Programske paradigme

— Programiranje ograničenja —

Milena Vujošević Janičić

Matematički fakultet, Univerzitet u Beogradu

Sadržaj

1 Uopšteno o programiranju ograničenja

1.1 Definicija programiranja ograničenja

Programiranje ograničenja (engl. *constraint programming*)

- Opšti problem programiranja ograničenja predstavlja se sistemom ograničenja nad upravljačkim (nepoznatim) promenljivama. Zadatak je naći dopustivo rešenje, odnosno odrediti vrednosti promenljivih koje zadovoljavaju sva postavljena ograničenja, ili dokazati da takvo rešenje ne postoji.
- Programiranje ograničenja je deklarativna programska paradigma
- Za razliku od imperativne paradigme, gde se postupak pronalaženja rešenja svakog problema zadaje u koracima, kod programiranja ograničenja, ne zadaje se postupak već se postavljaju uslovi koje promenljive moraju da ispunjavaju.

Semantičke razlike

- Ograničenja se razlikuju od ograničenja u imperativnoj paradigmi. Na primer, $x < y$ u imperativnoj paradigmi se evaluira u tačno ili netačno, dok u paradigmi ograničenja zadaje relaciju između objekata x i y koja mora da važi.
- Ograničenja mogu da budu različitih vrsta, na primer, ograničenja iskazne logike (A ili B je tačno), linearna ograničenja ($x \leq 15$), ograničenja nad konačnim domenima...

1.2 Oblast primene programiranja ograničenja

Primene

- Programiranje ograničenja je savremen pristup rešavanju teških kombinatornih problema. U skladu sa time, ima primene pre svega u operacionim istraživanjima, tj. u rešavanju kombinatornih i optimizacionih problema

Primer

Pekara *Kiflica*
Pekara *Kiflica* proizvodi hleb i kifle. Za mešenje i pečenje hleba potrebno je 10 minuta, dok je za kiflu potrebno 12 minuta. Testo za hleb sadrži 300g brašna, a testo za kiflu sadrži 120g brašna. Zarada koja se ostvari prilikom prodaje jednog hleba je 7 dinara, a prilikom prodaje jedne kifle je 9 dinara. Ukoliko pekara na dnevnom nivou ima tri radnika sa ukupno 20 radnih sati za mešenje i pečenje peciva i 20kg brašna, koliko komada hleba i kifli treba da se umesi kako bi se ostvarila maksimalna zarada (pod pretpostavkom da će pekara sve prodati)?

Primer

Kompanija Start
Kompanija Start ima 250 zaposlenih radnika. Rukovodstvo kompanije je odlučilo da svojim radnicima obezbedi dodatnu edukaciju. Da bi se radnik obučio programskom jeziku Elixir potrebno je platiti 100 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 150 projekat/sati mesečno, što bi za kompaniju značilo dobit od 5 evra po projekat/satu. Da bi se radnik obučio programskom jeziku Dart potrebno je platiti 105 evra po osobi za kurs, ali bi njegovo produktivno znanje ovog programskog jezika donelo 170 projekat/sati mesečno, koji bi za kompaniju značili dobit od 6 evra po satu. Ukoliko Start ima na raspolaganju 26000 evra za obuku i maksimalan broj 51200 mogućih projekat/sati mesečno, odrediti na koji način kompanija treba da obuči svoje zaposlene kako bi ostvarila maksimalnu dobit?

1.3 Podrška za programiranje ograničenja

Deklarativno opisivanje problema

- Sa aspekta prakse i primena, programiranje ograničenja je softverska tehnologija za deklarativno opisivanje i efikasno rešavanje prvenstveno kombinatornih problema.

- Osnovna ideja u programiranju ograničenja je da korisnik najpre postavi svoj problem na odgovarajući način, tj pomoću ograničenja, a zatim pronađe rešenje korišćenjem nekog opštenamenskog rešavača ograničenja.
- Dakle, od sistema se očekuje da izračuna rešenje, tj da izračuna vrednosti promenljivih koje zadovoljavaju data ograničenja.

Rešavanje problema

- Sa aspekta naučnog istraživanja, programiranje ograničenja je multidisciplinarna oblast u kojoj se kombinuju metode i tehnike iz računarskih nauka, veštačke inteligencije, operacinih istraživanja, baza podataka, teorije grafova i logičkog programiranja.
- Podrška za ograničenja su ili ugrađena u programski jezik ili su data preko neke biblioteke.

Direktna podrška za programiranje ograničenja

- Claire http://www.claire-language.com/
- Curry (zasnovan na Haskell-u) http://www-ps.informatik.uni-kiel.de/currywiki/
- Kaleidoscope https://constraints.cs.washington.edu/cip/kaleidoscope-asi.html
- Oz http://strasheela.sourceforge.net/strasheela/doc/01-Basics.html
- Wolfram language https://www.wolfram.com/language/

Podrška za programiranje ograničenja

- Postoje različite biblioteke za programiranje ograničenja za jezike C, C++, JAVA, Phyton, za .NET platformu, Ruby
- Biblioteke programiranja ograničenja mogu imati različite pristupe za rešavanje problema, ali nije neophodno poznavanje algoritama koje ove biblioteke koriste (npr, mogu se koristiti SAT i SMT rešavači).

Neke biblioteke

- Artelys Lakis (C++, Java, Python) https://www.artelys.com/en/optimization-tools/kalis
- Cassowary (C++, Java, JavaScript, Ruby, Smalltalk, Python) https://constraints.cs.washington.edu/cassowary/
- CHIP V5 (C++, C) http://www.cosytec.com/production_scheduling/chip/chip_technology.htm
- Choco (Java) http://choco-solver.org/
- Cream (Java) http://bach.istc.kobe-u.ac.jp/cream/
- Disolver (C++) http://research.microsoft.com/apps/pubs/default.aspx?id=64335
- Gecode (C++, Python) http://www.gecode.org/
- Google or-tools (Python, Java, C++, .NET) https://github.com/google/or-tools
- JaCoP (Java) http://jacop.osolpro.com/
- JOpt (Java) http://jopt.sourceforge.net/

- Numberjack (Python) http://numberjack.ucc.ie/
- Minion (C++) http://constraintmodelling.org/minion/
- python-constraint (Python) http://labix.org/python-constraint
- Z3 (C++, Java, Python, C, C#) https://github.com/Z3Prover/z3

Koreni programiranja ograničenja

- Programiranje ograničenja je nastalo u okviru logičkog programiranja (Prolog II, Jaffar i Lassez, 1987)
- Programiranje ograničenja je često raspoloživo u okviru sistema za logičko programiranje
- Programiranje ograničenja u logici — Constraint logic programing (CLP)
- Logičko programiranje i programiranje ograničenja imaju puno zajedničkih osobina
- Većina Prolog implementacija uključuje jednu ili više biblioteka za programiranje ograničenja
- B-Prolog, CHIP V5, Ciao, ECLiPSe, SICStus, GNU Prolog, Picat, SWI Prolog

2 Programiranje ograničenja u skript i logičkom programiranju

2.1 Programiranje ograničenja nad konačnim domenom

Konačni domen

- Programiranje ograničenja nad konačnim domenom sastoji se od tri dela
  1. Generisanje promenljivih i njihovih domena
  2. Generisanje ograničenja nad promenljivama
  3. Obeležavanje (labeling) — instanciranje promenljivih

Podrška učenju programiranja ograničenja

- Kriptaritmetike su matematičke igre u kojima se rešavaju jednačine kod kojih su cifre brojeva zamenjene određenim slovima.
- Primer

```
 SEND
+MORE
 ----
MONEY
```

- Kriptoaritmetike su zabavne i pogodne za razumevanje i učenje programiranja ograničenja, ali **NISU** osnovna primena ove vrste programiranja.

Kriptoaritmetika

```
GREEN + ORANGE = COLORS
MANET + MATISSE + MIRO + MONET + RENOIR = ARTISTS
COMPLEX + LAPLACE = CALCULUS
THIS + IS + VERY = EASY
CROSS + ROADS = DANGER
FATHER + MOTHER = PARENT
WE + WANT + NO + NEW + ATOMIC = WEAPON
EARTH + AIR + FIRE + WATER = NATURE
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS
SEE + YOU = SOON
NO + GUN + NO = HUNT
WHEN + IN + ROME + BE + A = ROMAN
DONT + STOP + THE = DANCE
HERE + THEY + GO = AGAIN
OSAKA + HAIKU + SUSHI = JAPAN
MACHU + PICCHU = INDIAN
SHE + KNOWS + HOW + IT = WORKS
COPY + PASTE + SAVE = TOOLS
```

Kriptoaritmetika

```
THREE + THREE + ONE = SEVEN
NINE + LESS + TWO = SEVEN
ONE + THREE + FOUR = EIGHT
THREE + THREE + TWO + TWO + ONE = ELEVEN
SIX + SIX + SIX = NINE + NINE
SEVEN + SEVEN + SIX = TWENTY
ONE + ONE + ONE + THREE + THREE + ELEVEN = TWENTY
EIGHT + EIGHT + TWO + ONE + ONE = TWENTY
ELEVEN + NINE + FIVE + FIVE = THIRTY
NINE + SEVEN + SEVEN + SEVEN = THIRTY
TEN + SEVEN + SEVEN + SEVEN + FOUR + FOUR + ONE = FORTY
TEN + TEN + NINE + EIGHT + THREE = FORTY
FOURTEEN + TEN + TEN + SEVEN = FORTYONE
NINETEEN + THIRTEEN + THREE + TWO + TWO + ONE + ONE + ONE = FORTYTWO
FORTY + TEN + TEN = SIXTY
SIXTEEN + TWENTY + TWENTY + TEN + TWO + TWO = SEVENTY
SIXTEEN + TWELVE + TWELVE + TWELVE + NINE + NINE = SEVENTY
TWENTY + TWENTY + THIRTY = SEVENTY
FIFTY + EIGHT + EIGHT + TEN + TWO + TWO = EIGHTY
FIVE + FIVE + TEN + TEN + TEN + TEN + THIRTY = EIGHTY
SIXTY + EIGHT + THREE + NINE + TEN = NINETY
ONE + NINE + TWENTY + THIRTY + THIRTY = NINETY
```

2.2 Modul *python-constraint*

Programiranje ograničenja u Python-u

- Modul *python-constraint* podržava programiranje ograničenja na konačnom domenu.
- Programiranje ograničenja nad konačnim domenom sastoji se od tri dela
  1. Generisanje promenljivih i njihovih domena
  2. Generisanje ograničenja nad promenljivama
  3. Obeležavanje (labeling) — instanciranje promenljivih
- U okviru Pythona, rešenja se daju za sve promenljive, tako da je instanciranje podrazumevano

Programiranje ograničenja u Python-u

```
import constraint

problem = constraint.Problem()

problem.addVariable("a", [1,2,3])  # generisanje promenljivih i njihovih domena
problem.addVariable("b", [4,5,6])  # generisanje promenljivih i njihovih domena
```

```
                                # nemamo nikakvih ogranicenja
resenja = problem.getSolutions()   # trazimo resenja

print resenja
```

```
[{'a': 3, 'b': 6}, {'a': 3, 'b': 5}, {'a': 3, 'b': 4},
 {'a': 2, 'b': 6}, {'a': 2, 'b': 5}, {'a': 2, 'b': 4},
 {'a': 1, 'b': 6}, {'a': 1, 'b': 5}, {'a': 1, 'b': 4}]
```

Programiranje ograničenja u Python-u

```
import constraint

problem = constraint.Problem()

problem.addVariable("a", [1,2,3])  # generisanje promenljivih i njihovih domena
problem.addVariable("b", [4,5,6])  # generisanje promenljivih i njihovih domena

def o(a,b):                        # definisanje ogranicenja
    if(2*a>b): return True

problem.addConstraint(o,"ab")      # dodavanje ogranicenja nad promenljivama a i b
resenja = problem.getSolutions()   # trazenje resenja

print resenja
```

```
[{'a': 3, 'b': 5}, {'a': 3, 'b': 4}]
```

Opšta ograničenja

- `AllDifferentConstraint()` - različite vrednosti svih promenljivih
- `AllEqualConstraint()` - iste vrednosti svih promenljivih
- `MaxSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) ne prelazi s
- `MinSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) nije manja od s
- `ExactSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) je s
- `InSetConstraint(skup)` - vrednosti promenljivih se nalaze u skupu skup
- `NotInSetConstraint(skup)` - vrednosti promenljivih se ne nalaze u skupu skup
- `SomeInSetConstraint(skup)` - vrednosti nekih promenljivih se nalaze u skupu skup
- `SomeNotInSetConstraint(skup)` - vrednosti nekih promenljivih se ne nalaze u skupu skup

Primeri

```
  SEND
+MORE
 ----
MONEY
```

SEND+MORE = MONEY

```
import constraint

problem = constraint.Problem()
# Definisemo promenljive i njihove vrednosti
problem.addVariables('SM',range(1,10))
problem.addVariables('ENDORY',range(10))

# Definisemo ogranicenje za cifre
def o(s,e,n,d,m,o,r,y):
    if(s*1000 + e*100 + n*10 + d + m*1000 + o*100 + r*10 + e)
       == (10000*m + 1000*o + 100*n + 10*e + y):
        return True

# Dodajemo ogranicenja za cifre na svim pozicijama
problem.addConstraint(o,"SENDMORY")

# Dodajemo ogranicenje da su sve cifre razlicite
problem.addConstraint(constraint.AllDifferentConstraint())
```

SEND+MORE = MONEY

```
resenja = problem.getSolutions()

for r in resenja:
    print "  "+str(r['S'])+str(r['E'])+str(r['N'])+str(r['D'])
    print " +"+str(r['M'])+str(r['O'])+str(r['R'])+str(r['E'])
    print "="+str(r['M'])+str(r['O'])+str(r['N'])+str(r['E'])+str(r['Y'])


python sendmoremoney.py
  9567
 +1085
=10652
```

2.3 Programiranje ograničenja u *B-Prologu*

B-Prolog

- B-Prolog http://www.picat-lang.org/bprolog/
- B-Prolog je nastao 1994.
- B-Prolog je komercijalni proizvod, ali se može koristiti za učenje i istraživanja besplatno
- B-Prolog implementira ISO standard Prologa ali i razna proširenja
- B-Prolog ima bogat sistem za programiranje ograničenja

Programiranje ograničenja u *B-Prologu*

- B-Prolog podržava programiranje ograničenja na različitim domenima, npr na konačnom i iskaznom domenu.
- Programiranje ograničenja nad konačnim domenom sastoji se od tri dela
  1. Generisanje promenljivih i njihovih domena
  2. Generisanje ograničenja nad promenljivama
  3. Obeležavanje (labeling) — instanciranje promenljivih
- Za programiranje ograničenja uvode se novi predikati, koje ćemo razmotriti kroz primere.

Programiranje ograničenja u B-Prologu

Generisanje promenljivih i njihovih domena

- Definisanje domena: Vars in D, ili Vars :: D
- D se definiše kao Početak..Korak..Kraj
- Na primer 1..5..20 definiše domen 1,6,11,16
- Korak nije obavezan, podrazumeva se da je jedan. Na primer, 1..10 — svi brojevi od 1 do 10

Programiranje ograničenja u *B-Prologu*

- Postoje predikati za opšta ograničenja, a mogu se zadavati numerička ograničenja
- Opšta, npr alldifferent, ili alldistinct
- Numerička počinju sa #, npr #<, #<=...
- Labeling se odnosi na instanciranje promenljivih i na njihovo prikazivanje

Primeri

Promenljive: X i Y, Domeni: X je bilo koji broj od 1 do 10, Y neparni brojevi od 5 do 30. Ograničenje: $X \geq Y$, Instanciranje: po X

```
 ?-  X in 1..10, Y in 5..2..30, X#>=Y, labeling(X).
X = 5
Y = 5 ?;
X = 6
Y = 5 ?;
X = 7
Y = _0490::[5,7] ?;
X = 8
Y = _0490::[5,7] ?;
X = 9
Y = _0490::[5,7,9] ?;
X = 10
Y = _0490::[5,7,9] ?;
no
```

Primeri

Promenljive: X i Y, Domeni: X je bilo koji broj od 1 do 10, Y neparni brojevi od 5 do 30. Ograničenje: $X \geq Y$, Instanciranje: po X i Y

```
 ?-  X in 1..10, Y in 5..2..30, X#>=Y, labeling(X), labeling(Y).
X = 5
Y = 5 ?;
X = 6
Y = 5 ?;
X = 7
Y = 5 ?;
X = 7
Y = 7 ?;
X = 8
Y = 5 ?;
...
```

Primeri

```
 SEND
+MORE
 ----
MONEY
```

Primeri

```
sendmoremoney(Vars) :- Vars = [S,E,N,D,M,O,R,Y], %generisanje promenljivih
    Vars :: 0..9,                                %definisanje domena
    S #\= 0,                                     %ogranicenja
    M #\= 0,
    all_different(Vars),
                1000*S + 100*E + 10*N + D
    +           1000*M + 100*O + 10*R + E
    #= 10000*M + 1000*O + 100*N + 10*E + Y,
    labeling(Vars).                              %instanciranje


| ?- consult(primer1).
| ?- sendmoremoney(Vars).
Vars = [9,5,6,7,1,0,8,2] ?;
no
```

3 Zaključak

Zaključak

- Programiranje ograničenja je korisna i svuda pristuna paradigma: većina programskih jezika ima podršku za programiranje ograničenja kroz različite biblioteke
- Važno je razumeti principe programiranja ograničenjna i modelovanje problema sistemima ograničenja: kada se to razume dalje se može lako prilagoditi sintaksi i biblioteci određenog programskog jezika
- I, pre svega, najbitnije je prepoznati probleme koji pripadaju ovoj paradigmi kako bi se iskoristile prednosti savremenih rešavača, tj. kako bi se brzo i efikasno rešavali problemi ograničenja

4 Literatura

Literatura

- Concepts of programming languages, Robert W. Sebesta
- Programming Language Pragmatics, Michael L. Scott
- http://labix.org/doc/constraint/
- https://pypi.python.org/pypi/python-constraint
- http://www.hakank.org/constraint_programming_blog/