Ogrodje paketa za avtomatsko reševanje logičnih ugank

Robert Meolic, Tatjana Kapus, Zmago Brezočnik
Fakulteta za elektrotehniko, računalništvo in informatiko
Univerza v Mariboru
Smetanova ulica 17, 2000 Maribor, Slovenija
{meolic,kapus,brezocnik}@uni-mb.si

A framework for automatic solving of logic riddles

We study a sort of logic riddles given as a set of facts from which other facts must be derived. These problems, usually stated for entertainment, can be easily solved if there is only a small number of possible solutions. However, they become much more complex when the number of facts and possible solutions grows. The problem motivated us to think about a framework for automatic solving using computer. It is based on a propositional calculus. The most complex part of the system is parsing and understanding of the facts stated in a natural language as well as their transformation into mathematical formulae.

1 Uvod

V članku obravnavamo posebno vrsto logičnih ugank, pri katerih moramo na osnovi danih dejstev poiskati nova. Srečamo jih v razvedrilnih revijah ter na tekmovanjih osnovnošolcev in srednješolcev. Tekmovalci jih ponavadi rešujejo s pomočjo logičnih razpredelnic. Dokler je število možnih rešitev majhno, takih ugank ni težko rešiti. A težavnost se z večanjem števila možnih rešitev hitro povečuje. Naš cilj je zgraditi sistem za avtomatsko reševanje z računalnikom, katerega vhodni podatki bi bili čim bližje opisu z naravnim jezikom.

Ker gre za reševanje logičnih problemov, se lahko močno naslonimo na programski jezik prolog. Interpreter za prolog v bistvu ves čas rešuje podoben problem, ko na osnovi danih ali že dokazanih izrekov ovrednoti nov izrek oz. poišče veljavne izreke. Po drugi strani pa za rešitev takšne uganke ne potrebujemo tako zmogljivega ogrodja, kot je prolog. V članku prikazujemo, da lahko problem rešimo že v okviru izjavnega računa, če ne potrebujemo razlage rešitve v obliki zaporedja sklepanj.

Paketi za učinkovito obdelavo izjavnega računa z računalnikom so se v zadnjih desetih letih zelo razvili. To je predvsem posledica uvedbe nove podatkovne strukture, imenovane binarni odločitveni graf ("Binary Decision Diagram", BDD) [1]. Na FERI smo razvili lasten paket BDD [5], ki ga že vrsto let uporabljamo pri reševanje problemov s področja formalne verifikacije sistemov.

2 Primer naloge in postopka reševanja

Logične uganke temeljijo na opisu z naravnim jezikom. Zato njihovo reševanje poleg logičnega sklepanja zahteva tudi razumevanje danega besedila. Obstaja nevarnost slabo zastavljenih trditev, ki niso enoumne. V nadaljevanju bomo podali primer logične uganke in nakazali postopek reševanja, po katerem bi lahko takšne naloge avtomatično reševal računalnik.

Imamo tri fante z imeni **Andrej (**A**)**, **Borut (**B**)**, **Cene (**C**)**. Eden je iz **Celja (**ce**)**, drugi iz **Maribora (**mb**)** in tretji iz **Ljubljane (**lj**)**. Po velikosti so **majhen (**s**)**, **večji (**l**)** in **največji (**x**)**. V oklepajih smo navedli okrajšave, ki jih bomo pozneje uporabili v matematičnih formulah. Naloga zahteva, da s pomočjo danih trditev ugotovimo, iz katerega mesta prihajajo fantje in kako so razvrščeni po velikosti. Trditve so naslednje:

1. Andrej ali Borut je iz Maribora.
2. Največji je iz Ljubljane, najmanjši pa ni iz Celja.
3. Če je Andrej manjši od Ceneta, potem Andrej ni iz Maribora.
4. Cene je večji od tistega iz Celja.

Za človeka je takšna uganka dokaj enostavna, saj je potrebno le logično sklepanje. Verjamemo, da bo večina bralcev podano nalogo rešila brez pretiranega naprezanja sivih celic v nekaj minutah. Iz druge trditve na primer hitro ugotovimo, da je, ker najmanjši ni iz Celja, pa tudi največji ni iz Celja (je iz Ljubljane), iz Celja tisti, ki je srednje velik, najmanjši pa je torej iz Maribora. Mnogo težje, kot rešiti dano nalogo, je narediti računalniški program, ki bi razumel besedilo in samostojno izpeljal pravilne sklepe. Pri tem pa ni problem računanje in povezovanje sklepov, saj je računalnik že v osnovi logičen stroj, ampak je predvsem težko pretvoriti trditve iz naravnega jezika v take matematične formule, ki se bodo lahko avtomatično obdelale.

V članku prikazujemo enega od možnih načinov reševanja, ki temelji na obdelavi izrazov izjavnega računa in ki se ga da dokaj učinkovito avtomatizirati z računalnikom. Izpuščene so podrobnosti glede predstavitve in obdelave izrazov izjavnega računa.

Množice lastnosti, kot so na primer ime, kraj in velikost, imenujmo *kategorije*. Eno od kategorij (npr. ime) izberimo in jo imenujmo *glavna kategorija*. Elementi kategorije so posamezne lastnosti, kot na primer "ime mu je Andrej", "prihaja iz Maribora" itd. Elemente glavne kategorije imenujmo *osebki*, podane trditve pa naj bodo *pogoji*. V **prvem koraku reševanja** tvorimo *osnovne izjave*, ki so sestavljene iz dveh lastnosti, od katerih mora biti ena osebek iz glavne kategorije. Tukaj so primeri:

- Amb (Andrej je iz Maribora),
- Bl (Borut je srednje velik),
- cex (ni osnovna izjava, ker ne vsebuje osebka),
- $Ambx$ (ni osnovna izjava, ker vsebuje več kot dve lastnosti).

Vsaka osnovna izjava ni nujno pravilna. V obravnavanih logičnih ugankah veljata naslednja dva izreka:

1. Vsak osebek ima eno lastnost iz vsake kategorije.
2. Nobena dva osebka nimata iste lastnosti.

Označimo z znakom $*$ konjunkcijo osnovnih izjav. Glede na prvi izrek je konjunkcija določenih osnovnih izjav enaka nič. Na primer: $Amb * Ace = 0$, $Bs * Bx = 0$ itd. Drugi izrek bomo pozneje postavili kot omejitev pri reševanju. Konjunkcijo osnovnih izjav, ki določa vse lastnosti vseh osebkov, imenujmo *rešitev*. Primer rešitve je konjunkcija $Ace * As * Bmb * Bl * Clj * Cx$.

V **drugem koraku reševanja** pretvorimo pogoje v matematične formule. To je bistven in zelo težak korak, saj gre za razumevanje naravnega jezika. Da bi omogočili avtomatično pretvorbo, se je potrebno omejiti na uporabo vnaprej definiranih fraz v jeziku. Primera raziskav na tem področju sta pretvarjanje slovenščine v formule predikatne logike prvega reda [3] in projekt The Alvey Natural Language Tools [2], v okviru katerega nastaja splošnonamenski morfološki in semantični analizator za angleški jezik. Slednjega na Univerzi v Edinburgu uporabljajo pri formalni verifikaciji sistemov za pretvarjanje lastnosti formalno opisanih sistemov v formule temporalne logike [4]. Pogoje iz našega primera lahko pretvorimo v naslednje izraze izjavnega računa, v katerih poleg konjunkcije uporabljamo še operatorje za negacijo (!), implikacijo ($\rightarrow$) in disjunkcijo ($+$):

$$P1 \triangleq Amb + Bmb$$

$$P2a \triangleq (Ax \rightarrow Alj) * (Bx \rightarrow Blj) * (Cx \rightarrow Clj)$$

$$P2b \triangleq (As \rightarrow !Ace) * (Bs \rightarrow !Bce) * (Cs \rightarrow !Cce)$$

$$P3 \triangleq (As * Cl + As * Cx + Al * Cx) \rightarrow !Amb$$

$$P4 \triangleq (Ace * As + Bce * Bs) * (Cl + Cx) + (Ace * Al + Bce * Bl) * Cx$$

Kot vidimo, smo drugi pogoj razdelili v dva izraza, saj v resnici vsebuje dve neodvisni trditvi. Pri avtomatičnem razpoznavanju so takšne sestavljene trditve eden od problemov, ki ga je potrebno učinkovito rešiti.

V **tretjem koraku reševanja** postavimo omejitve za rešitev, ki izhajajo iz izreka, da ima lahko vsako lastnost le en osebek (npr. Andrej in Borut ne moreta biti oba iz Maribora). Omejitve podamo v obliki izrazov, ki so precej dolgi (pri n osebkih potrebujemo za vsako kategorijo disjunkcijo toliko konjunkcij, kolikor je različnih kombinacij z n elementi), vendar jih lahko zaradi njihove enostavne stukture z računalnikom hitro tvorimo. V danem primeru uporabimo naslednji dve omejitvi:

$$O1 = Ace * Bmb * Clj + Ace * Blj * Cmb + Amb * Bce * Clj + Amb * Blj * Cce + Alj * Bce * Cmb + Alj * Bmb * Cce$$

$$O2 = As * Bl * Cx + As * Bx * Cl + Al * Bs * Cx + Al * Bx * Cs + Ax * Bs * Cl + Ax * Bs * Cl$$

V **četrtem koraku reševanja** naredimo enostavno konjunkcijo med vsemi omejitvami in pogoji. Dobljen izraz nato predstavimo dvonivojsko kot vsoto rešitev. Če so pogoji protislovni in naloga nima rešitve, dobimo rezultat 0. Za našo nalogo dobimo rezultat, ki vsebuje eno samo rešitev:

$$R = O1 * O2 * P1 * P2a * P2b * P3 * P4 = Ace * Bmb * Clj * Al * Bs * Cx$$

3 Uporaba predikatnega računa

Ker je pisanje pogojev v obliki izrazov izjavnega računa dokaj neintuitivno opravilo, predlagamo, da se pretvorba naravnega jezika v matematične formule izvede v dveh korakih. V prvem koraku na osnovi razumevanja besedila tvorimo izraze predikatnega računa prvega reda. V drugem koraku jih nato pretvorimo v ustrezne izraze izjavnega računa. Idejo bomo prikazali na prej obravnavanem primeru naloge.

Z znakom $\mathcal{S}$ označimo množico vseh osebkov. Vpeljimo predikata *kraj* in $velikost$, ki vsakemu osebku pripišeta njegov kraj oz. velikost. Dodatno vpeljimo še predikat $vecji$, ki opisuje urejenost osebkov glede na velikost. Pogoje iz primera lahko potem zapišemo s predikatnim računom na naslednji način:

$$\mathbf{P}1 \triangleq kraj(A, mb) + kraj(B, mb)$$

$$\mathbf{P}2a \triangleq \forall X \in \mathcal{S} : (velikost(X, x) \rightarrow kraj(X, lj))$$

$$\mathbf{P}2b \triangleq \forall X \in \mathcal{S} : (velikost(X, s) \rightarrow !kraj(X, ce))$$

$$\mathbf{P}3 \triangleq vecji(C, A) \rightarrow !kraj(A, mb)$$

$$\mathbf{P}4 \triangleq \forall X \in \mathcal{S} : (kraj(X, ce) \rightarrow vecji(C, X))$$

Zapisane izraze glede na omejeno število osebkov in lastnosti razširimo tako, da izločimo predikat *vecji* in vse kvantifikatorje. V danem primeru uporabimo zvezi:

$$vecji(X,Y) = velikost(X,l) * velikost(Y,s) + velikost(X,x) * velikost(Y,s) + velikost(X,x) * velikost(Y,l)$$

$$(\forall X \in \mathcal{S} : F) = F|_{X=A} * F|_{X=B} * F|_{X=C}$$

Če upoštevamo, da ima lahko vsak osebek le eno lastnost iz posamezne kategorije, je mnogo konjunkcij v dobljenem rezultatu enakih 0. Na koncu namesto predikatov *kraj* in *velikost* zapišemo ustrezne osnovne izjave. Za ilustracijo postopka je tukaj primer razširitve izraza $\mathbf{P}4$:

$$\mathbf{P}4 \triangleq \forall X \in \mathcal{S} : (kraj(X,ce) \rightarrow vecji(C,X)) \equiv$$
$$\forall X \in \mathcal{S} : (kraj(X,ce) \rightarrow (velikost(C,l) * velikost(X,s) + velikost(C,x) * velikost(X,s) + velikost(C,x) * velikost(X,l))) \equiv$$
$$(kraj(A,ce) \rightarrow (velikost(C,l) * velikost(A,s) + velikost(C,x) * velikost(A,s) + velikost(C,x) * velikost(A,l))) * (kraj(B,ce) \rightarrow (velikost(C,l) * velikost(B,s) + velikost(C,x) * velikost(B,s) + velikost(C,x) * velikost(B,l))) * (kraj(C,ce) \rightarrow (velikost(C,l) * velikost(C,s) + velikost(C,x) * velikost(C,s) + velikost(C,x) * velikost(C,l))) \equiv$$
$$!Ace * !Cs * !Cl * !Bce * !Cce +$$
$$!Ace * !Cs * !Cl * Bce * !Bl * !Cce * Cx * Bs +$$
$$!Ace * !Cs * !Cl * Bce * Bl * !Cce * Cx +$$
$$!Ace * !Cs * Cl * !Bce * !Cce +$$
$$!Ace * !Cs * Cl * Bce * !Bl * !Cce * Bs +$$
$$!Ace * Cs * !Cl * !Bce * !Cce +$$
$$Ace * !Cs * !Al * !Cl * !Bce * !Cce * Cx * As +$$
$$Ace * !Cs * !Al * !Cl * Bce * !Bl * !Cce * Cx * As * Bs +$$
$$Ace * !Cs * !Al * !Cl * Bce * Bl * !Cce * Cx * As +$$
$$Ace * !Cs * !Al * Cl * !Bce * !Cce * As +$$
$$Ace * !Cs * !Al * Cl * Bce * !Bl * !Cce * As * Bs +$$
$$Ace * !Cs * Al * !Cl * !Bce * !Cce * Cx +$$
$$Ace * !Cs * Al * !Cl * Bce * !Bl * !Cce * Cx * Bs +$$
$$Ace * !Cs * Al * !Cl * Bce * Bl * !Cce * Cx = \mathbf{P}4'$$

Dobljen izraz $\mathbf{P}4'$ je različen od prej uporabljenega $P4$. Vendar sta za uporabo enakovredna, ker so zaradi omejitev uganke nepomembne tiste konjunkcije, v katerih je iz Celja več kot eden oz. v katerih ni iz Celja noben osebek. V rezultatu lahko tudi izpustimo vse negirane člene.

$\mathbf{P}4'$ ob upoštevanju omejitev $O1$ in $O2 \equiv$
$$!Ace * !Cs * !Cl * Bce * !Bl * !Cce * Cx * Bs +$$
$$!Ace * !Cs * !Cl * Bce * Bl * !Cce * Cx +$$
$$!Ace * !Cs * Cl * Bce * !Bl * !Cce * Bs +$$
$$Ace * !Cs * !Al * !Cl * !Bce * !Cce * Cx * As +$$
$$Ace * !Cs * !Al * Cl * !Bce * !Cce * As +$$
$$Ace * !Cs * Al * !Cl * !Bce * !Cce * Cx \equiv$$
$$Bce * Bs * Cx + Bce * Bl * Cx + Bce * Bs * Cl +$$
$$Ace * As * Cx + Ace * As * Cl + Ace * Al * Cx =$$
$$(Ace * As + Bce * Bs) * (Cl + Cx) +$$
$$(Ace * Al + Bce * Bl) * Cx = P4$$

Ker ima vsak osebek natanko eno lastnost iz vsake kategorije, lahko pri izražanju pogojev uporabimo univerzalnostni ali eksistencialnostni operator. V prikazanem primeru smo vedno izbrali univerzalnostni operator. Z uporabo eksistencialnostnega operatorja bi na primer zadnji pogoj podali takole:

$$\mathbf{P}4 \triangleq \exists X \in \mathcal{S} : (kraj(X,ce) * vecji(C,X))$$

Eksistencialnostni operator v primeru dane naloge pretvorimo s pomočjo naslednje zveze:

$$(\exists X \in \mathcal{S} : F) = F|_{X=A} + F|_{X=B} + F|_{X=C}$$

Kot rezultat sicer dobimo izraz, ki je različen od tistega pri uporabi univerzalnostnega operatorja, vendar sta za uporabo pri reševanju naloge enakovredna.

Včasih se zdi, da je pri oblikovanju pogoja eksistencialnostni operator nujno potreben, a to ni res. Imejmo na primer pogoj: *Vsaj eden od fantov je manjši od Andreja*. Če natančno povzamemo dano besedilo, dobimo naslednji izraz:

$$\mathbf{P} \triangleq \exists X \in \mathcal{S} : (vecji(A,X))$$

Navedeni pogoj pa pravzaprav trdi le to, da Andrej ni najmanjši, in ga lahko podamo tudi brez eksistencialnega operatorja:

$$\mathbf{P} \triangleq velikost(A,l) + velikost(A,x)$$

4 Zaključek

V članku je predstavljeno ogrodje sistema za avtomatsko reševanje določene vrste logičnih ugank. Sistem temelji na uporabi izjavnega računa. Predlagani pristop ni nujno najbolj učinkovit način reševanja takih nalog, vendar pa učinkovit namenski algoritem niti ni naš cilj. Bolj nas zanima splošen, matematični pristop, ki se ga bo pozneje dalo razširiti in uporabiti na področju formalne specifikacije in verifikacije sistemov.

Predstavljeni postopek je dobro razdelan in vse korake razen razumevanja in avtomatske pretvorbe naravnega jezika v matematične formule se da učinkovito izvesti le s sorazmerno enostavnim računanjem. Na področju razumevanja naravnega jezika, ki se mu bomo v prihodnosti najbolj posvetili, vidimo naslednje možne rešitve:

- uporaba dobro definirane podmnožice naravnega jezika, pri čemer posameznim besedam in frazam omejimo obliko, pomen in možnost pojavljanja,
- vnašanje pogojev na osnovi vnaprej definiranih fraz oz. vzorcev, ki jih uporabnik v celoti ali po delih izbere iz pripravljenih menujev,
- interaktivno vnašanje pogojev s pomočjo dialogov, ki jih vodi računalnik.

Einsteinova uganka (Einstein's Riddle)
`http://csaba.org/TechNotes/Einstein/`

Albert Einstein je menil, da 98% ljudi ni sposobnih rešiti spodnjega problema. Preizkusite se!

Dejstva:

- V ulici je 5 hiš različnih barv.
- V vsaki hiši živi oseba drugačne narodnosti.
- Vsak od teh 5 ljudi pije drugačno pijačo, kadi drugačne cigarete in ima drugačno domačo žival.

Gremo:

- Britanec živi v rdeči hiši.
- Šved ima psa.
- Danec pije čaj.
- Zelena hiša je levo poleg bele hiše.
- Lastnik zelene hiše pije kavo.
- Oseba, ki kadi Pall-mall, ima papagaja.
- Lastnik rumene hiše kadi Dunhill.
- Človek, ki živi v hiši na sredini, pije mleko.
- Norvežan živi v prvi hiši.
- Človek, ki kadi Blend, živi zraven človeka, ki ima mačko.
- Človek, ki ima konja, živi poleg človeka, ki kadi Dunhill.
- Človek, ki kadi Blue Master, pije pivo.
- Nemec kadi Prince.
- Norvežan živi poleg modre hiše.
- Človek, ki kadi Blend, ima soseda, ki pije vodo.

Vprašanje: KDO IMA RIBO?

Uspešno razvit sistem bo lahko dobro izkoristil zmožnosti računalnikov, ki zmorejo pomniti in preračunati ogromne količine podatkov. Tak sistem bo, podobno kot pri igranju šaha, resno konkuriral in tudi presegel človekove zmožnosti obvladovanja in reševanja zapletenih logičnih ugank. Da ta cilj ni tako oddaljen, se prepričamo, če si ogledamo priloženo Einsteinovo uganko (podobna naloga je tudi t. i. *The Zebra Problem*). Ko jo bo računalnik razumel in znal samostojno rešiti, bo vsaj glede na izjavo, ki jo pripisujejo slavnemu fiziku, že krepko presegel sposobnosti večine njegovih sodobnikov. Avtomatično rešitev Einsteinove uganke smo zato določili tudi kot pomemben mejnik pri razvoju sistema in upamo, da ga bomo dosegli.

Za konec pa je tu še nekaj spletnih strani, ki so povezane z obravnavanimi logičnimi ugankami:

- http://www2.arnes.si/ ikralj/
- http://torina.fe.uni-lj.si/∼izidor/logic/index.html
- http://cernet.g-kabel.si/oblika/o_meni.htm

Literatura

[1] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient Implementation of a BDD Package. V *The Proceedings of the 27th ACM/IEEE Design Automation Conference*, str. 40–45, 1990.

[2] J. Carroll. *Practical Unification-Based Parsing of Natural Language*. Doktorska disertacija, University of Cambridge, UK, 1993.

[3] P. Holozan. Računalniško prevajanje iz slovenščine v logiko prvega reda. *Logika in razvedrilna matematika*.

[4] A. Holt. Formal verification with natural language specifications: guidelines, experiments and lessons so far. *South African Computer Journal*, 24:253–257, 1999.

[5] R. Meolic. Preverjanje pravilnosti obnašanja sistemov s sočasnostjo. Magistrsko delo, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija, 1999.