Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Andoljšek

Reševanje Ravenovih inteligenčnih testov v arhitekturi ACT-R

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: izr. prof. dr. Marko Robnik-Šikonja
Somentor: red. prof. dr. Valentin Bucik

Ljubljana 2014

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jernej Andoljšek, z vpisno številko **63060009**, sem avtor diplomskega dela z naslovom:

Reševanje Ravenovih inteligenčnih testov v arhitekturi ACT-R

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Marka Robnika-Šikonje in somentorstvom red. prof. dr. Valentina Bucika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne: Podpis avtorja:

Zahvaljujem se svojemu mentorju prof. dr. Marku Robniku-Šikonji, za mentorstvo in nasvete pri izdelavi naloge. Prav tako se zahvaljujem somentorju prof. dr. Valentinu Buciku iz oddelka za psihologijo Filozofske fakultete za njegovo pomoč pri dostopu do potrebnega gradiva ter vpogled v delovanje inteligenčnih testov. Na koncu se zahvaljujem še moji družini za vsa leta podpore med mojim študijem.

Kazalo

Povzetek

Kognitivna arhitektura ACT-R omogoča visokonivojsko simuliranje človeških miselnih procesov pri opravljanju raznovrstnih nalog. V preteklih letih je bila spisana peščica modelov, ki simulirajo delovanje uma med reševanjem Ravenovih matričnih inteligenčnih testov, med drugim tudi v arhitekturi ACT-R. Dosedanji modeli pisani v ACT-R niso zmožni rešiti vseh primerov standardiziranih testov Ravenovih progresivnih matric, tudi kadar odstranijo določene hibe človeškega mišljenja, kot je pozabljanje. V ACT-R smo ustvarili model, temelječ na delu Ragni-ja & sodelavcev (2012), na katerem smo poskušali razumeti težave pri reševanju testov, ter ugotoviti, ali je težavnost reševanja določenih primerov posledica omejitev kognitivne arhitekture ali slabosti modela. Zaključimo, da so zahteve po točnosti reševanja problemov, natančnosti simulacije uma in obvladljivosti kode med seboj v konfliktu.

Ključne besede: ACT-R, Ravenove progresivne matrice, računalniško modeliranje mišljenja, inteligenčni testi, kognitivne arhitekture

Abstract

The ACT-R cognitive architecture allows high level simulations of human mental processes during the execution of various tasks. So far a small number of models exist, that simulate the working of the mind when it is solving Raven's progressive matrices intelligence tests. Some of these models use the ACT-R architecture. The existing ACT-R models are incapable of solving all the standardized Raven's progressive matrices tests, even when removing human limitations such as forgetting. We created a model based on the work of Ragni et al. (2012), with which we tried to understand the problems associated with solving these tests. We try to determine whether the difficulties in solving certain tests stem from the limitations of the cognitive architecture, or are perhaps avoidable with modifications of the model. We find that there is a trade-off between a model's performance at solving tests, its accuracy as a simulation of the mind, and the level of its complexity.

Keywords: ACT-R, Raven's progressive matrices, cognitive modelling, intelligence tests, cognitive architectures

Poglavje 1

Uvod

Inteligenčni testi so orodja, s katerimi se da numerično oceniti spretnost ljudi pri opravljanju določenih kategorij miselnih nalog. Ravenovi matrični testi so vrsta inteligenčnih testov, ki ocenjujejo sposobnost luščenja relevantnih informacij iz problemov (*angl.* eductive component of Spearman's g) [9]. Trening reševanja inteligenčnih testov ni prenosljiv na druge situacije - kar pomeni da ne pomaga pri opravljanju ostalih nalog, za katere s testi ocenjujemo nadarjenost. Za kognitivne arhitekture (teorije, ki omogočajo računalniške simulacije človeškega uma) pa je izdelava vedno boljših programov za reševanje tovrstnih testov kljub temu zanimivo raziskovalno področje.

Računalniške simulacije človeškega mišljenja imajo zaenkrat še težave z reševanjem Ravenovih inteligenčnih testov. Eden od najbolj uspešnih modelov je spisan v visoko-nivojski arhitekturi ACT-R (Adaptive Control of Thought - Rational) in je zmožen rešiti več kot devet desetin testiranih primerov iz standardnega in naprednega sklopa Ravenovih inteligenčnih testov [6, str.1]. To pomeni, da se še vedno najdejo primeri, ki jih model ne zna rešiti. Večino primerov v pogosto uporabljenih standardiziranih testih se da rešiti z uporabo relativno preprostega nabora pravil. Vprašamo se, v čem je težava težavnih primerov in kako bi jih lahko rešili.

Namen naloge je izdelati model reševanja Ravenovih testov v arhitekturi ACT-R, proučiti ali ga lahko z uporabo določenih nestandardnih pristopov

izboljšamo in ugotoviti, ali bi tovrstne izboljšave lahko uporabili pri drugih obstoječih modelih. V ta namen poskušamo razumeti, od kod izvirajo težave pri reševanju standardnih testov.

V nadaljevanju v poglavju 2 pogledamo principe kognitivnega modeliranja v okviru arhitekture ACT-R. V poglavju 3 si ogledamo Ravenove inteligenčne teste in pristope k njihovemu reševanju, ki so jih uporabili drugi raziskovalci. V poglavju 4 opišemo našo implementacijo modela reševanja, pred zaključkom v poglavju 6 pa v poglavju 5 še preučimo problematične testne primere.

Poglavje 2

ACT-R kognitivna arhitektura

2.1 Kognitivno modeliranje

Področje kognitivnega modeliranja se ukvarja z ustvarjanjem računalniških modelov inteligentnega mišljenja, skozi katere poskuša boljše razumeti delovanje človeškega uma in napovedati, kako se um odzove pod različnimi pogoji [4, str.57].

Modeli so lahko spisani v okviru ustaljenih kognitivnih arhitektur, ki predstavljajo svoje teorije dojemanja in na podlagi njihovih specifikacij omogočajo simulacije [4, str.57]. Različne teorije uporabljajo različne nabore predpostavk. Arhitekture imajo svoje prednosti in slabosti, npr. ne modelirajo vseh delov človeškega uma. Obstaja več različnih arhitektur, med drugimi poznamo EPIC, Brahms, CHREST, Clarion, SAMPLE, Soar in ACT-R [4, str.57]. Ker gre za teorije, se sčasoma razvijajo in dopolnjujejo, tako da se bolje prilegajo spoznanjem kognitivne znanosti [7]. Glede na to, kako teorije pristopajo k problemu človeškega dojemanja, jih delimo na simbolične in konektivistične [7]. Simbolične teorije k problemu pristopajo od zgoraj navzdol in poskušajo izdelati modele, ki simbole obdelujejo tako, da se obnašanje modela sklada z rezultati poskusov izvedenih z ljudmi. Konektivistične teorije k problemu pristopajo od spodaj navzgor in skušajo na podlagi nevronskih mrež sestaviti modele, sposobne reševanja raznovrstnih nalog.

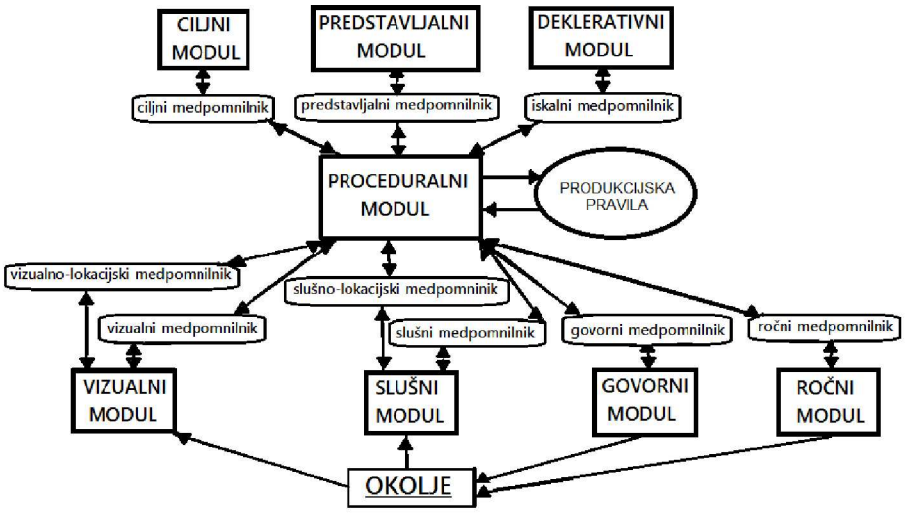
2.2 Arhitektura ACT-R

ACT-R je visokonivojska, simbolična kognitivna arhitektura [1, str.1] [7]. Modeli ACT-R so uporabljeni v več kot sedemsto znanstvenih publikacijah na področjih od kognitivne nevroznanosti do izobraževanja [7]. S pomočjo ACT-R je bilo med drugim modelirano človeško vedenje med vožnjo in letenjem, kar bi lahko pomagalo pri iskanju načinov za preprečevanje nesreč [7].

Prve zasnove arhitekture ACT-R izhajajo iz modela človeškega spomina HAM (Human Associative Memory) Johna. R. Andersona in Gordona Bowera leta 1973, teorija sama pa je bila ustvarjena v zgodnjih devetdesetih letih, z združitvijo modela HAM z Andersonovim modelom spoznavanja, imenovanim racionalna analiza. Od tedaj je doživela vrsto predelav in razširitev. Različica ACT-R4.0 je dodala zaznavne in odzivne sposobnosti po navdihu arhitekture EPIC, ACT-R5.0 pa je restrukturirala arhitekturo na module [7].

Kognitivna znanost govori o možganskih modulih, funkcionalno povezanih enotah v možganih, ki opravljajo specifične naloge. Medtem ko so možganski centri fizično zaokroženi deli možganov, se večina modulov razprostira preko večih centrov, ki si jih delijo z deli drugih modulov. Arhitektura ACT-R se poskuša prilegati organizaciji možganov in je prav tako razčlenjena na module, katerih funkcije v grobem ustrezajo nekaterim prepoznanim modulom v možganih.

Teorija ACT-R se poslužuje koncepta dveh vrst človeškega spomina: deklerativnega in proceduralnega. Deklerativni spomin beleži dejstva, kot je datum našega rojstnega dne ali barva neba. Proceduralni spomin beleži delovne postopke, kot sta plavanje ali vožnja s kolesom. V ACT-R se elementi deklerativnega spomina imenujejo koščki (*angl.* chunks), kar je tudi sicer v kognitivni znanosti izraz za osnovne enote deklerativnega znanja, ki jih lahko držimo v kratkoročnem spominu. Osnovni elementi proceduralnega spomina se v ACT-R imenujejo proceduralna pravila. V najpreprostejši razlagi teorija ACT-R predpostavlja, da kompleksna spoznanja pri ljudeh izhajajo iz interakcij med proceduralnim in deklerativnim znanjem, kjer so aktivacije



Slika 2.1: Arhitektura ACT-R.

proceduralnih pravil statistično utežene pod vplivom okolja [1, str.1].

Arhitektura ACT-R sicer ne določa točnega števila modulov, toda določeni moduli so del osnovne arhitekture. Ti moduli so prikazani na sliki 2.1. Sredi sheme je proceduralni modul, ki v vsakem koraku preveri, katera proceduralna pravila bi se lahko nanašala na trenutno situacijo, izbere najprimernejše pravilo in ga izvrši. Proceduralni modul pri izbiri pravila upošteva stanja medpomnilnikov preostalih modulov. Izvršena produkcijska pravila proceduralnega modula dostopajo do in upravljajo z vsemi ostalimi moduli, po pravilih, ki se razlikujejo med moduli. Spodnji štirje moduli na sliki 2.1 spadajo med zaznavno-gibalne in modelu omogočajo simulacijo interakcije z zunanjim okoljem. Vizualni in slušni modul lahko pobereta informacije iz simuliranega zunanjega sveta, govorni in ročni pa lahko nanj vplivata. Preostali moduli so deklerativni modul, ki hrani deklerativno znanje modela, ciljni modul, ki predstavlja del spomina, ki beleži kaj trenutno počnemo, ter predstavljalni modul, ki omogoča ustvarjanje novih koščkov [2, str.2].

2.3 Implementacija ACT-R 6.0

Kognitivne teorije ne moremo enačiti z njeno implementacijo, ki se lahko od osnovne teorije precej razlikuje [7]. ACT-R 6.0 je najnovejša in najširše uporabljana implementacija teorije, spisana v programskem jeziku Lisp. Poleg implementacije osnovnih modulov arhitektura omogoča razširitev z dodatnimi. Zaradi omejitev implementacije lahko določene programske funkcije, kot sta ura ali generator naključnih števil v model vključimo le kot dodatne module, čeprav z njimi ne prestavljamo delov človeškega uma. Implementacija določa osnovne interakcije med okoljem in zaznavno-gibalnimi moduli. Osnovno simulacijo okolja lahko razširimo glede na potrebe modela, vendar moramo paziti na stvari, ki jih arhitektura ne specificira in niso simulirane. ACT-R je visokonivojska arhitektura in tako specificira obstoj vizualnega modula in kako ta modul komunicira z drugimi, ne specificira pa točno, kakšne vrste podatkov je sposoben izluščiti iz okolja in kako. Nizkonivojskih procesov, ki so odgovorni za to, model ne obravnava, tako da brez razširitev ne specificiramo, kakšna vizualna informacija je na voljo v okolju, temveč katere informacije vizualni modul razbere iz okolja.

Tu bomo na kratko opisali primer modela spisanega v ACT-R, ki je prikazan na sliki 2.2. Model *sestevanje* sešteje števili ena in dva, ki sta na začetku podani kot del cilja. V vrstici 5 so definirani splošni parametri modela, pod njimi so definirane vrste koščkov, ki jih uporabljamo v tem modelu, pod njimi pa je v bloku add-dm definirana vsebina deklerativnega modula ob zagonu modela. Z goal-focus nastavimo prvi cilj modela, ki ga model sam pobere iz deklerativnega modula. Na koncu definiramo še dve proceduralni pravili. Vsa proceduralna pravila v okolju ACT-R so bloki kode oblike IF-THEN. Pogoji se nahajajo nad dolgo puščico (==>), pod njo pa napišemo opravila. Ko proceduralni modul opravi eno opravilo, se posveti naslednjemu. To je lahko katerokoli opravilo, čigar pogoji so v tistem trenutku izpolnjeni. Če so izpolnjeni pogoji večih pravil, proceduralni modul s postopkom razreševanja konfliktov izbere eno izmed njih. V našem primeru je vrednost atributa korak pri začetnem cilju “zacetek”, kar je pogoj le enega od pravil,

```
(clear-all) ;počisti za drugimi modeli

(define-model sestevanje

(sgp :esc t :lf .05 :needs-mouse nil :trace-detail high)

(chunk-type sestevanje korak num1 num2 focus)
(chunk-type mat-dejstvo funkcija num1 num2 num3)
(chunk-type iskani-rezultat num)  ;v ta košček bomo shranili rezultat, vendar bi ga
;v deklerativnem spominu hitro izgubili brez dodatnih oznak na kaj se to število nanaša

(add-dm                          ;deklerativni spomin
(prvi-cilj ISA sestevanje korak "zacetek" num1 1 num2 2)
(ISA mat-dejstvo funkcija "sestevanje" num1 1 num2 1 num3 2)
(ISA mat-dejstvo funkcija "sestevanje" num1 1 num2 2 num3 3))
;za splošno uporabo bi morali vnesti veliko množico tovrstnih pravil: vsaj za števila 0-9
;za seštevanje večjih števil bi morali dodati tudi druga proceduralna pravila

(goal-focus prvi-cilj) ;nastavi prvi cilj v modelu

(p p-sestevanje-inicializacija ;definicija proceduralnega pravila
    =goal>                  ;preverjanje vrednosti v medpomnilniku ciljnega modula
    ISA         sestevanje  ;preden se sklicujemo na lastnosti koščka, moramo vedno navesti njegov tip
    korak       "zacetek"   ;vrednost atributa korak se mora biti niz "sestevanje"
    num1        =var1       ;v atributu num1 mora biti prisotna neka vrednost
    num2        =var2       ;ta vrednost se priredi spremenlljivki
    ;nad puščico so pogoji za izbiro tega pravila; t.i. leva stran (angl. LHS) proceduralnega pravila
    ==>
    ;pod puščico so navodila; t.i. desna stran (angl. RHS) proceduralnega pravila
    =goal>                  ;sprememba vrednosti atributov koščka v ciljnem medpomnilniku
    korak       "izracun"   ;vrednost se spremeni v "izracun"
    +retrieval>             ;naročilo iskalnemu medpomnilniku deklerativnega modula, naj poišče košček
    ISA         mat-dejstvo ;tip iskanega koščka mora biti vedno podan
    funkcija    "sestevanje";vrnjen košček bo ustrezal tem pogojem (če spomin najde tak košček)
    num1        =var1       ;vrednost v atributu num1 mora ustrezati vsebini spremenljivke =var1 z LHS
    num2        =var2
)

(p p-sestevanje-izracun
    =goal>
    ISA         sestevanje
    korak       "izracun"
    =retrieval>
    ISA         mat-dejstvo
    num3        =var1
    ?imaginal>              ;preverimo lastnosti predstavljalnega medpomnilnika
    state free              ;ta procedura se ne sproži dokler je zaseden

    ==>

    =goal>
    korak       "izracunano"
    +imaginal>                  ;naročilo predstavljalnemu medpomnilniku naj ustvari nov košček
    ISA         iskani-rezultat ;tip koščka
    num         =var1           ;določimo vrednost za lastnost, ni obvezno definirati vrednosti za vse
    ; vrednost se bo prestavila v deklerativni pomnilnik najkasneje ob naslednjem naročilu temu modulu
))
```

Slika 2.2: Primer seštevanja v ACT-R.

tako da konflikta nimamo. Izbrano je proceduralno pravilo p-sestevanje-inicializacija, ki spremeni vrednost atributa korak v ciljnem medpomnilniku, ter naroči da se iz deklerativnega modula v njegov medpomnilnik prenese matematično dejstvo tipa seštevanje, z atributoma seštevancev, kar skopiramo iz ciljnega medpomnilnika. Sedaj so izpolnjeni pogoji za izbiro pravila p-sestevanje-izracun, ki se s svojimi zahtevami prepriča, da je predstavljalni modul trenutno prost. V tem modelu ga sicer nismo uporabljali, vendar je na splošno prav, da se izogibamo tovrstnim konfliktom. Iz medpomnilnika deklerativnega modula poberemo iskano vrednost vsote in predstavljalnemu pomnilniku naročimo naj ustvari nov košček tipa iskani-rezultat, v katerega to vrednost shranimo. Zopet spremenimo tudi atribut korak, zato niso več izpolnjeni pogoji za izvršitev kateregakoli pravila. Iskani rezultat ostane v predstavljalnem medpomnilniku, vendar bi ga bilo od tam preprosto premakniti v deklarativni modul.

Poglavje 3

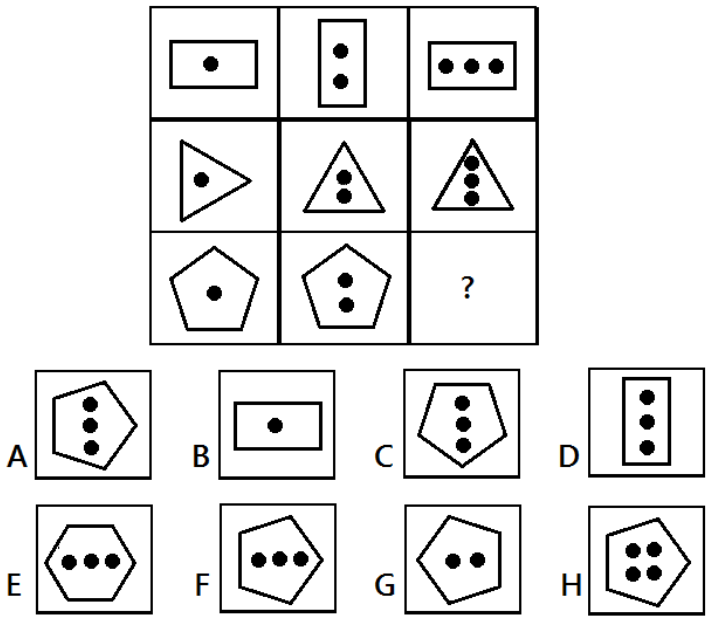
Inteligenčni testi in Ravenov test

3.1 Inteligenčni testi

Inteligentnost je dober napovedovalec uspeha pri izobrazbi in delu [8]. Inteligenčni testi so standardizirani testi, ki ocenjujejo inteligentnost. Ker je inteligentnost kompleksen konstrukt, je možno ocenjevati in testirati različne aspekte inteligence. Obstaja hierarhija dejavnikov, ki deli inteligentnost na deset širokih in sedemdeset ozkih zmožnosti. Sodobni testi inteligentnosti pogosto podrobneje ocenjujejo več različnih sposobnosti, čeprav ne zajemajo vseh. Velik del testov določi končno splošno oceno IQ (inteligenčni količnik), ki je močno korelirana z rezultati drugih IQ testov in inteligentnostjo na splošno. Danes razvitim testom je skupno, da se mediana rezultatov normirnega vzorca reševalcev postavi kot IQ 100 in dve tretjini populacije padeta na območje 85-115 [8] .

3.2 Zakaj je Ravenov test zanimiv

Ravenovi testi so najbolj pogosta in popularna vrsta inteligenčnega testa za starostne skupine od pet let do starostnikov [9]. Test je neverbalen in ne zah-



Slika 3.1: Primer Ravenovega matričnega testa.

teva predhodnih znanj, zato ga je mogoče uporabiti pri skoraj vsaki družbeni skupini. Zaradi teh lastnosti je zanimiv tudi za kognitivno modeliranje.

Vsak testni primer (z izjemo nekaterih poskusnih, ki so manjši) prikazuje mrežo dimenzije 3 x 3, kjer je deveta celica prazna, vsebine ostalih celic pa sledijo določenemu vzorcu. Naloga reševalcev testa je ugotoviti, kakšna bi morala biti vsebina devete celice, da bi ustrezala vzorcu. Rešitev se običajno izbere iz seznama možnosti. Primeri se med seboj razlikujejo po težavnosti. Sklopi primerov se uporabljajo kot inteligenčni testi ter so namenjeni testiranju različnih skupin ljudi - tako obstaja npr. standarden sklop testov za osnovnošolske otroke. Pri lažjih primerih se celice med seboj razlikujejo samo v eni lastnosti, pri težjih pa se lahko tudi po precej več.

Slika 3.1 prikazuje primer tovrstnega testa.

3.3 Pristopi k reševanju Ravenovih testov

Identificirana je bila množica pravil, s pomočjo katerih se da rešiti veliko večino primerov v standardnih sklopih Ravenovih matričnih testov [3, str.18]. Za vsako posamezno lastnost, ki se med celicami razlikuje, je potrebno le uporabiti ustrezno pravilo, da določimo, kakšna mora biti ta lastnost v rešitvi. V primeru na sliki 3.1 se celice med seboj razlikujejo po likih, po smeri likov in po številu pik v vsakem liku. Ravnamo lahko glede na sledečih pet pravil:
1. Konstanta v vrsti: lastnost je lahko konstantna pri vseh elementih vrstice ali stolpca. Na sliki so liki v vsaki vrsti enaki - tako lahko na konecu tretje vrstice pričakujemo peterokotnik. Poleg tega število pik ostaja konstantno po stolpcih, zato lahko v rešitvi pričakujemo tri pike.
2. Številčno napredovanje: lastnost enakomerno raste med prvo, drugo in tretjo celico vrstice ali stolpca. Na sliki se število pik poveča za eno na vsakem prehodu med celicami v isti vrstici. Tako lahko pričakujemo, da se bo število pik v deveti celici za ena večje kot v osmi.
3. Razporeditev treh elementov: v vsakem stolpcu ali vrstici imamo razporeditev istih treh vrednosti. Na sliki se to zgodi z liki v stolpcih in v zadnjem stolpcu nam manjka peterokotnik.
4. Seštevek polj: vsebina celic v tretjem stolpcu ali vrstici je vsota ali ostanek prejšnjih dveh. V našem primeru je število pik v celicah tretjega stolpca seštevek pik ustreznih celic v prvih dveh stolpcih.
5. Razporeditev dveh vrednosti: v vsaki vrstici ali stolpcu je enaka razporeditev istih dveh vrednosti, npr. ena vrednost se pojavi enkrat, druga pa dvakrat. Na sliki to vidimo pri orientaciji likov, kjer je v prvih dveh vrsticah en lik obrnjen drugače od ostalih dveh. Enako velja za te v prvih dveh stolpcih. Za tretji stolpec in vrstico tako vidimo, da bo obrnjen lik v zadnji celici.

Ljudje se pri reševanju Ravenovih testov običajno poslužujemo enega od dveh pristopov: primere poskušamo rešiti z iskanjem vizualnih vzorcev ali z upoštevanjem logičnih relacij [5, str.3]. V praksi se izkaže, da prvi pristop

ne more rešiti zahtevnejših primerov v standardiziranih inteligenčnih testih [5, str.3]. Lahko pa na ta način rešimo primer, pri katerem so skozi več celic potegnjene neprekinjene črte. Takega primera ni mogoče rešiti z množico logičnih pravil. Uspeh logičnih pravil tako ne pomeni, da so zmožna rešiti vse primere, le primerna so za nabor primerov uporabljen v težjih testih.
Kljub temu so pravila izvrstno izhodišče za izgradnjo modela za reševanje Ravenovih testov, če imamo opravka s simboli. Za primere, pri katerih je potrebno prepoznavanje vizualnih vzorcev so bolj primerni modeli zasnovani na konektivističnem pristopu.

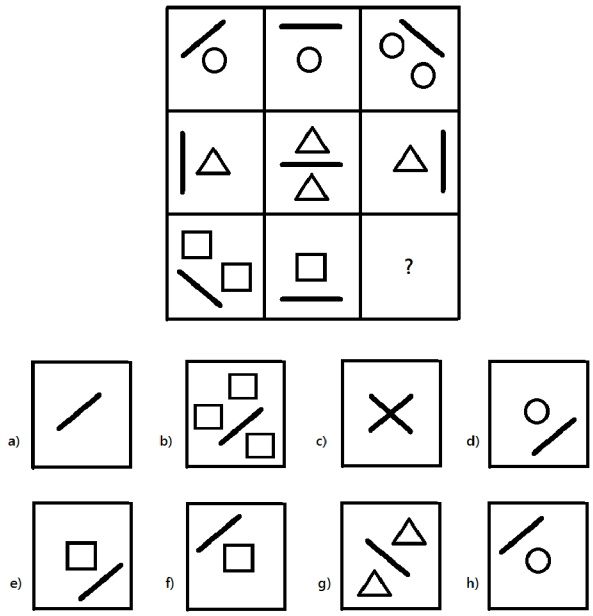
Poglavje 4

Reševanje Ravenovega testa z ACT-R

Ustvarjeni model simulira "list papirja", na katerem je testni primer Ravenove matrike. To matriko pretvori v notranjo obliko, kjer kot vizualni objekt obravnavamo vsako posamezno celico matrike in vse možne rešitve. Lastnosti posameznih likov v vsaki celici so v našem modelu atributi celice. Posamezni liki so opisani z atributi velikost, višina in širina (kadar se spreminjata relativno glede na velikost), število stranic, x in y relativna pozicija v celici, plast (višinska), rotacija (v stopinjah), oblika (uporabljamo jo tudi za beleženje obstoja - lik brez definirane oblike za naš model ne obstaja), stil črt (odebeljene, črtkane, ukrivljene, ...) in število enakomerno razpostavljenih, a sicer popolnoma identičnih likov.
Za atribute se uporabi interpretacija, ki načeloma omogoča rešitev problema (več o tem bomo povedali v nadaljevanju). Del interpretacije je ureditev atributov, ki vsebuje informacijo o tem, kateri objekt v drugi celici najverjetneje ustreza določenemu objektu v prvi. V modelu je omogočena simulacija nedeterminističnih procesov znotraj posameznih modulov, vendar je pozabljanje koščkov v deklerativnem spominu izklopljeno. Ker se osredotočamo na celotne celice, parametra za omejitev števila vidnih koščkov, na katere smo pozorni, ne spreminjamo.

Model se najprej horizontalno in po potrebi vertikalno sprehodi čez matriko, da določi število likov v zadnji celici. Pri sprehodu uporablja pogosto strategijo, pri kateri istočasno gleda en objekt, se spominja drugega in primerjavo začasno hrani v predstavljalnem medpomnilniku kot del delovne rešitve. Če se število različnih likov od celice do celice spreminja, poskusi uporabiti pravilo seštevka polj, kjer je ena od celic v vrstici ali stolpcu seštevek ostalih dveh, ali pa številčnega napredovanja, kjer število elementov v eno smer narašča. Zatem se horizontalno sprehodi čez celice in si sekvenčno ustvarja rešitev za vsak atribut vsakega lika. Če ne more aplicirati pravila konstante, poskusi s pravili seštevka polj in številčnega napredovanja. Na koncu poskusi še s pravili razporeditve dveh ali treh vrednosti. Rezultate si sproti beleži v košček, ki je namenjen hranjenju rešitve. Sproti si pri vsakem atributu beleži prepričanost v dobljeno vrednost. Če za katero od iskanih vrednosti ne more dobiti gotove vrednosti z uporabo standardnih pravil, se čez matriko sprehodi še vertikalno. Zatem primerja dobljeno rešitev z navedenimi možnimi rešitvami. V idealnem primeru že ima rešitev, ki se ujema z eno od ponujenih, ali pa se z eno rešitvijo ujema v atributih, v katerih vrednosti je prepričan. Zadnja možnost ni pogosta.

Če naš model soočimo s primerom na sliki 4.1, najprej po vrsticah preveri, ali je v vsaki celici prisotno enako število različnih likov. Kot liki v tem primeru štejejo debelejše črte sredi celic, krogi, trikotniki in kvadrati. V celici, v katerih imamo dva kroga, sta ta zabeležena kot en lik s pozicijo med njunima dejanskima postavitvama. Tako imamo v prvi celici množico krogov, v kateri je en sam krog, v tretji celici pa množico, v kateri sta dva. Model na ta način določi, da sta v vsaki celici prisotna dva različna lika in zabeleži, da rešitev prav tako sestoji iz dveh različnih likov. Model nato začne ugotavljati vrednosti lastnosti za vsak lik. Pri tem se poslužuje vnaprej definiranega zaporedja, definiranega v deklerativnem spominu. Pri resnični osebi bi to pomenilo, da se je na pamet naučila zaporedja, po katerem potem pregleduje lastnosti objektov za vsak objekt zapovrstjo. Privzamemo, da je kot prvi lik v interpretaciji problema definiran tisti sredi vsake celice. Model



Slika 4.1: Primer Ravenovega matričnega testa.

najprej zanima oblika, za katero ugotovi, da je v prvi vrstici konstantna. Ker je konstantna tudi v drugi in tretji vrstici, določi da je oblika prvega objekta v rešitvi kvadrat, tako kot preostala lika v tretji vrstici. Za število članov v množici pri prvem liku konec prve vrstice določi, da gre verjetno za seštevanje. V prvi in drugi celici imamo namreč po en lik, v tretji pa dva. Konec druge vrstice ugotovi, da ta ne sledi istemu vzorcu. Preveri, če vzorec prejšnje vrstice lahko ustreza tudi čemu drugemu. Vzorec ne ustreza enakomerno rastočim vrstam, lahko pa je distribucija dveh vrednosti. Tudi vzorec v drugi vrstici ustreza distribuciji dveh vrednosti, zato to postane nova predpostavka o spreminjanju vrednosti te lastnosti. V tretji vrstici določi, da bo prvi lik rešitve (ki je štirikotne oblike) del množice, ki vsebuje en sam lik. Za ostale vrednosti, kot so pozicija, lokacija, rotacija, tekstura lika in debelina črte drugo za drugo preveri, da so pri tem liku skozi celo matriko v vsaki vrstici konstantne. V enakem zaporedju preveri iste lastnosti tudi pri drugem liku (odebeljeni črti). Pri njej določi, da je v različnih celicah rotirana za različno

število stopinj, pri čemer gre v vsaki za vrstici za numerično napredovanje. Po tem pravilu določi, da je lik v rešitvi za 135° rotiran glede na lik v osmi celici. Zaplete se, ko določa relativno pozicijo črte v vsaki celici. V prvi vrstici so relativni odmiki višine zabeleženi kot 2, 1 in 2 za posamezne celice, kar tedaj sicer ustreza vzorcu distribucije dveh vrednosti, po drugi vrstici pa ne ustreza več nobenemu poznanemu vzorcu. Model se vrne v prvo celico in poskuša najti poznan vzorec razporeditve s pregledovanjem stolpcev. Najde enakomerno naraščajoča zaporedja, s katerimi lahko določi tudi vrednost te lastnosti v rešitvi. Če tudi po pregledu stolpcev ne bi našel rešitve, bi si model to zabeležil in si kasneje poskusil pomagati z rešitvami. Ko model konča z določanjem vseh lastnosti vseh likov v rešitvi, si drugo za drugo pogleda vse možne rešitve. Pri vsaki po vrsti primerja vse lastnosti ugotovljene in ponujene rešitve. Zavrne vse rešitve, s katerimi se ne strinja glede lastnosti, katere vrednost je ugotovil sam. Prvo rešitev zavrne na podlagi števila likov v njej. Drugo zavrne na podlagi števila kvadratov. Tretjo zavrne, ker namesto črte pričakuje kvadrat. Enako zavrne četrto rešitev. Peto rešitev zabeleži kot ustrezno.

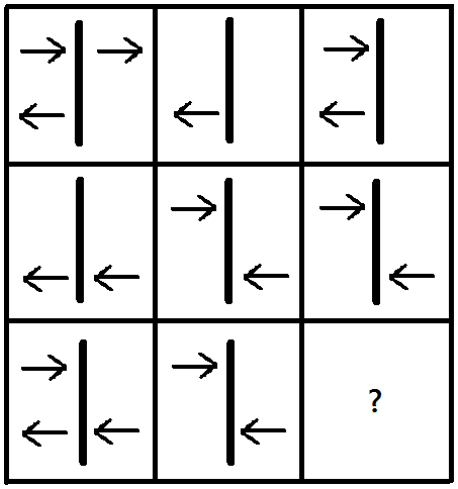
Poglavje 5

Evalvacija

Model je testiran na ustreznih interpretacijah osnovnih in naprednih sklopov standardiziranih Ravenovih matričnih inteligenčnih testov, ki so bile izpisane iz testnih primerov z dovoljenjem za uporabo v raziskovalne namene. Ilustracije v tej nalogi sicer niso del standardiziranih testov.

Od 36 primerov v sklopu imenovanem standardne Ravenove progresivne matrice (SPM) jih model reši 28. Od 36 primerov v sklopu imenovanem napredne Ravenove progresivne matrice (APM) jih model reši 20. Skupna uspešnost na preizkušenih primerih je tako 67%. To zaostaja za modelom (Ragni & sod., 2012), katerega uspešnost na istem naboru problemov znaša 91%. Naš model je po uspešnosti reševanja bolj primerljiv z modelom (Cirrilo in Ström, 2010), ki pri reševanju SPM prav tako reši osemindvajset problemov [6, str.1].

Model ima precej težav predvsem s primeri, ki odstopajo od primerov rešljivih z osnovnimi pravili. Tako model pri ugotavljanju števila likov na sliki preveri možnost seštevanja likov dveh celic v tretjo, medtem ko lahko primer namesto funkcije ALI zahteva npr. funkcijo IN, ali pa celo dve različni funkciji, ki se uporabita na dveh podskupinah likov. V nekaterih primerih se vrednosti atributov rotirajo preko več različnih objektov. Pri seštevanju se



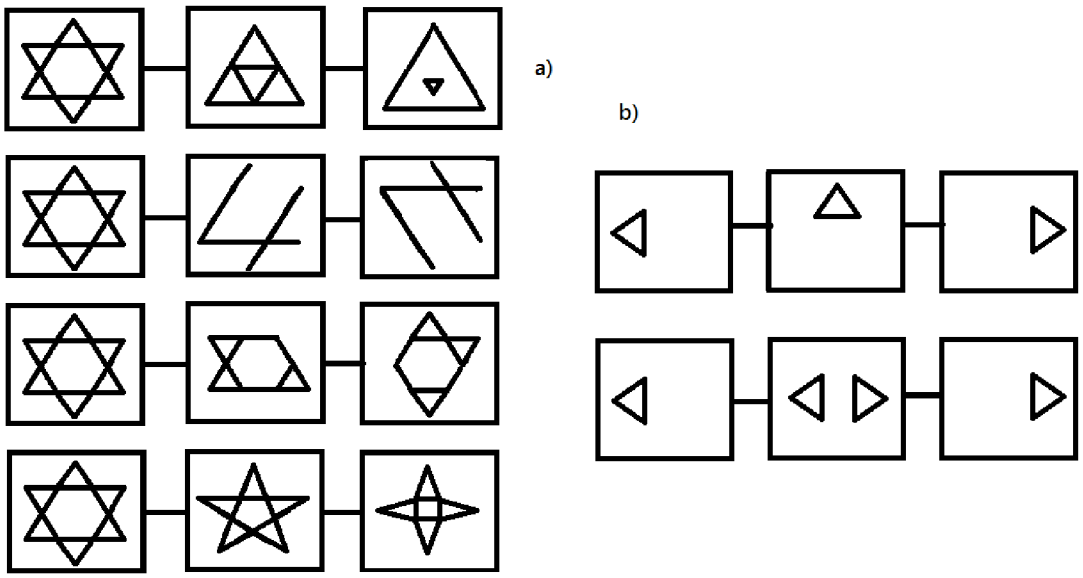
Slika 5.1: Primer Ravenovega matričnega testa.

v nekaterih primerih liki prištevajo ali odštevajo od vsote glede na vrednost enega od preostalih atributov. Tovrstni primeri so pogostejši v naprednem sklopu in v zadnjem delu osnovnega sklopa ter ilustrirajo, da uporaba petih pravil ne more rešiti vseh primerov. Tak primer si lahko ogledamo na sliki 5.1, kjer je prisotnost puščic v tretjih celicah stolpcev in vrstic funkcija njihove prisotnosti v prvih dveh. Funkcija za puščice obrnjene proti sredini posameznih celic je pri tem ALI, za obrnjene proč od sredine pa IN.

Model bi lahko izboljšali z implementacijo dodatnih pravil, ki naslavljajo te probleme, vendar bi bil model še vedno usmerjen k reševanju standardiziranih testov, ki ne predstavljajo vseh možnih primerov, ki bi se jih dalo zastaviti. Dodajanje dodatnih pravil ni preprosto, saj zahteva spremembo poteka določanja, katero pravilo je v dani situaciji najprimernejše. Ker se mora biti model med pregledovanjem zmožen vrniti, da lahko preveri, ali se uporabljajo druga pravila, to pomeni kompleksno strukturo z večjo verjetnostjo, da v določenih primerih model uporabi napačno pravilo. Ljudje v tovrstnih situacijah običajno poskušamo uporabiti več strategij. Pri tem se učimo, katere strategije so verjetneje uspešne in prilagodimo njihove prioritete v nadaljnjem reševanju. Tovrstno učenje je v ACT-R mogoče simulirati,

a ne vodi do ene same dobro načrtovane strategije.

Potrebno je omeniti, da je zaradi narave okolja ACT-R reševanje problemov v njem morda lažje, kot bi moralo biti. ACT-R ne simulira procesov, s katerimi vizualni modul obdela vidne podatke. Ti procesi so skupaj enako kompleksni kot ACT-R sam [2, str.3]. Ker model spisan v ACT-R deluje s "pravilno" interpretacijo vizualne informacije, se s tem izogne problemu iskanja le-te. To je za ljudi, ki rešujejo teste, pomemben del problema. Spomnimo se, da je sposobnost interpretacije informacij to, kar Ravenovi testi ocenjujejo. Pri precejšnjem delu primerov je del možnih rešitev zasnovan tako, da bi reševalce napeljal k napačni interpretaciji problema, v podani pravilni interpretaciji pa te namenoma zavajajoče rešitve pravilnemu rezultatu sploh niso podobne. Kot vidimo na sliki 5.2, je pravilna interpretacija vsebine celice za namen reševanja problema lahko močno odvisna od vsebine drugih celic. Zvezd v levih celicah primera a) ni smiselno vedno interpretirati kot en lik, niti ne kot zbirko črt. Glede na to kaj je v celicah, ki ji sledita, je zvezdo lahko bolj smiselno zapisati kot dva trikotnika, kot šest črt, ali pa kot šesterokotnik in šest trikotnikov. Pri primeru b) moramo trikotnik v zadnji celici vrste razumeti kot prestavljen in zavrten trikotnik iz prve celice, ali pa kot drug trikotnik. Tu je pravilna interpretacija različna glede na vsebino vmesne celice zaporedja. Med reševanjem primerov mora človeški um po potrebi preklapljati med več možnimi interpretacijami podatkov - kot preklop med različnimi pogledi na Neckerjevo kocko, vizualno iluzijo pri kateri um črte risbe interpretira v dveh različnih tridimenzionalnih konfiguracijah, a le v eni naenkrat. V ACT-R bi bilo to možno simulirati s preklapljanjem med različnimi nabori opisov objektov v vidnem polju, vendar bi tu večja korektnost simulacije prav tako otežila izdelavo dobro delujoče rešitve.



Slika 5.2: Različne interpretacije istih likov.

Poglavje 6

Sklepne ugotovitve

V okviru naloge smo se seznanili s kognitivno arhitekturo ACT-R in s sestavljanjem programov za, iz računalniškega vidika, precej nekonvencionalno platformo. V arhitekturi smo implementirali model reševanja Ravenovih inteligenčnih testov in se seznanili s problemi, s katerimi se soočajo kognitivni modeli v tej arhitekturi.

Za naš model so možne izboljšave, ki bi implementirale potrebna pravila za reševanje dodatnih primerov - npr. dodatne logične funkcije in ustrezne prilagoditve za razlikovanje med njimi in ostalimi vzorci. Smiselno bi bilo tudi poskusiti model bolj približati človeškim strategijam razmišljanja med reševanjem testov.

Človeški um ne deluje po pristopu od zgoraj navzdol, zato je na ta način težko narediti njegovo dobro delujočo simulacijo. Glede na teorijo ACT-R naše miselne procese vodi veliko število proceduralnih pravil, ki se jih učimo celo življenje. Modeli v kognitivni arhitekturi ACT-R zato nudijo zanimiv pogled v visokonivojske procese človeškega mišljenja.

Literatura

[1] J. R. Anderson, "ACT A Simple Theory of Complex Cognition", American Psychologist, 1996, Vol. 51, No. 4, 355-365

[2] J. R. Anderson, M. D. Byrne, S. Douglass, C. Lebiere, Y. Qin, "An Integrated Theory of the Mind" Psychological Review, 2004, Vol. 111, No. 4, 1036–1060

[3] P. A. Carpenter, M. A. Just, P. Shell, "What One Intelligence Test Measures: A Theoretical Account of the Processing in the Raven Progressive Matrices Test", Psychological Review, 1990, Vol.97 , No. 3, 404-431

[4] Dr. Ir. Niels, C. C. M. Moes, "Digital Human Models", Delft University of Technology Faculty of Industrial Design Engineering, 2010

[5] J. R. Kirby, M. J. LAWSON, "Effects of Strategy Training on Progressive Matrices Performance", Contemporary Educational Psychology, 1983, No. 8, 127- 140

[6] M. Ragni, S. Neubert, "Solving Raven's IQ-tests: An AI and cognitive Modelling Approach", ECAI Press, 2012

[7] "ACT-R',' Dostopno na https://en.wikipedia.org/wiki/ACT-R [Zadnji ogled: 6.10.2014]

[8] "Intelligence quotient", Dostopno na https://en.wikipedia.org/wiki/IQ [Zadnji ogled: 6.10.2014]

[9] "Raven's Progressive Matrices", Dostopno na https://en.wikipedia.org/wiki/ACT-R [Zadnji ogled: 6.10.2014]