEME AKO ŠPEC. AKCIE:

$a_0$ (POČ. STAV) – ŽIADNE precond, effects $= s_0$

$a_\infty$ (KON. STAV) – precond $= g$, effects $= \emptyset$

PLÁNOVANIE ZALOŽENÉ NA ODSTRAŇOVANÍ KAZOV (FLAWS)

01. 02. 2009 $S_4$
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

ČIASTOČNÝ PLÁN $(A, <, B, L)$

$A$ = MNOŽINA ČIASTOČNE INSTANCIOVANÝCH PLÁN. OPERÁTOROV $\{a_1, \dots, a_k\}$

$<$ = ČIASTOČNÉ USP. NA $A$

$B$ = MNOŽINA VÄZIEB $x = y$, $x \neq y$, $x \in D_i$

$L$ = MNOŽINA KAUZÁLNYCH VÄZIEB $a_i \xrightarrow{p} a_j$

KDE $a_i, a_j \in A$, $a_i < a_j$

$p$ JE VÝRAZ, KTORÝ JE EFEKTOM $a_i$ A PREDP. $a_j$

V $B$ SÚ VÄZBY ZVÄZUJÚCE PRÍSLUŠNÉ PREMENNÉ V $p$

(a) <u>OTVORENÝ CIEĽ</u> (OPEN GOAL) JE <u>KAZOM PLÁNU</u>

JEDNÁ SA O PREDPOKLAD $p$ NEJAKÉHO OPERÁTORU $b$, O KT. ZATIAĽ NEBOLO ROZHODNUTÉ AKO HO SPLNIŤ (NEEXISTUJE KAUZÁLNA VÄZBA $a \xrightarrow{p} b$)

<u>ODSTRÁNENIE OTVORENÉHO CIEĽA</u>

(1) NÁJDI OPERÁTOR $a$ (BUĎ V PLÁNE ALEBO NOVÝ) KTORÝ MOŽNO POUŽIŤ NA SPLNENIE $p$ OPERÁTORU $b$ ($p$ SA VYSKYTUJE MEDZI EF. $a$ A $a$ MÔŽE BYŤ PRED $b$)

(2) ZVIAŽ PREMENNÉ

(3) VYTVOR KAUZÁLNU VÄZBU

(b) <u>HROZBA</u> (THREAT) ĎALŠÍ KAZ, AKCIA, KT. MÔŽE POKAZIŤ KAUZ.V. $a_i \xrightarrow{p} a_j$, $b$ MÁ EFEKT UNIFIKOV. S NEGÁCIOU $p$

$\Rightarrow$ <u>ODSTRÁNENIE</u>: $b < a_i$, $b > a_j$, ŠPEC. NAVIAZAŤ PREMENNÉ

ČIASTOČNÝ PLÁN $\Pi = (A, <, B, L)$ JE RIEŠIACIM PLÁNOM PRE $P = (\Sigma, s_0, g)$ $\Leftrightarrow$

(1) $<$ A VÄZBY V B SÚ GLOBÁLNE KONZISTENTNÉ

(2) ĽUBOVOĽNÁ LINEÁRNE USPORIADANÁ POSTUPNOSŤ PLNE INSTANCIOVANÝCH AKCIÍ A SPĹŇAJÚCICH $<$ A VÄZBY Z B VEDIE Z $s_0$ DO STAVU $\in S_g$.

VETA: $\Pi$ JE RIEŠIACI $\Leftrightarrow$ NEMÁ KAZY (OTV. CIELE, ANI HROZBY) & $<$ A B SÚ GLOBÁLNE KONZISTENTNÉ

## PSP (PLAN-SPACE PLANNING)

PSP ($\pi$)
- flaws $\leftarrow$ OPENGOALS($\pi$) $\cup$ THREATS($\pi$)
- if flaws $= \emptyset$ then return $\pi$
- SELECT ANY $\phi \in$ flaws (DETERMINISTICKÁ VOĽBA)
- resolvers $\leftarrow$ RESOLVE($\phi$, $\pi$)
- if resolvers $= \emptyset$ then return failure
- NONDETERMINISTICALLY CHOOSE $\rho \in$ resolvers
- $\pi' \leftarrow$ REFINE($\rho$, $\pi$)
- return PSP($\pi'$)

OTVORENÉ CIELE - AGENDA O PREDPOKLADOCH AKCIÍ

HROZBY : PRESKÚMANIE $\forall$ TROJÍC AKCIÍ $O(n^3)$

POUŽÍVAJÚ SA IBA KONZISTENTNÉ ZJEDNENIA PLÁNU

(a) DETEKCIA CYKLOV / TRAN. UZÁVER

(b) KONZISTENCIA VZŤAHOV V B (AK JE $\neg$, NP-ÚPLNÝ P.)

01.02.2009 Su
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## ZHRNUTIE ALGORITMOV

| | |
|---|---|
| TREE-SEARCH | PREHĽADÁVANIE STAVOVÉHO PRIESTORU \| NAJSKÔR INICIALIZUJE SEARCH TREE A POTOM V CYKLE EXPANDUJE leaf node NA ZÁKL. strategy \| OKRAJ (FRINGE), VÝKON, b, d, m |
| BFS BREADTH-FIRST SEARCH | UZOL V NAJMENŠEJ HĹBKE, ÚPLNOSŤ, OPTIMALITA IBA AK CENA JE NEKLESAJÚCA FCIA. HĹBKY, $O(b^{d+1})$ / $O(b^{d+1})$ |
| UNIFORM-COST SEARCH | UZOL S NAJMENŠOU HĹBKOU, ÚPLNOSŤ A OPTIMALITA IBA AK $\ldots \geq \varepsilon > 0$ $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$ |
| DFS DEPTH-FIRST SEARCH | NAJHLBŠÍ UZOL, NIE JE ÚPLNÝ ANI OPTIMÁLNY $O(b^m)$ / $O(bm)$ BACKTRACKING $\rightsquigarrow O(m)$ |
| DLS DEPTH-LIMITED SEARCH | ÚPLNOSŤ AK $\ell \geq d$, OPTIMALITA NIE $O(b^\ell)$ / $O(b\ell)$ |
| IDS ITERATIVE DEEPENING DFS | ÚPLNOSŤ, OPTIMALITA AK CENA JE NEKLES. FCIA. HĹBKY $O(b^d)$ / $O(bd)$ |
| BS BIDIRECTIONAL SEARCH | AK BFS NA OBOCH STRANÁCH $\Rightarrow$ ÚPLNOSŤ AJ OPTIMALITA (CENA JE NEKL. FC. HĹBKY), $b^{d/2} + b^{d/2} \ll b^d$ |
| GRAPH-SEARCH | UDRŽUJEME MNOŽINU UZAVRETÝCH STAVOV (closed) |
| (G)BFS GREEDY BEST-FIRST SEARCH | EVALVAČNÁ FCIA. $f(n) := $ HEUR. FCIA. $h(n)$, NIE JE ÚPLNÝ ANI OPTIMÁLNY $O(b^m)$ / $O(b^m)$ |
| A* | $f(n) := g(n) + h(n)$ \| $h(n)$ PRÍPUSTNÁ H. $\Rightarrow$ TREE-SEARCH OPT. \| $h(n)$ KONZIST. HEUR. $\Rightarrow$ GRAPH-SEARCH OPT. EXPANDUJE $\forall$ UZLY $n$: $f(n) < C^*$ A NIEKTORÉ $f(n) = C^*$ MÁ PROBLÉM S PAMÄŤOU |
| IDA* ITERATIVE-DEEPENING A* | ZMENŠENIE PAM. NÁROKOV, ČASTO POUŽÍVANÝ |
| RBFS RECURSIVE BEST-FIRST SEARCH | |
| SMA* SIMPLIFIED MEMORY-BOUNDED A* | AK JE PAMÄŤ DOSTATOČNÁ PRE OPT. CESTU $\Rightarrow$ SMA* NÁJDE OPT. RIEŠ. INAK NÁJDE NAJLEPŠIE MOŽNÉ RIEŠ. S DOSTUP. PAM. |
| HC HILL-CLIMBING | STOCHASTICKÝ HC (RULETA MEDZI ZLEPŠ.) \| HC PRVEJ VOĽBY (1. NÁH. ZLEPŠUJÚCI) HC S NÁH. RESTARTMI (ÚNIK Z LOK. OPTIM) |
| SA SIMULATED ANNEALING | KOMBINUJE METÓDY HC A RANDOM WALK $\Delta E = F[next] - F[current]$, $prob = e^{\Delta E/T}$ |
| LBS LOCAL BEAM-SEARCH | $k$ NÁHODNÝCH STAVOV, ZO $\forall$ SUSEDOV VYBERIE $k$ NAJLEPŠÍCH |
| GA GENETIC ALGORITHMS | FITNESS / SELECTION / CROSS-OVER / MUTATION |
| ONLINE DFS | NEMOŽNO PRESKOČIŤ DO ÚPLNE NOVÉHO STAVU BEZ ABSOLVOVANIA CELEJ CESTY ZAZNAMENÁVAME SI: result[a,s], unbacktracked[s], unexplored[s] |
| LRTA* LEARNING REAL-TIME A* | UČÍ SA result[a,s] A VZDIALENOSŤ DO CIEĽA H[s]: LRTA*-COST(s,a,s',H) |
| BACKTRACKING | VARIABLE ORDERING (DOM / DEG HEURISTIKA) VALUE ORDERING (SUCCEED-FIRST) VYUŽITIE ŠTRUKTÚRY PROBLÉMU, FORWARD-CHECKING, ARC-CONSISTENCY |
| AC-3 ARC CONSISTEN. 1 | SMEROVÉ ODSTRÁNENIE $O(ed^3)$ $e$ = POČET PODMIENOK $d$ = VEĽKOSŤ DOMÉNY |

| | |
|---|---|
| MINMAX | HRA DVOCH HRÁČOV S NULOVÝM SÚČTOM<br>+ ... S NÁHODOU $O(b^m n^m)$, α-β AK M. E OHRANIČ. FCIU. ZISKU |
| α-β PRUNNING | α - NAJLEPŠIA ALTERNATÍVA PRE MAXA<br>β - NAJLEPŠIA ALTERNATÍVA PRE MINA<br>EVALUAČNÁ FCIA. → PROBLÉMY S OREZANÍM<br>→ VYLEPŠENIA |
| TT-CHECK-ALL | ENUMERÁCIA PRAVDIVOSTNEJ TABUĽKY |
| DPLL DAVIS-PUTNAM-LOGEMANN-LOVELAND | TEST SPLNITEĽNOSTI CNF, HEURISTIKA RÝDZEJ PREMENNEJ, JEDNOTKOVEJ KLAUZULE |
| WALKSAT | SPLNITEĽNOSŤ CNF,<br>HOROLEZECKÁ METÓDA + NÁH. PRECHÁDZKA |
| PL-RESOLUTION | POMOCOU REZOLUČNÉHO PRAVIDLA GENERUJEME ∀ MOŽNÉ KLAUZULE, ÚPLNÝ + KOREKTNÝ<br>CNF = KB ∧ ¬α AK VYGENERUJEME PRÁZDNU KLAUZULU → CNF NESPLNITEĽNÁ → KB ⊨ α |
| PL-FC-ENTAILS | HORNOVSKÉ KLAUZULE, GENERUJEME ČO VŠETKO SA DÁ ODVODIŤ (DOPREDNÉ REŤAZENIE)<br>count[clause], inferred[symbol], agenda |
| PL-BC-ENTAILS | HĽADÁME ROZKLAD, HORN. KL. NA PODDOTAZY (SPÄTNÉ REŤAZENIE)<br>PODOBNE AKO PROLOG |
| UNIFY | HĽADÁ MGU DANÝCH TERMOV, UNIFY, UNIFY-VAR, BEZ OCCUR-CHECK BEŽÍ V LIN. ČASE |
| FOL-FC-ASK | DOPREDNÉ REŤAZENIE, KOREKTNÝ + ÚPLNÝ, AK ODVODENIE NEEX., NEMUSÍ SKONČIŤ<br>HĽADANIE VZOROV (PATTERN MATCHING), INKREMENT. DOPR. REŤAZENIE, RETE ALG. |
| FOL-BC-ASK | ZODPOVEDÁ LOG. PROGRAMOVANIU, SPÄTNÉ REŤAZENIE |
| ACTIONS & SITUATIONS | AKCIE, SITUÁCIE, FLUENTY, RIGID PRED., PROJEKČNÁ / PLÁN. ÚLOHA |
| AXIÓMY | AXIÓM POUŽITEĽNOSTI, AXIÓM EFEKTU, AXIÓM RÁMCA<br>KLASICKÝ / UPRAVENÝ AXIÓM NASL. STAVU, CWA, UNA<br>PROBLÉM DÔSLEDKU (RAMIFICATION PROBLEM) |
| FORWARD-SEARCH | POSTUPNE APLIKUJE POUŽITEĽNÉ AKCIE AŽ KÝM STAV NESPĹŇA CIEĽ |
| BACKWARD-SEARCH | POSTUPNE APLIKUJE RELEVANTNÉ AKCIE PRE DANÝ CIEĽ (INVERZNE)<br>AŽ KÝM POČIATOČNÝ STAV NESPĹŇA AKT. CIEĽ |
| LIFTED BACKWARD-SEARCH | NAMIESTO AKCIÍ POUŽÍVAME LIFTOVANÉ OPERÁTORY, MGU<br>KOMPLIKOVANÁ DETEKCIA CYKLOV |
| PSP PLAN-SPACE PLANNING | POSTUPNÉ ODSTRAŇOVANIE KAZÍ: OTVORENÝ CIEĽ / HROZBA |

01.02.2009 S4
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

## VYBRANÉ ALGORITMY

function RECURSIVE-DLS (node, problem, limit)
cutoff-occured? ← false
(1) if GOAL-TEST then return SOLUTION
(2) else if DEPTH[node] = limit then return cutoff
(3) else EXPANDUJEME node A V CYKLE SKÚŠAME ∀ NÁSLED.
if cutoff-occured? then return cutoff
else return failure

NAJSKÔR G-TEST AŽ POTOM DEPTH-LIMIT !

function IDA* (problem)
INICIALIZÁCIA, f-limit ← f-COST(root)
loop
solution, f-limit ← DFS-CONTOUR(root, f-limit)
if solution ≠ ∅ then return solution
if f-limit = +∞ then return failure

VRACIAME SOL. + + NOVÝ LIMIT !

function DFS-CONTOUR (node, f-limit)
(1) if f-COST(node) > f-limit then return null, f-COST(node)
(2) if GOAL-TEST then return SOLUTION, f-limit
next-f ← ∞
(3) EXPANDUJEME node A SKÚŠAME NÁSL., new-f MINIMALIZ.
return null, next-f

NAJSKÔR f-LIMIT AŽ POTOM G-TEST!

VRACIAME DVOJICU
SOL. + NEW f-COST
LEPŠÍ ODHAD

function RBFS (problem, node, f-limit)
  if GOAL-TEST then return SOLUTION
  EXPANDUJEME node → successors
  AK successors = ∅ then return failure, +∞
  UPDATUJEME f[s] ∀s ∈ successors NA max (g(s)+h(s), f[node])
  loop
    VYBERIEME NAJLEPŠIEHO best ∈ successors
      A ALTERNATÍVU alternative
    AK f[best] > f-limit then return failure, f[best]
    result, f[best] ← RBFS (problem, best, min (f-limit, alternative))
    if result ≠ failure then return SOLUTION


h PRÍPUSTNÁ ⇒ OPTIMALITA
ČAS : EXPONENCIÁLNY

→ POZRI SMA* !

function ONLINE-DFS-AGENT (s) returns an action
  if GOAL-TEST then STOP
  if s IS NEW then unexplored[s] ← ACTIONS (s)
  if r ≠ NULL then result[a,r] ← s, ADD r TO unback.[s]
  if unexpl.[s] = NULL then
    if unback.[s] = NULL then STOP
    else ..
  else ..
  r ← s, return a

01. 02. 2009 Su
PETER ČERNO

# NAIL069 UMELÁ INTELIGENCIA I

function LRTA*-AGENT(s) returns an action
  if GOAL-TEST then STOP
  if s is NEW then $H[s] \leftarrow h(s)$
  if $r \neq$ NULL then result$[a, r] \leftarrow s$
    UPDATUJEME $H[r]$ CEZ VŠETKY AKCIE Z $r$
  VYBERIEME AKCIU S MIN. CENOU $\rightsquigarrow$ LRTA*-COST

function AC-3
  while $Q \neq \emptyset$ do
    VYBERIEME $(X_i, X_j)$, ODSTRÁNIME Z $X_i$
    $\rightsquigarrow$ DO $Q$ NAHÁDŽEME $(X_k, X_i)$

AXIÓM POUŽITEĽNOSTI   PRECOND $\Rightarrow$ Poss$(a, s)$

AXIÓM EFEKTU   Poss$(a, s) \Rightarrow$ CHANGES

AXIÓM RÁMCA

AXIÓM NASL. STAVU

$$\text{Poss}(a,s) \Rightarrow \left( F_i(\ldots, \text{RESULT}(a,s)) \Leftrightarrow \underbrace{(a=A_1 \vee a=A_2 \vee \ldots)}_{\text{PosEffect}(a, F_i)} \vee (F_i(\ldots, s) \wedge \underbrace{a \neq B_1 \wedge \ldots}_{\neg \text{NegEffect}(a, F_i)}) \right)$$