Multiagentové systémy

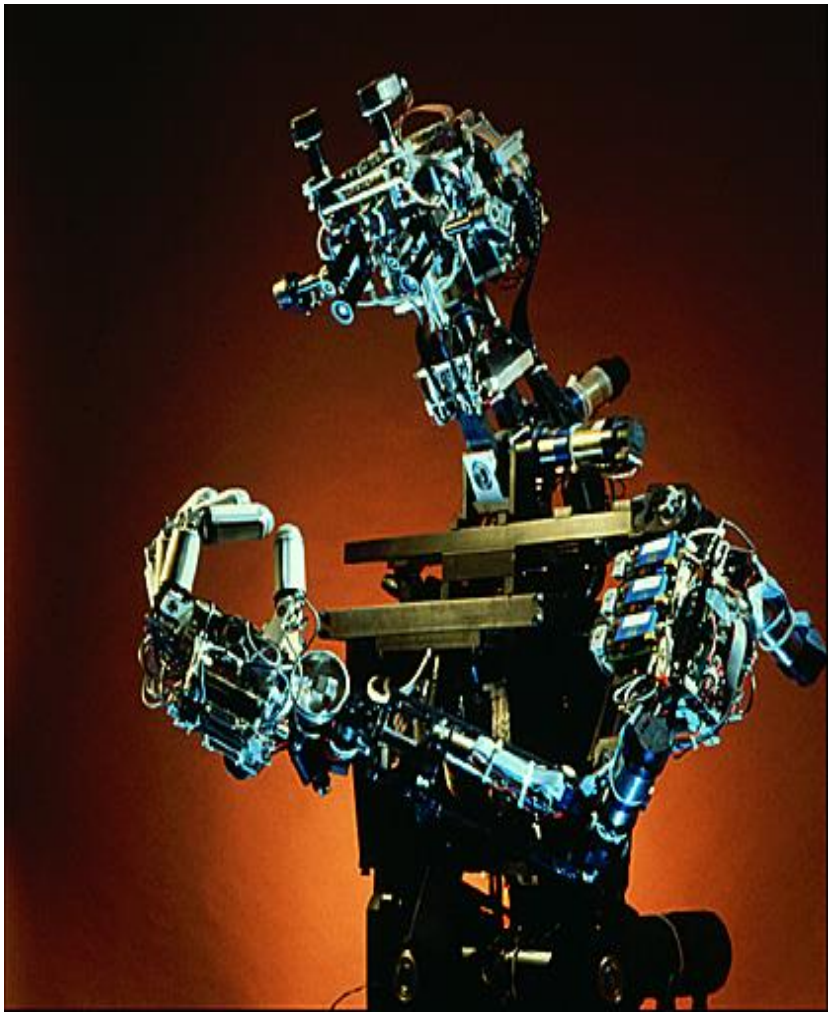
Andrej Lúčny

KAI FMFI UK

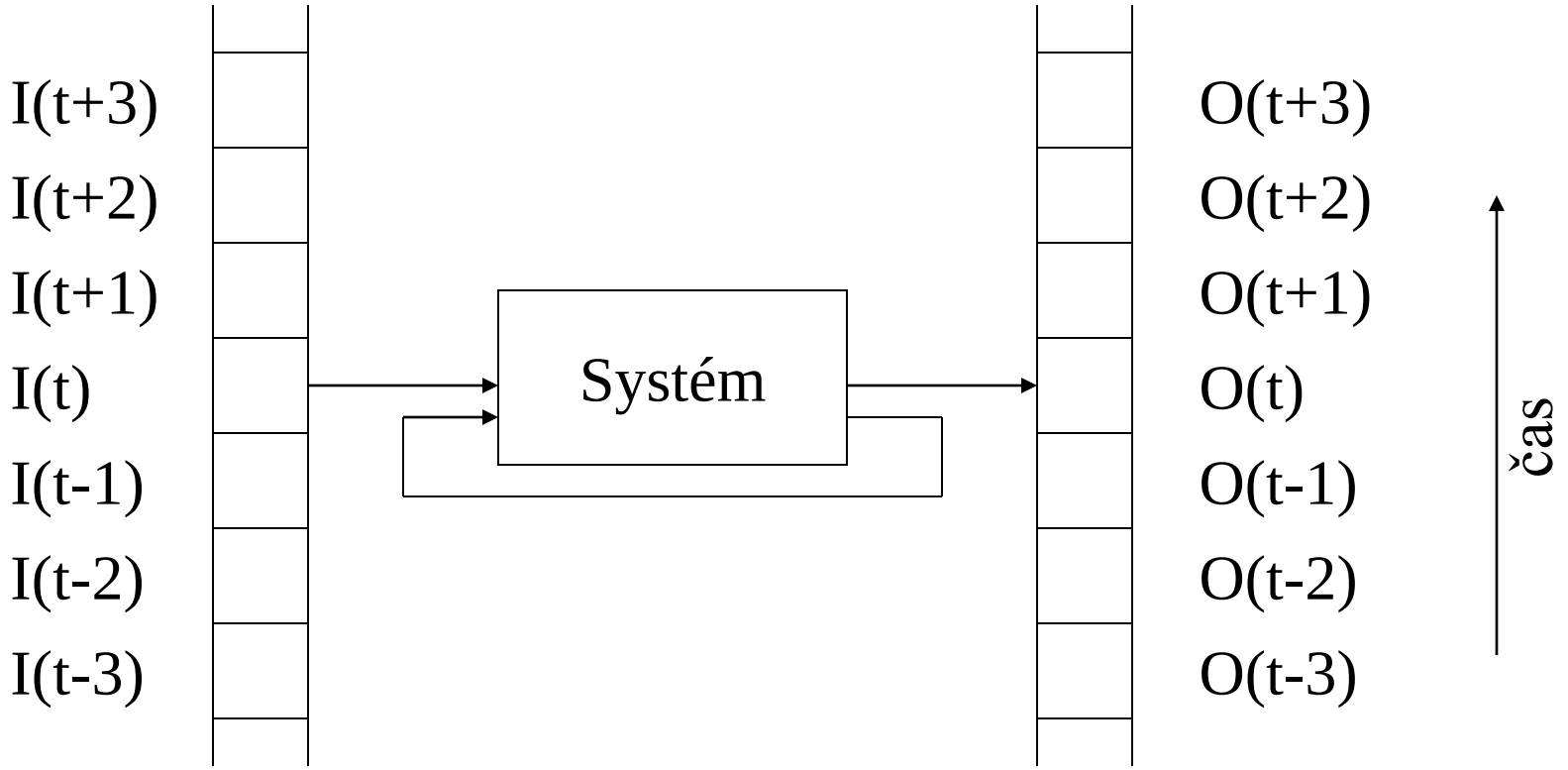
lucny@fmph.uniba.sk

http://www.agentspace.org/mas

Multiagentový prístup k riadeniu



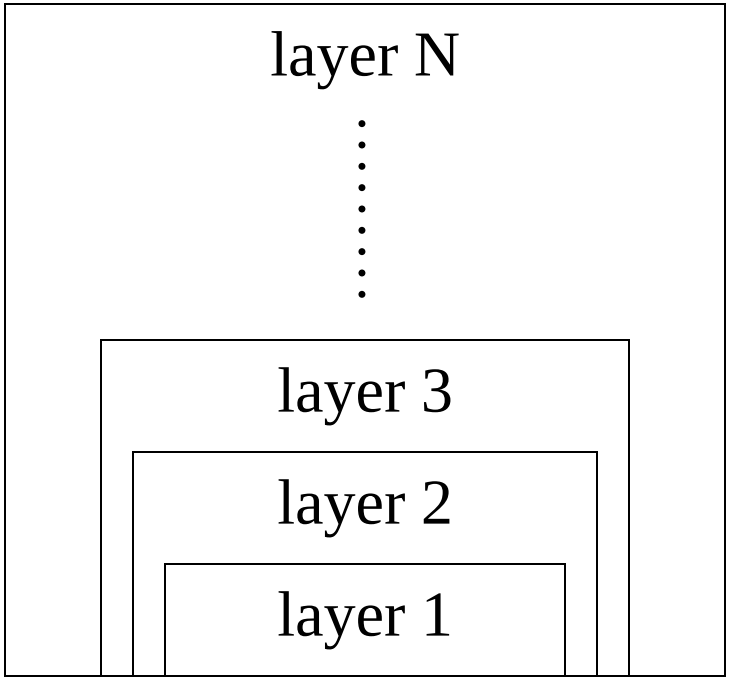
Riadenie



Subsumpčná architektúra

- Je architektúrou autonómneho systému, spravidla mobilného robota
- zjednodušene napodobňuje evolúciu v prírode tým, že nová verzia zahrňuje celú verziu starú (subsume = zahrňovať).
- jednotlivé evolučné kroky však vykonáva vývojár
- Vychádza z princípov tzv. kambrickej alebo „novej“ umelej inteligencie

Biologická motivácia



Kambrická inteligencia ("Embodied intelligence")

Základné pojmy

- Situovanosť
- Stelesnenosť
- Emergencia
- Interakcia
- Hierarchia (Inkrementálnosť)

Situovanosť

- Riadenie navrhujeme pre konkrétny súbor situácii, v ktorých sa systém môže nachádzať.
- Môžeme najprv urobiť verziu, ktorá funguje len pre časť týchto situácii a neskôr ju rozšíriť na ostatné
- Nebudeme situácie ošetrovať jediným modulom, ale špecifické situácie budú ošetrované špecifickými modulmi – „raz zaberie taký trik, inokedy onaký“
- Nesnažíme sa budovať abstraktnú reprezentáciu sveta. „Najlepšou reprezentáciou sveta je svet sám“.

Stelesnenosť

- Riadenie navrhujeme pre konkrétne telo
- Neoddeľujeme návrh od implementácie
- Navrhujeme s prototypom v ruke
- Tým je umožnené po skončení každej fázy vývoja systém testovať a ladiť.
- Pracujeme so skutočnými vstupmi, spoliehame sa na ich reálny charakter.

Emergencia

- Správanie systému je výsledkom interakcie jednotiek z ktorých sa skladá
- Súvislosť medzi správaním jednotiek a systému môže byť na prvý pohľad nejasná
- Pri vhodnej štruktúre systému môžeme explicitnou implementáciou určitých schopností implicitne implementovať schopnosť doteraz neuvažovanú. Vývojár môže byť neschopný tento jav predvídať, hoci jeho dodatočné vysvetlenie by aj nebolo zložité.

Interakcia

- Systém môže inteligenciu čerpať z dynamiky prostredia. Často možno zložitý riadiaci systém nahradiť jednoduchým, ktorý vhodne stavia na dynamike prostredia
- Paradoxne (na rozdiel od systémov ktoré obsahujú kognitívny modul) v statickom prostredí sa takýto systém správa horšie než v dynamickom.
- Inteligencia sa prejavuje reakciami systému na stav jeho prostredia, teda v globálnom správaní, nemožno ju nájsť vnútri systému
- Reaktivita systému zabezpečuje jeho odolnosť voči rušivým zmenám a prechodným stavom v prostredí

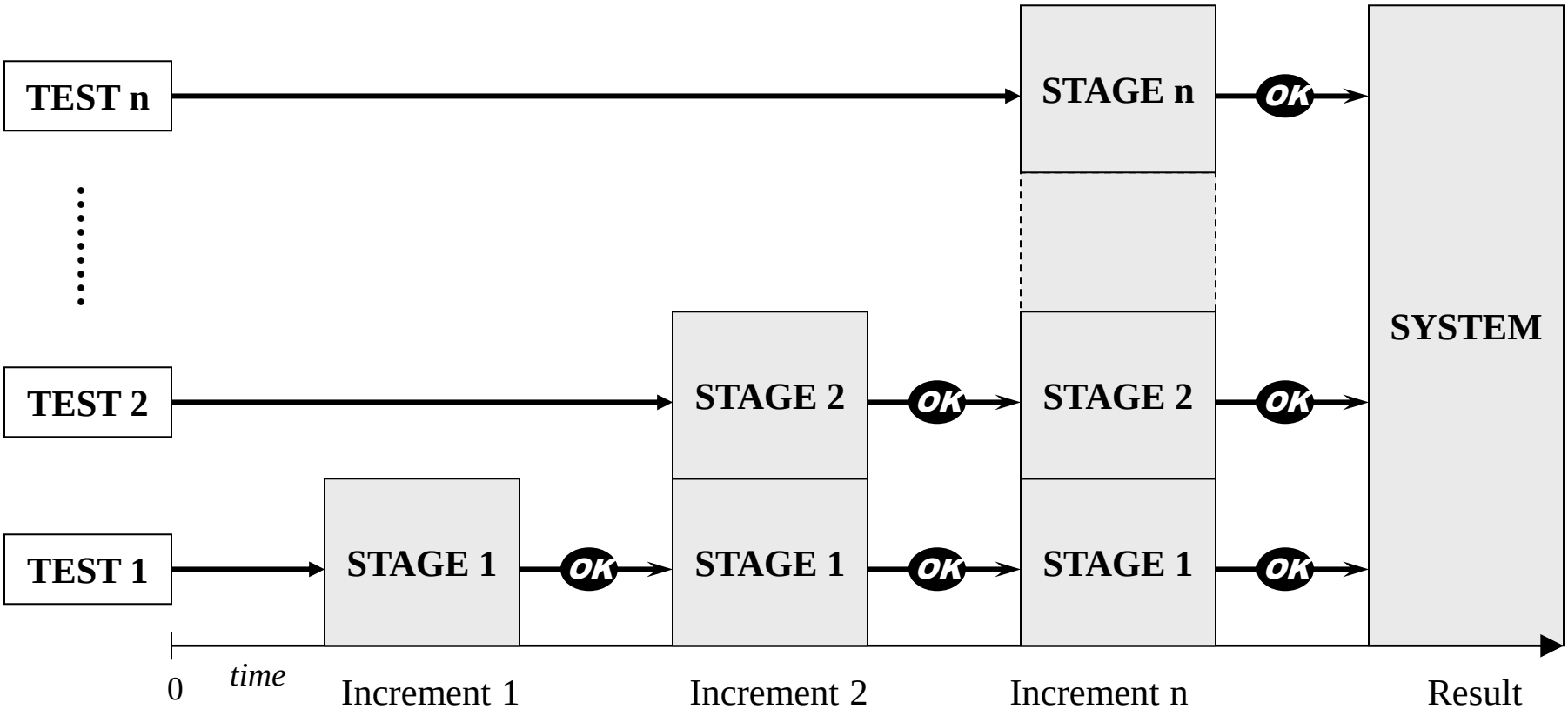
Hierarchia (Inkrementálnosť)

- Systém vyvíjame inkremenálne, po vrstvách, pričom každá implementuje určitú aktivitu
- Postupujeme pritom zdola nahor. Začíname s hierarchicky nižšími vrstvami (biologická metafora: historicky staršími), teda s primitívnejšími aktivitami
- Postupne pridávame nové a nové vrstvy, pričom po každom inkremente, testujeme, ladíme a opravujeme, kým naozaj nezbehnú všetky testy určené pre danú fázu vývoja úspešne

Hierarchia (Inkrementálnosť)

- Vďaka tomu sa nám nemôže stať že implementácia zlyhá na integrácii častí v celok. Hoci na druhej strane nám potenciálne hrozí, že systém dostaneme do stavu, že novú aktivitu už nemožno konzistentne pridať
- Hierarchia systému spočíva v tom, že vyššie vrstvy môžu využívať nižšie. Môžu ich pasívne monitorovať alebo aj aktívne na ne pôsobiť tým, že potláčajú alebo dokonca modifikujú ich činnosť

Inkrementálne, zdola - nahor



Subsumpcia

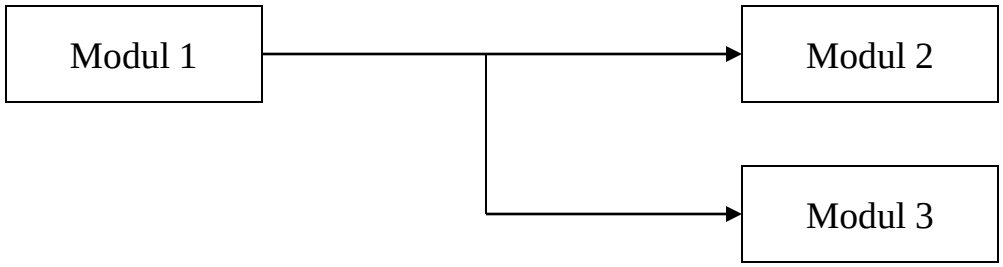
- Ako však môže hierarchicky vyššia – teda neskoršia - vrstva pôsobiť na hierarchicky nižšiu – teda skoršiu ?
- Veď pri implementácii tej nižšej sme takúto možnosť neuvažovali a teda sme nevybudovali žiadne rozhranie, ktorým by mohla vyššia vrstva na nižšiu vplývať

Subsumpcia

- Riešenie: systém musí mať takú modularitu, pri ktorej je toto rozhranie implicitne prítomné

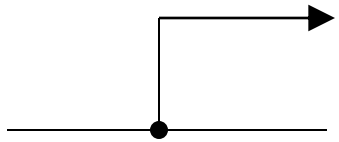
Subsumpčná architektúra

- Systém sa skladá z autonómych modulov (Augmented Finite State Automata)
- Tie sú prepojené komunikačným vedením, po ktorom sú prenášané správy

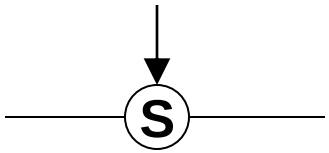


Subsumpčná architektúra

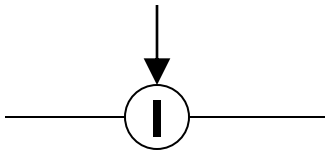
- Vyššia vrstva môže s nižšou manipulovať pomocou:
  - odpočúvania



  - supresie

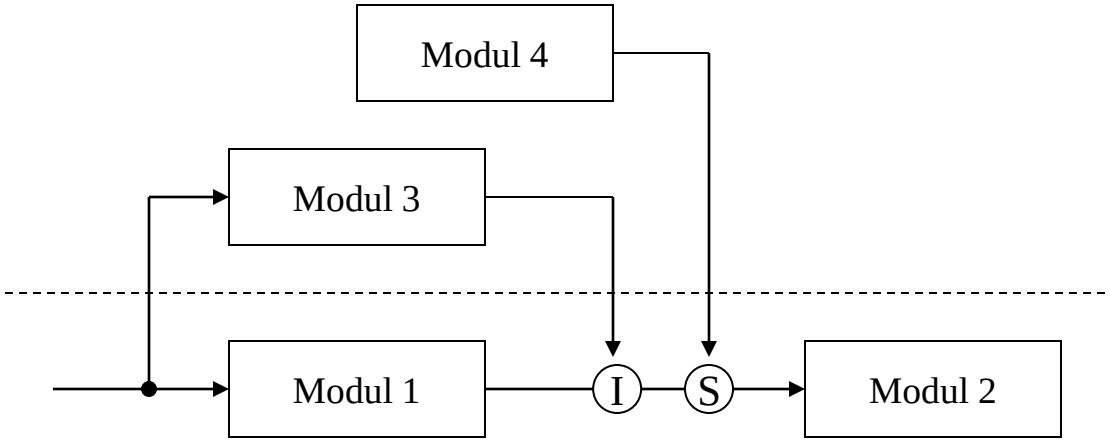


  - inhibície



Subsumpčná architektúra

- Pri pridaní novej vrstvy použijeme tieto mechanizmy na zavedenie kooperácie

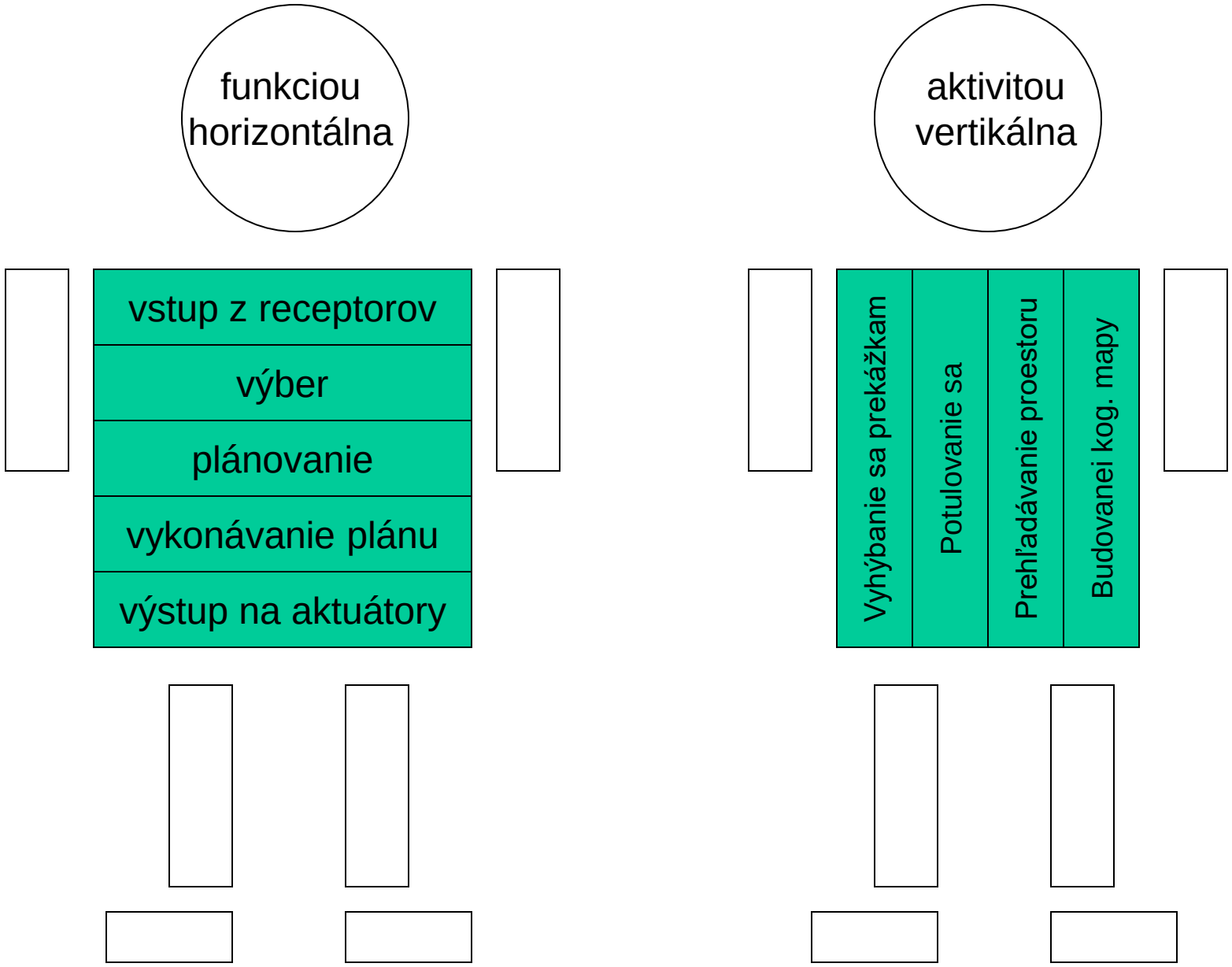


Dekompozícia

- Funkciou = spolu (z hľadiska hierarchických vrstiev) sú kódy realizujúce jednu funkciu, napr. spracovanie obrazu

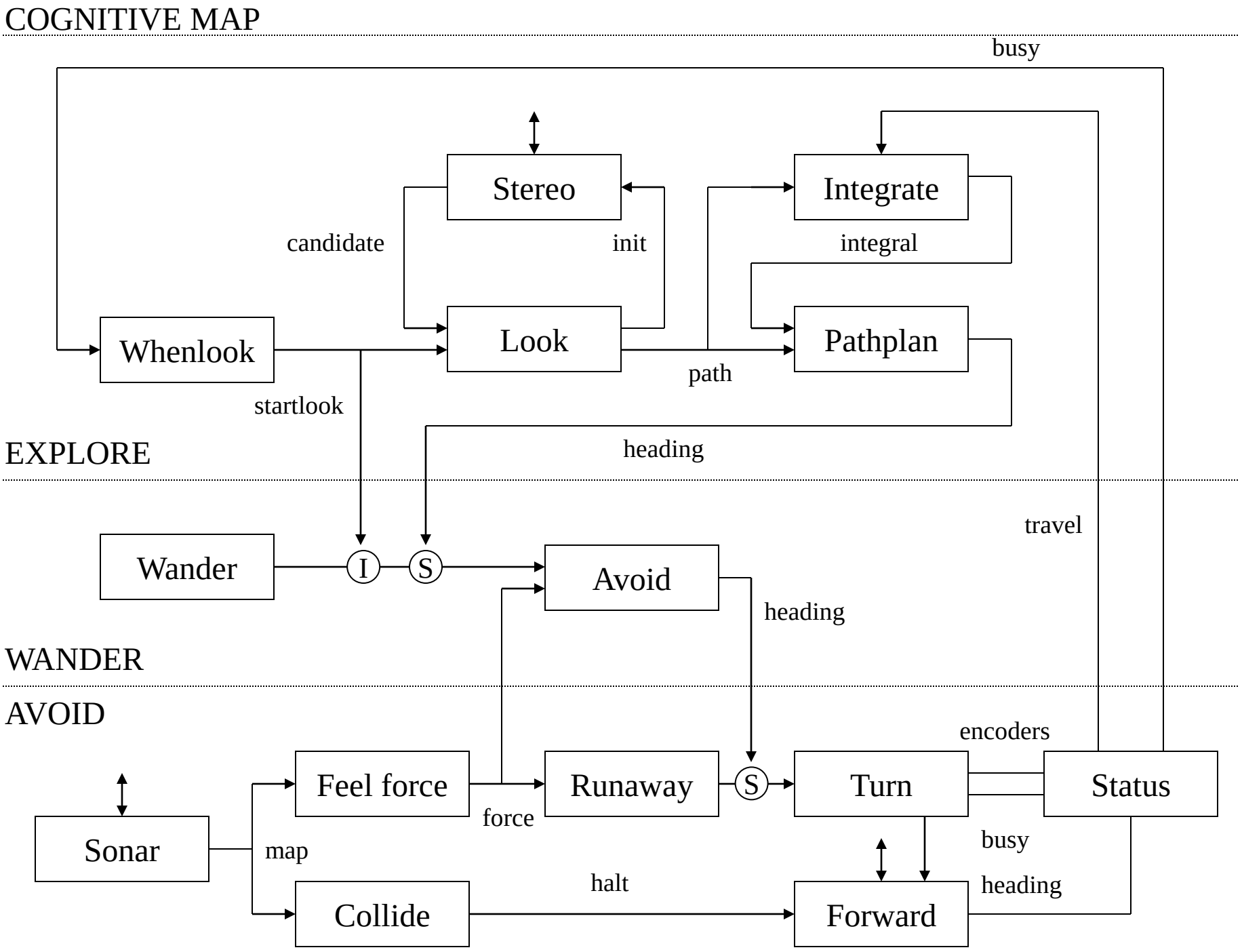
- Aktivitou = spolu sú kódy realizujúce jednu aktivitu, napr. vyhýbanie sa prekážkam

Subsumčná architektúra je založená na dekompozícii aktivitou


funkciou
horizontálna
vstup z receptorov
výber
plánovanie
vykonávanie plánu
výstup na aktuátory
aktivitou
vertikálna
Vyhýbanie sa prekážkam
Potulovanie sa
Prehľadávanie proestoru
Budovanei kog. mapy

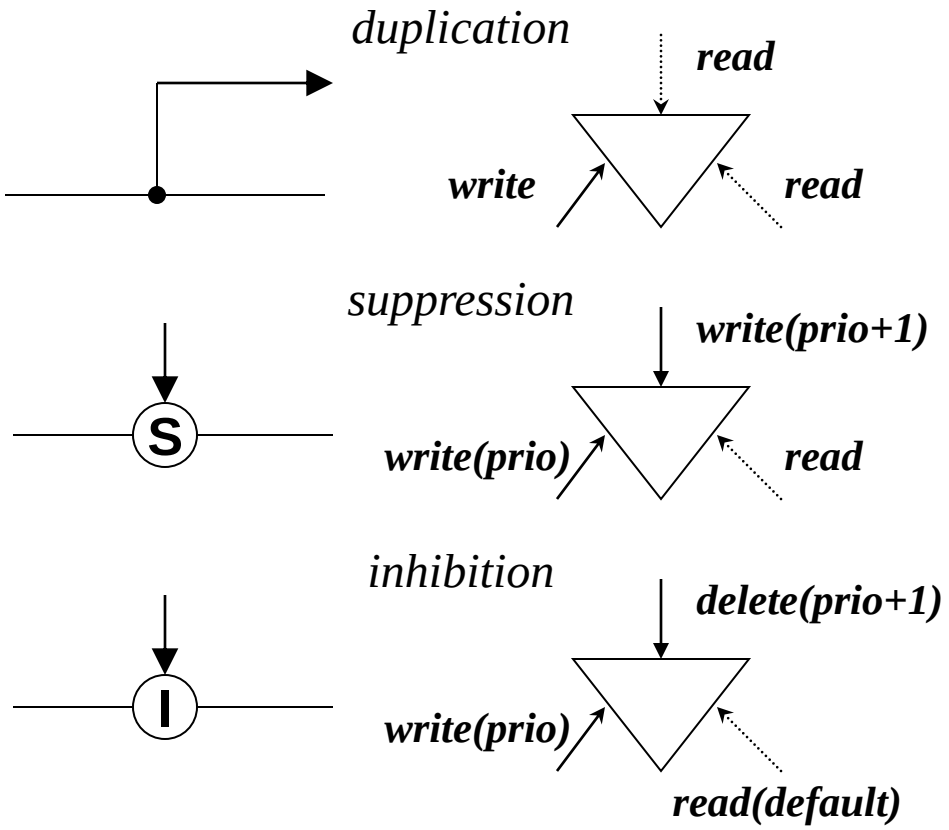
Príklad:

- *Vývoj navigácie robota s dvojkolesovým podvozkom v kancelárskych priestoroch môže byť nasledovný: Začneme s robotom, ktorý ide výlučne vpred. Potom pridáme vrstvu riadenia, ktorá rozpoznáva prekážky a pokiaľ sú detegované, tak nahradí správy pre jedno z kolies pokynom ísť vzad. Následkom toho sa robot vyhýba prekážkami. Pomerne ľahko však zotrvá v určitej cyklickej trajektórii. Preto pridáme vrstvu, ktorá raz za čas spôsobí náhodné pootočenie. Toto točenie robíme len vtedy, keď nehrozí zrážka a realizujeme ho tak, že navodzujeme stav, kedy sa nižšie vrstvy domnievajú, že rozpoznali prekážku. Ďalej pridáme vrstvu, ktorá aktívne vyhľadáva absolútny smer, ktorým sa dá presunúť do inej časti priestoru. Keď je takýto smer nájdený, implementujeme jeho sledovanie tak vyvolávame v nižšej vrstve ako keby náhodné otáčanie, regulujeme ho však tak, aby robot sledoval zvolený smer. Ďalšie vrstvy môžu detegovať význačné miesta priestoru a navigáciu do takého miesta zadaného používateľom realizovanú ako zdanlivý presun na voľné miesto.*


COGNITIVE MAP
busy
Stereo
Integrate
candidate
init
integral
Whenlook
Look
Pathplan
path
startlook
heading
EXPLORE
travel
Wander
I
S
Avoid
heading
WANDER
AVOID
encoders
Feel force
Runaway
S
Turn
Status
Sonar
force
map
busy
halt
heading
Collide
Forward

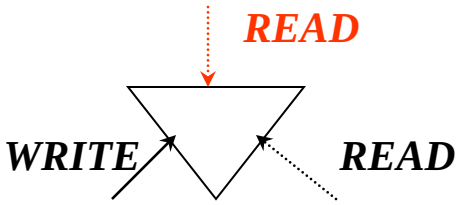
Realizácia v Agent - Space

- Riešenie vyjadrené v subsumpčnej architektúre možno poľahky transformovať do agent-space



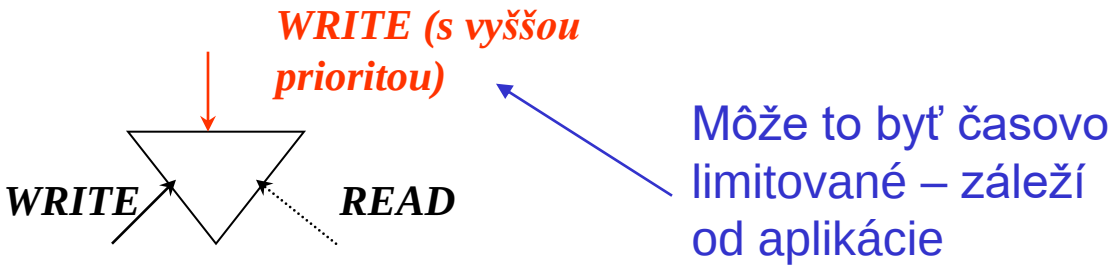
Odpočúvanie

- Vyššia vrstva nedeštruktívne odoberá údaje z nižšej, sleduje čo sa tam deje



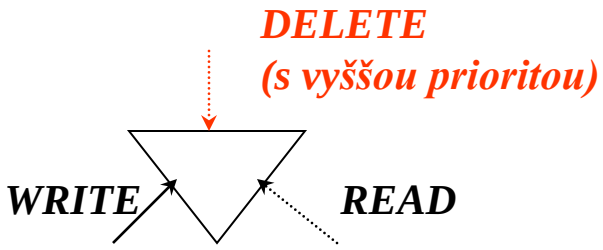
Supresia

- Vyššia vrstva nahradí údaj plynúci v nižšej vlastnou hodnotou



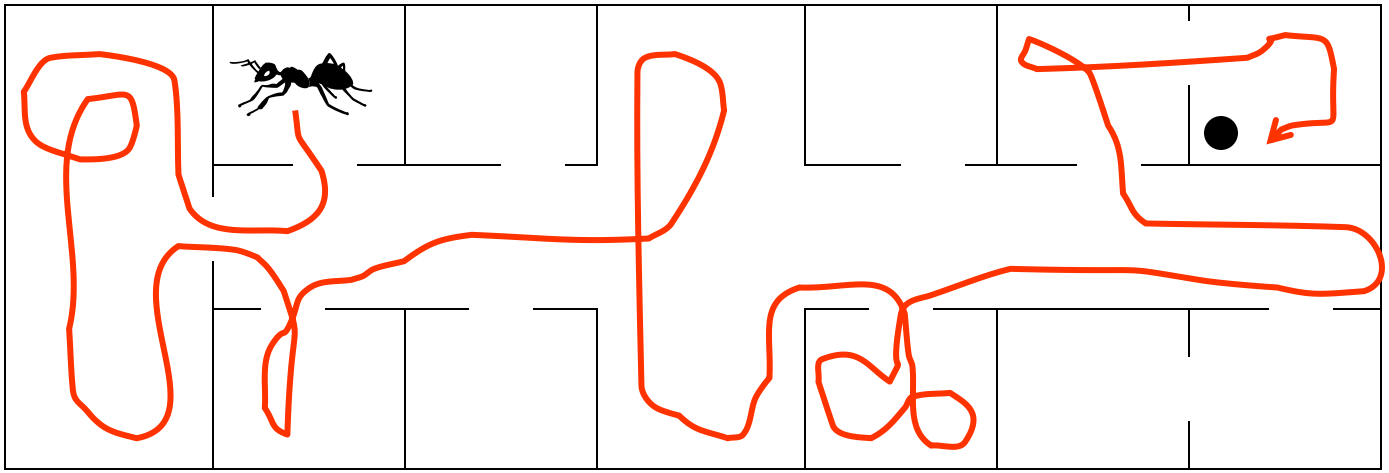
Inhibícia

- Vyššia vrstva zmaže údaj v nižšej vrstve . Zastaví tam dátový tok

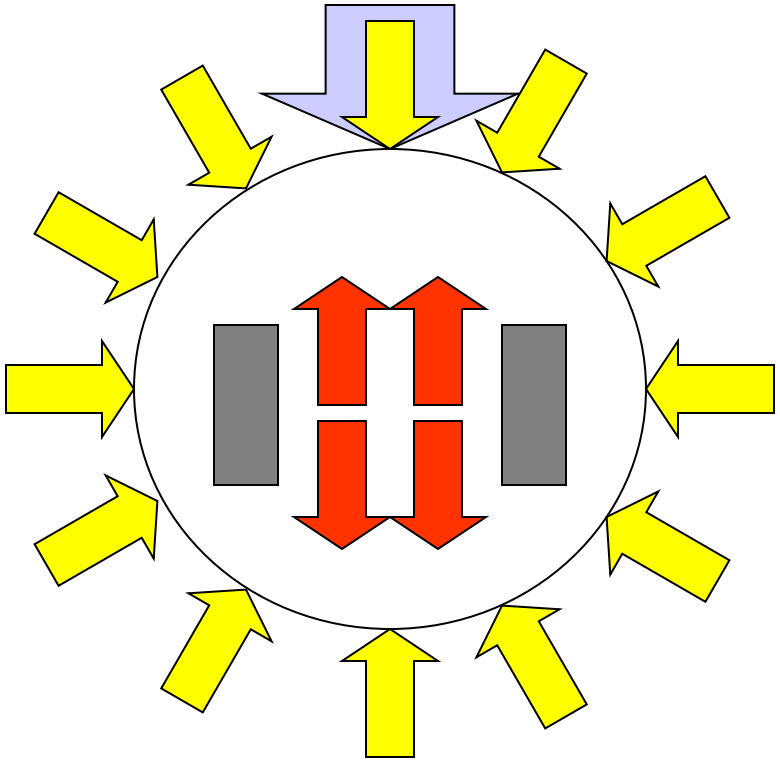


Riadiaci systém robota

Úloha: urobiť robota, ktorý dokáže nájsť na zem umiestnenú kovovú guličku niekde v kancelárskych priestoroch na jednom poschodí



Senzory a aktuátory



Sonary na detekciu najbližej prekážky

Kolesá na pohyb vpred, vzad a otáčanie

Detektor guličky

Návrh systému

Podľa princípu subsumpcie

STOP – Zastavenie pri guličke

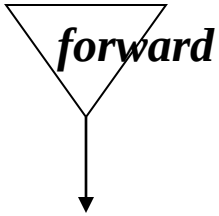
EXPLORE – Prehľadávnie priestoru

WANDER - Potulovanie sa

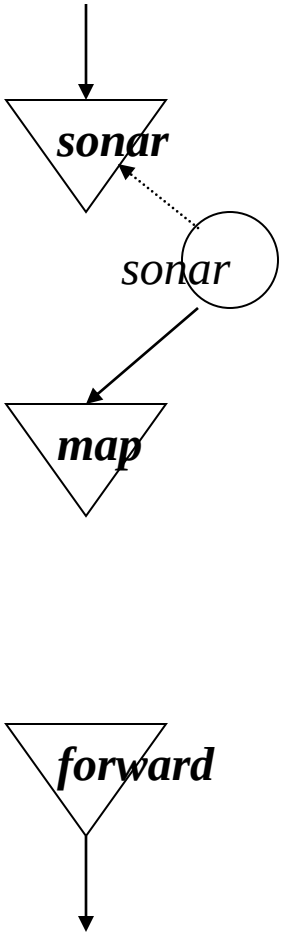
AVOID - Výhýbanie sa prekážkam

AVOID

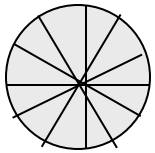
Za normálnych okolností, ideme stále vpred



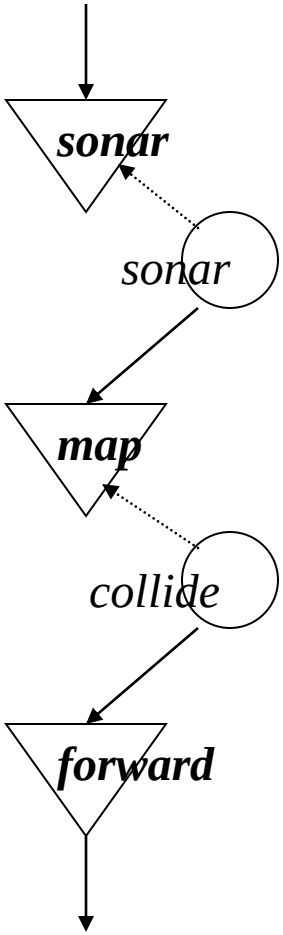
AVOID



Sonar zostaví pole vzdialeností najbližších prekážok v rôznych smeroch



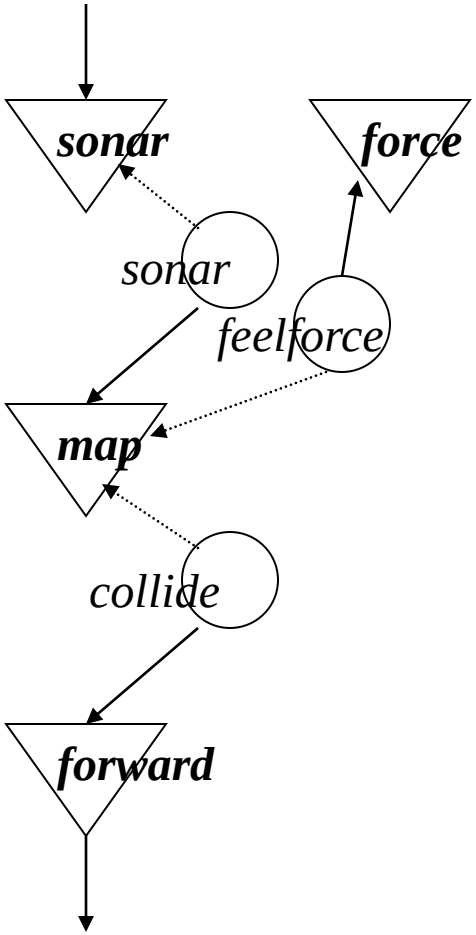
AVOID



Ked hrozí náraz, collide zastaví forward. Zastaví ho na základe údajov z dopredného sonaru

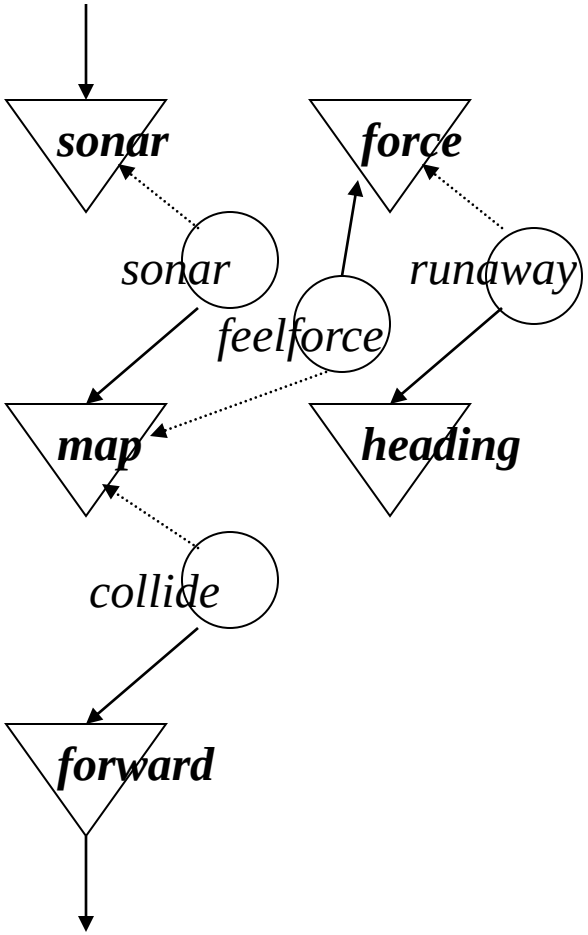
Tým pádom ideme rovno až kým nehrozí náraz, potom zastavíme

AVOID



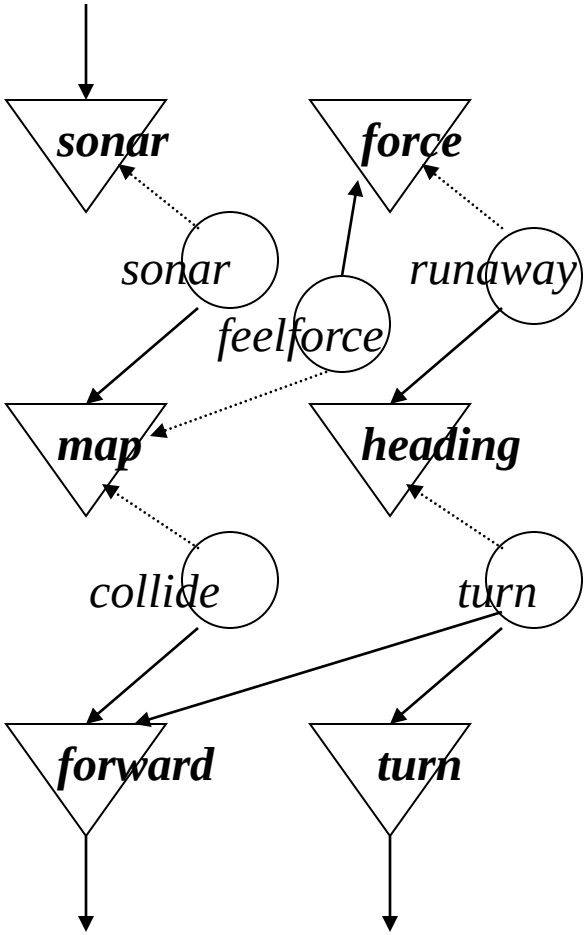
Feelforce vyhodnocuje smer v ktorom najviac hrozí zrážka

AVOID



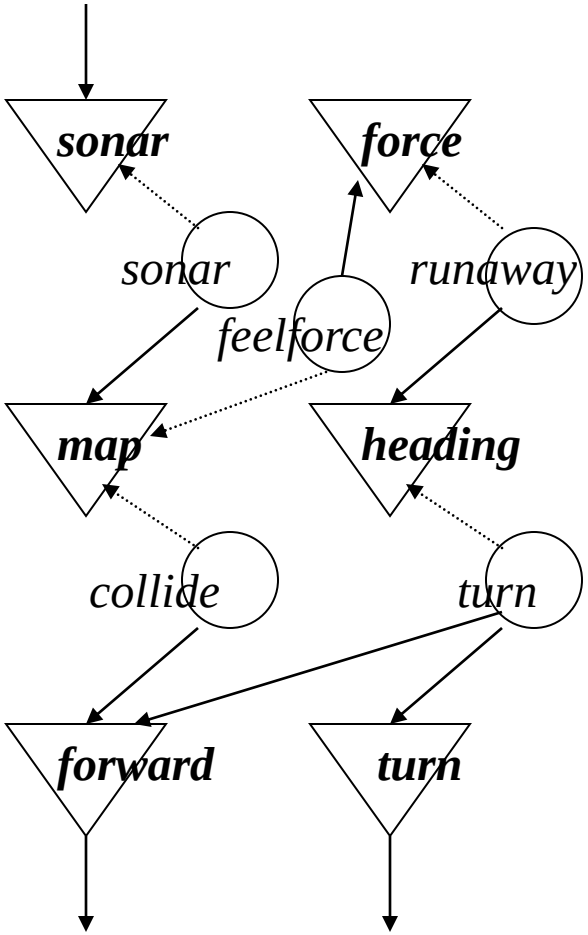
Runaway zavelí uberať sa opačným smerom

AVOID



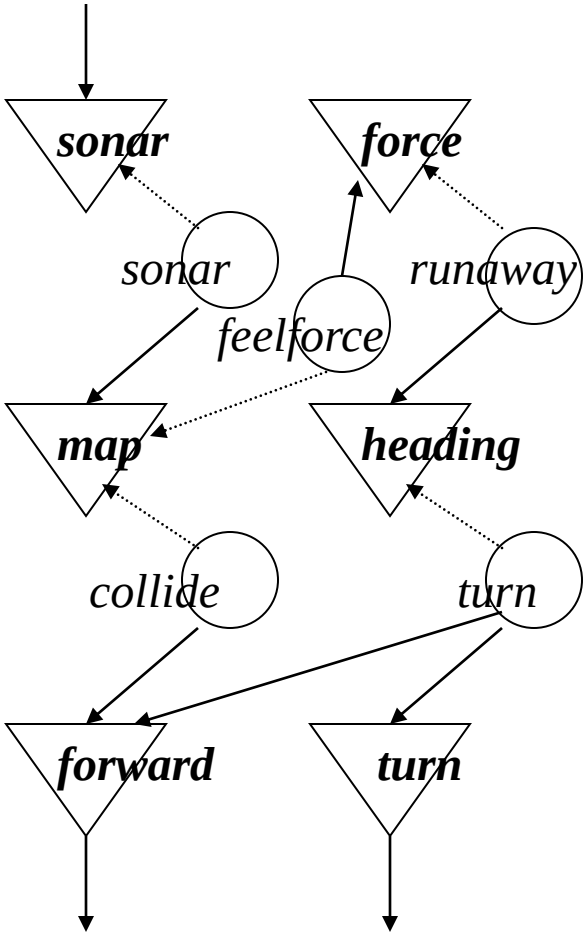
A turn podľa toho smeru riadi otáčanie kolies, prípadne aj cúvanie

AVOID



Uvedomme si, že toto otáčanie má spätne vplyv na obsah map a tým pádom na heading. Čím viac sa robot odkloní od smeru v ktorom hrozí zrážka, tým menší je odklon požadovaný v heading

AVOID

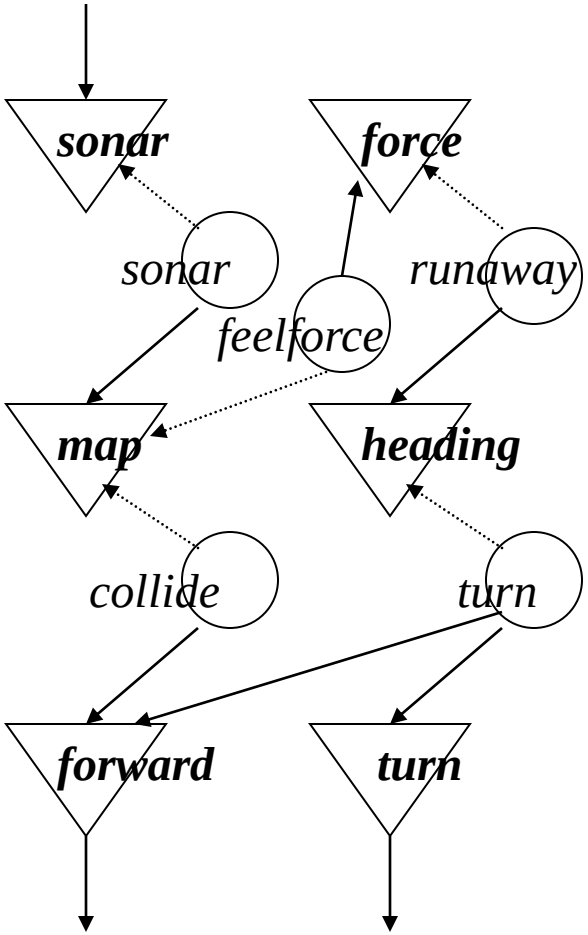


Tým pádom pojdeme rovno až kým nehrozí náraz. Potom sa žačneme vyhýbať, a potom sa opäť pohybujeme rovno.

Pri tomto pohybe sa pomerne ľahko dostaneme do nejakého cyklu, ale na vrstvu AVOID to stačí.

AVOID je hotová.

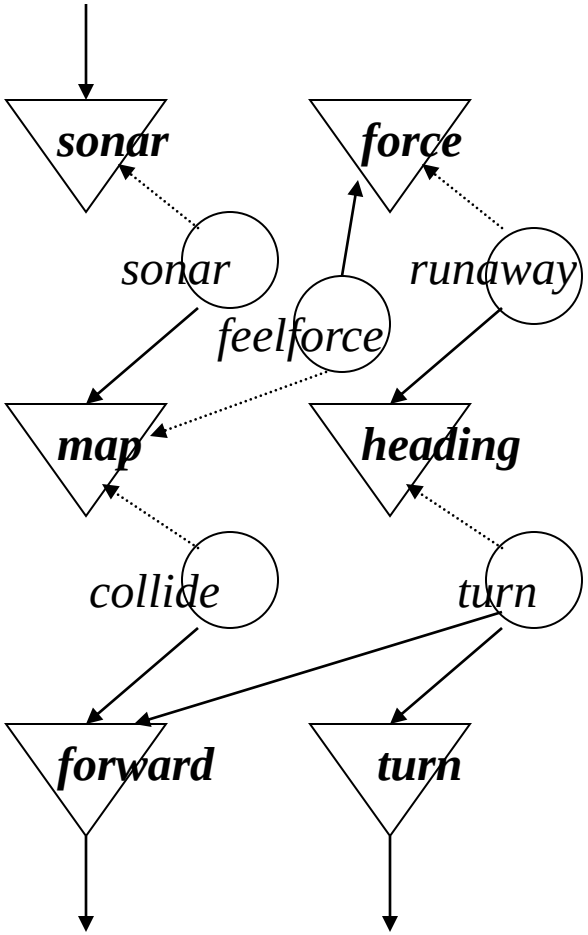
AVOID



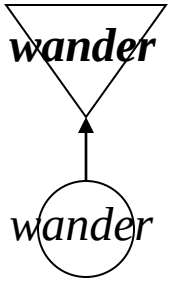
Všimnime si, že realizáciou vyhýbania sa statickým prekážkam, sme zároveň realizovali výbanie sa pohybujúcim objektom, ktoré nebezpečne križujú smer nášho pohybu.

Podobné prípady sa niekedy označujú ako tzv. emergencia.

AVOID



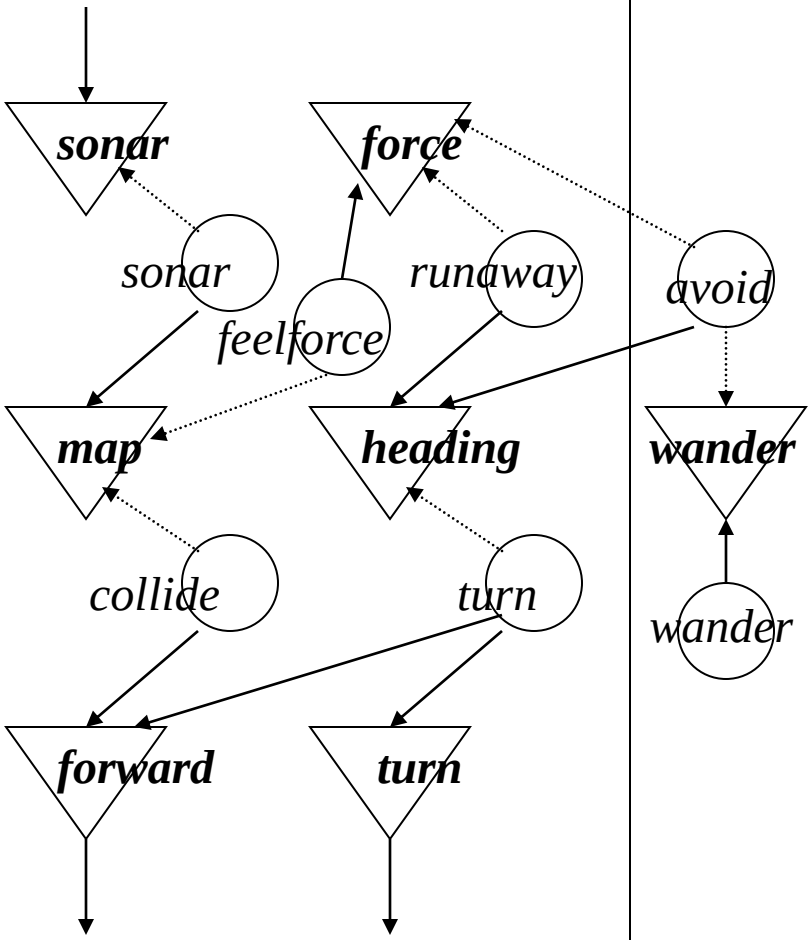
WAN DER



Teraz by sme chceli aby sem-tam porušil prípadné cyklenie náhodným pohybom

Preto agent wander s veľkým sleepom sem-navrhne určitý odklon smeru pohybu

AVOID



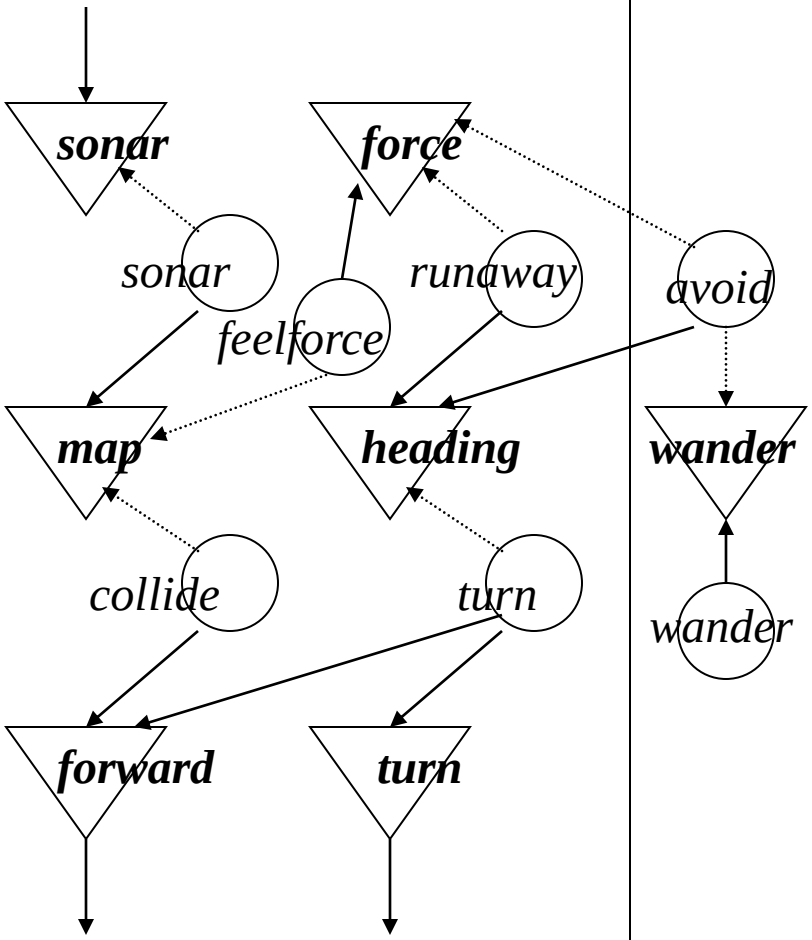
WAN DER

A agent avoid, ak sa cez force presvedčí, že momentálne nehrozí zrážka, prepíše navhrnutým smerom heading, tak ako keby nejaká zrážka hrozila.

Tým pádom využije vyhýbanie sa prekážkam na potulovanie sa

AVOID

WAN
DER



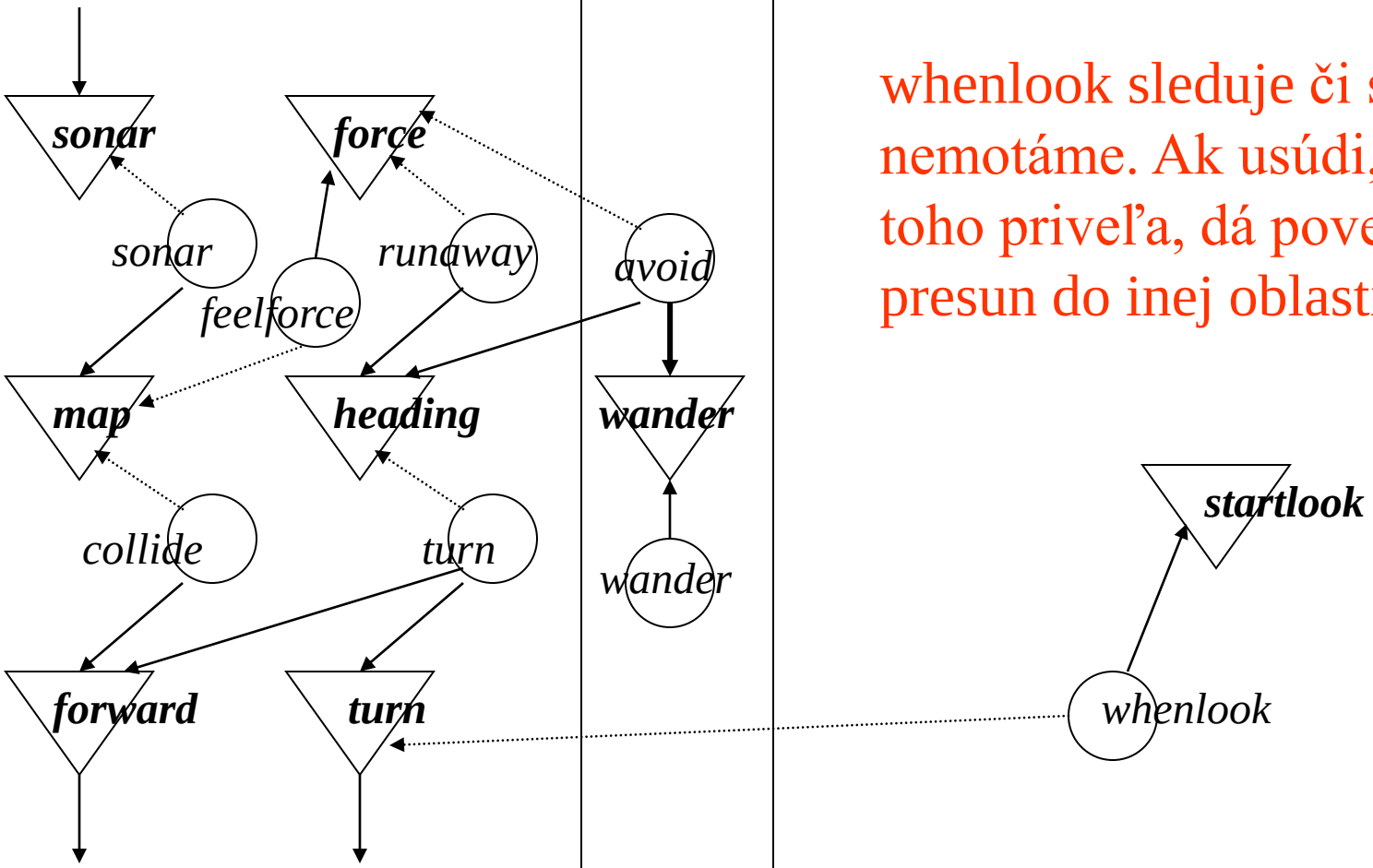
Tým pádom sa nielen vyhýbame prekážkam, ale sa aj potulujeme a pohyb nášho robota už nie je predvídateľný.

Ale zatiaľ sa dosť motá na mieste.

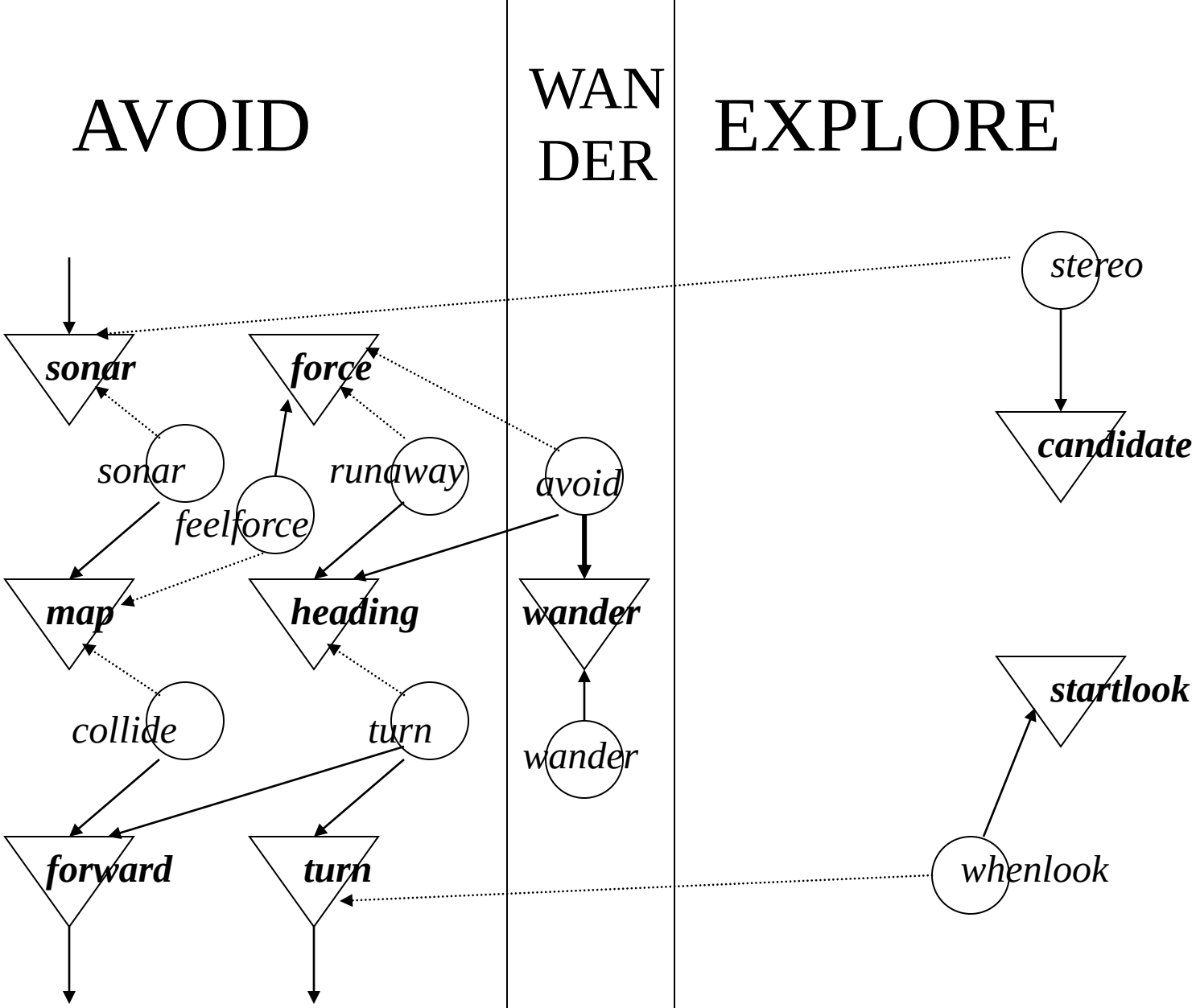
AVOID

WAN DER

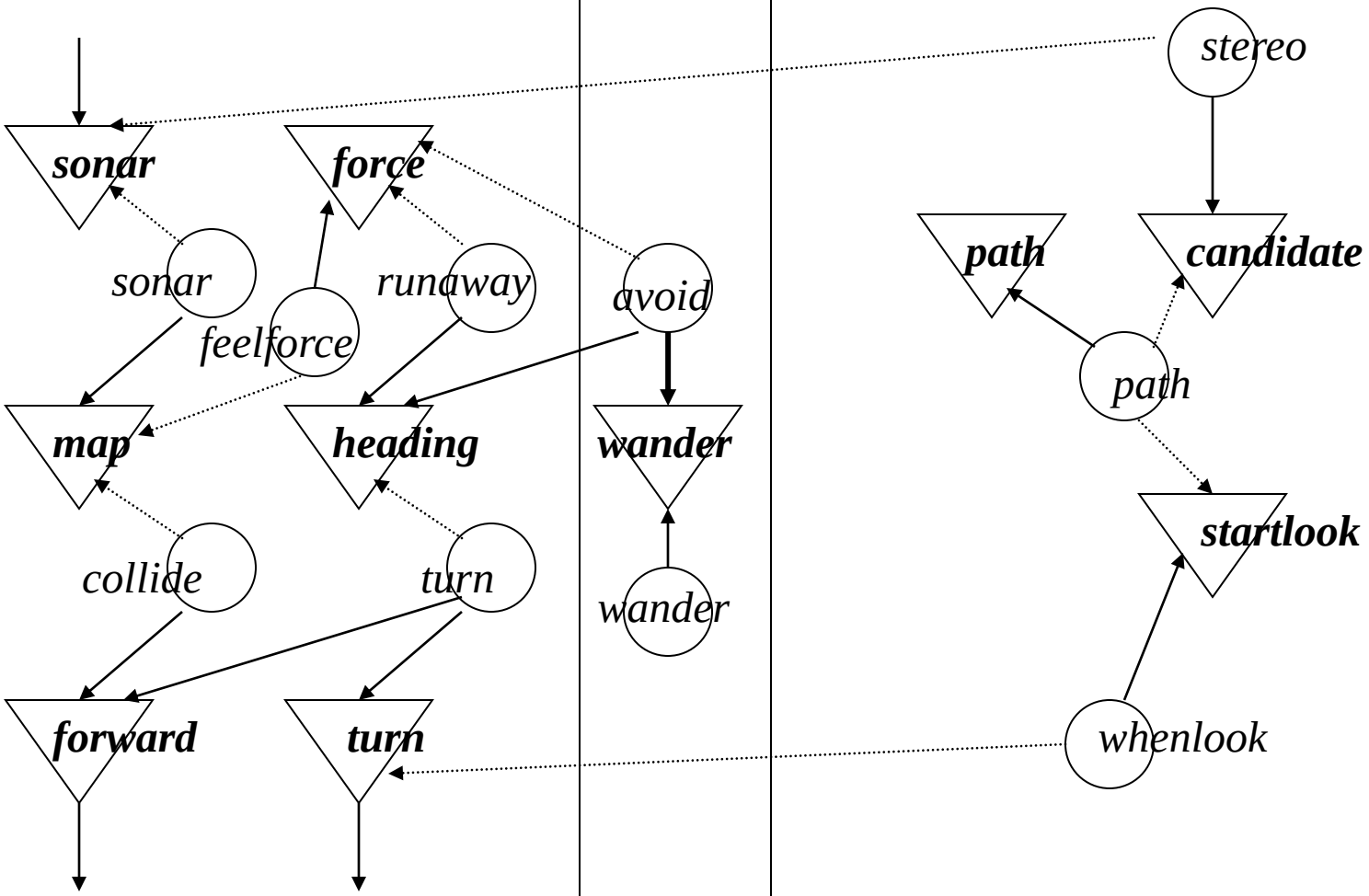
EXPLORE



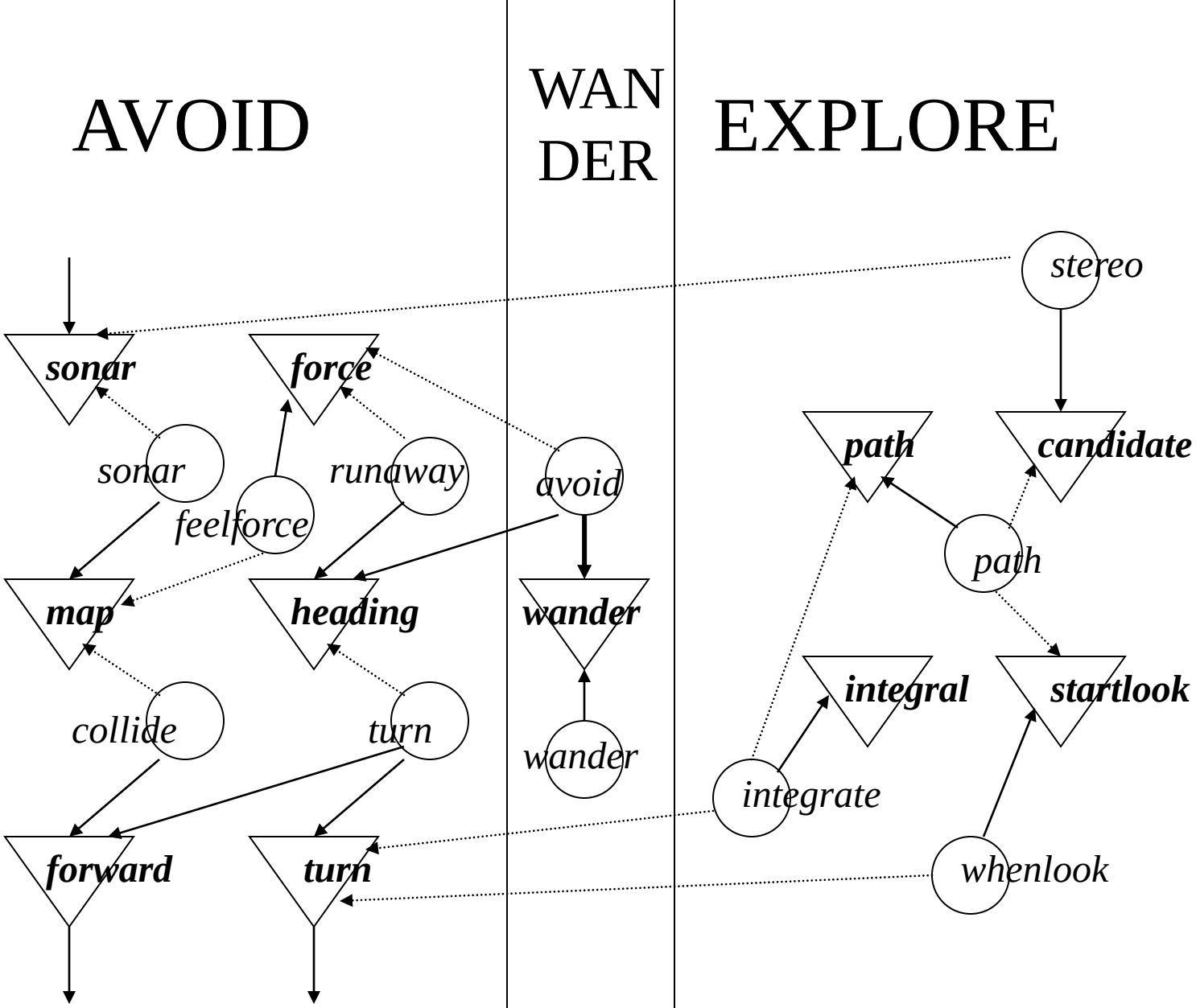
whenlook sleduje či sa nemotáme. Ak usúdi, že už je toho priveľa, dá povel na presun do inej oblasti



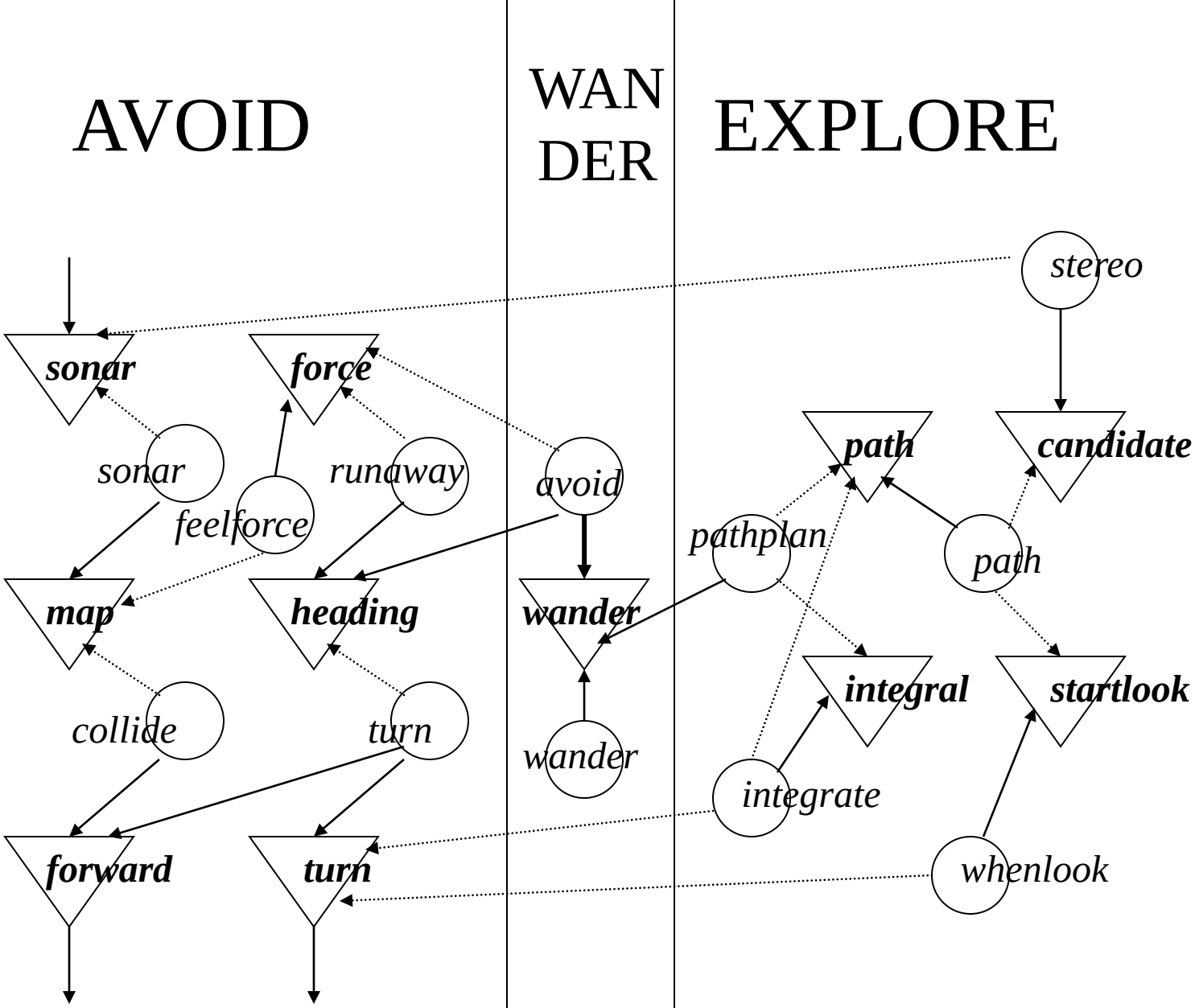
Stereo sleduje či sa na sonare objaví smer v ktorom je voľno



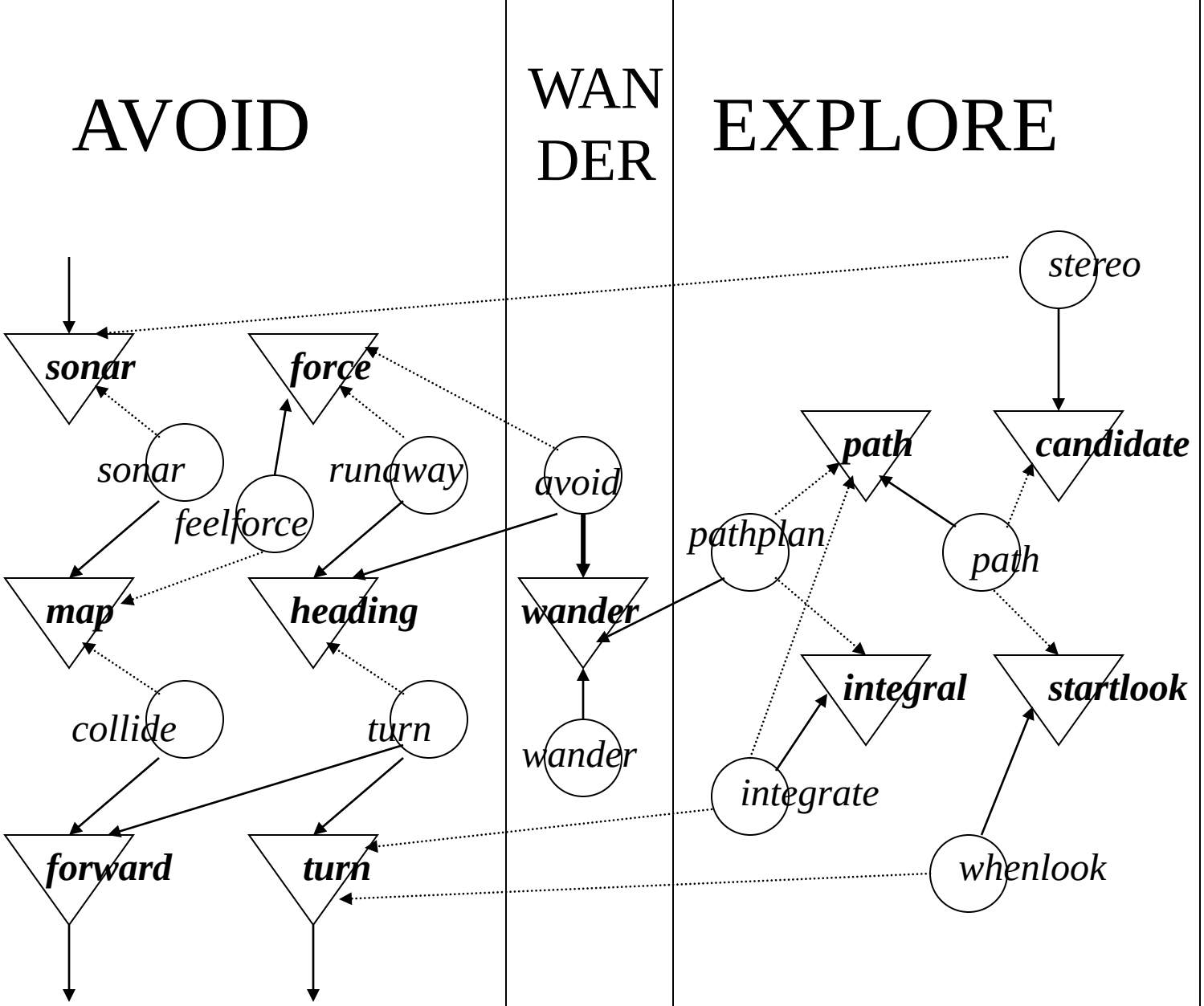
Path vytýči ktorým smerom sa presunie-me do inej oblasti



Je to ale relatívny smer, takže na to aby sme ho mohli absolutne chápať potrebuje-me sumárnu odchýľku

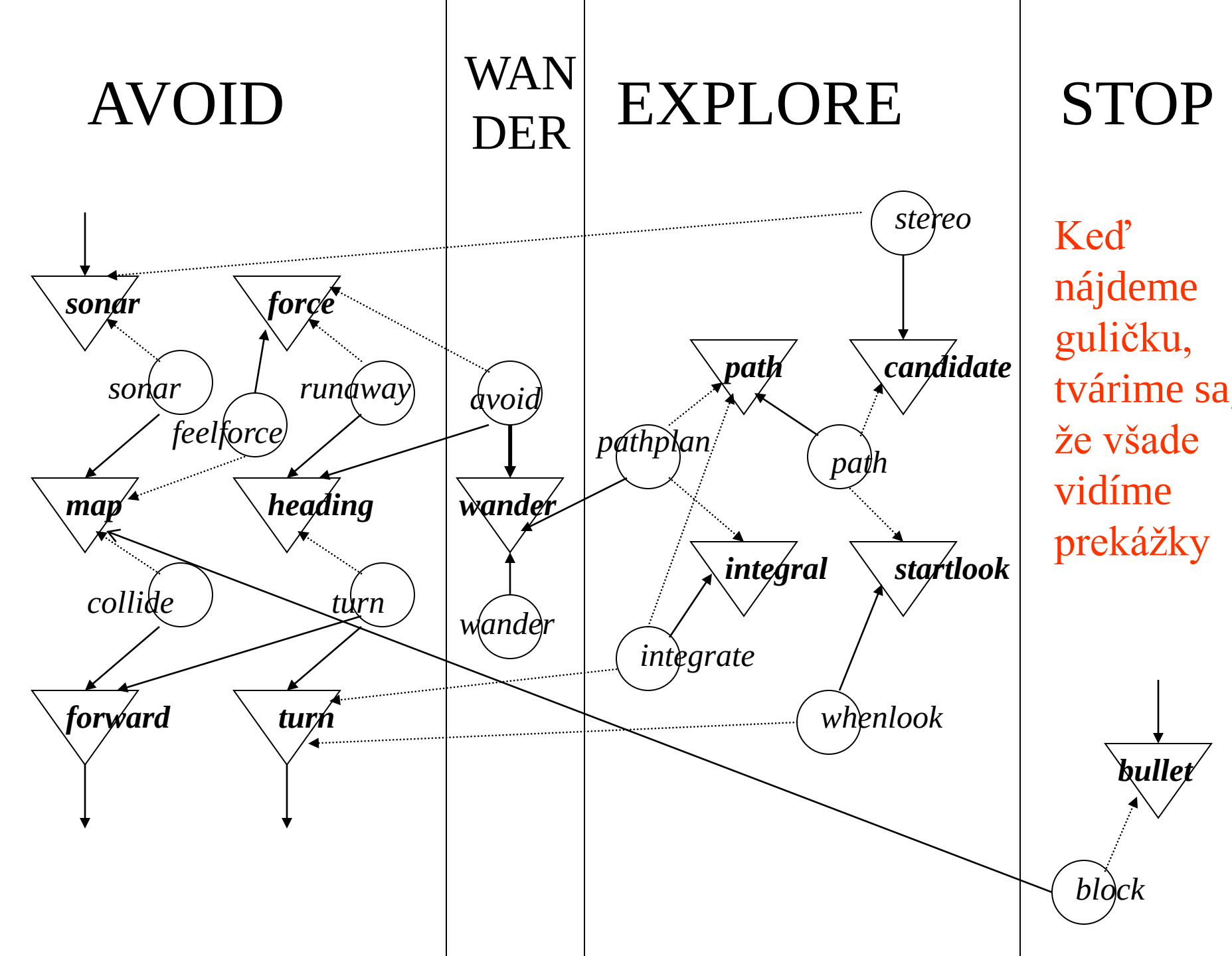


Pathplan v každej chvíli vie o aký uhol sa treba odchýli aby sme dodržali zvolený uhol na presun



Už vieme prechádzať priestorom.

Hoci neuplatňujeme žiadne logicky správne prehľadávanie.



Keď nájdeme guličku, tvárime sa, že všade vidíme prekážky