Deklaratív Programozás

Szeredi Péter[1] Kápolnai Richárd[2]

[1] szeredi@cs.bme.hu
BME Számítástudományi és Információelméleti Tanszék

[2] kapolnai@iit.bme.hu
BME Irányítástechnika és Informatika Tanszék

2011 ősz

Az előadók köszönetüket fejezik ki Hanák Péternek, a tárgy alapítójának

A tárgy témája

- Deklaratív programozási nyelvek – gyakorlati megközelítésben
- Két fő irány:
  - funkcionális programozás **Erlang** nyelven
  - logikai programozás **Prolog** nyelven
- Bevezetésként foglalkozunk a C++ egy deklaratív résznyelvével, a Cékla nyelvvel – C(É) deKLAratív része
- A két fő nyelv (lásd követelmények): Erlang és Prolog

I. rész

Bevezetés

Tartalom

Honlap, ETS, levelezési lista

- Honlap: `http://dp.iit.bme.hu`
  a jelen félév honlapja: `http://dp.iit.bme.hu/dp-current`
- ETS, az Elektronikus TanárSegéd
  `http://dp.iit.bme.hu/ets`
- Levelezési lista:
  `http://www.iit.bme.hu/mailman/listinfo/dp-l`
- A listára automatikusan felvesszük a tárgy hallgatóit az ETS-beli címükkel. Címet módosítani csak az ETS-ben lehet.
- A listára levelet küldeni a `dp-l@iit.bme.hu` címre lehet.
- Csak a feliratkozási címről küldött levelek jutnak el moderátori jóváhagyás nélkül a listatagokhoz.

Prolog-jegyzet

- Szeredi Péter, Benkő Tamás: Deklaratív programozás. Bevezetés a logikai programozásba. Budapest, 2005
  - Elektronikus változata letölthető a honlapról (ps, pdf)
  - Nyomtatott változata kifogyott
  - Kellő számú további igény esetén megszervezzük az újranyomtatást
- A SICStus Prolog kézikönyve (angol):
  `http://www.sics.se/isl/sicstuswww/site/documentation.html`

Magyar nyelvű Prolog szakirodalom

- Farkas Zsuzsa, Futó Iván, Langer Tamás, Szeredi Péter:
  Az MProlog programozási nyelv.
  Műszaki Könyvkiadó, 1989
  jó bevezetés, sajnos az MProlog beépített eljárásai nem szabványosak.
- Márkusz Zsuzsa: Prologban programozni könnyű.
  Novotrade, 1988
  mint fent
- Futó Iván (szerk.): Mesterséges intelligencia. (9.2 fejezet, Szeredi Péter)
  Aula Kiadó, 1999
  csak egy rövid fejezet a Prologról
- Peter Flach: Logikai Programozás. Az intelligens következtetés példákon keresztül.
  Panem — John Wiley & Sons, 2001
  jó áttekintés, inkább elméleti érdeklődésű olvasók számára

Angol nyelvű Prolog szakirodalom

- Logic, Programming and Prolog, 2nd Ed., by Ulf Nilsson and Jan Maluszynski, Previously published by John Wiley & Sons Ltd. (1995) Letölthető a `http://www.ida.liu.se/~ulfni/lpp` címről.
- Prolog Programming for Artificial Intelligence, 3rd Ed., Ivan Bratko, Longman, Paperback - March 2000
- The Art of PROLOG: Advanced Programming Techniques, Leon Sterling, Ehud Shapiro, The MIT Press, Paperback - April 1994
- Programming in PROLOG: Using the ISO Standard, C.S. Mellish, W.F. Clocksin, Springer-Verlag Berlin, Paperback - July 2003

Erlang-szakirodalom (angolul)

- Joe Armstrong: Programming Erlang. Software for a Concurrent World. The Pragmatic Bookshelf, 2007.
  `http://www.pragprog.com/titles/jaerlang/programming-erlang`
- Joe Armstrong, Robert Virding, Claes Wikström, Mike Williams: Concurrent Programming in Erlang. Second Edition. Prentice Hall, 1996. Az első rész szabadon letölthető PDF-ben:
  `http://erlang.org/download/erlang-book-part1.pdf`

További irodalom:

- On-line Erlang documentation
  `http://erlang.org/doc.html`
- On-line help (csak unix/linux rendszeren)
  `man erl` vagy `erl -man <module>`
- Wikibooks on Erlang Programming
  `http://en.wikibooks.org/wiki/Erlang_Programming`
- Francesco Cesarini, Simon Thompson: Erlang Programming. O´Reilly, 2009. `http://oreilly.com/catalog/9780596518189/`

Fordító- és értelmezőprogramok

- SICStus Prolog — 4.2.0 verzió (licensz az ETS-en keresztül kérhető)
- Erlang (szabad szoftver)
- Letöltési információ a honlapon (Linux, Windows)
- Webes Prolog gyakorló felület az ETS-ben (ld. honlap)
- Kézikönyvek HTML-, ill. PDF-változatban
- Más programok: SWI Prolog `http://www.swi-prolog.org/`, Gnu Prolog `http://www.gprolog.org/`
- emacs-szövegszerkesztő Erlang-, ill. Prolog-módban (Linux, Win95/98/NT/XP/Vista/7)
- Eclipse fejlesztői környezet (SPIDER, erlIDE)

Deklaratív programozás: félévközi követelmények

Nagy házi feladat (NHF)

- Programozás mindkét fő nyelven (Prolog, Erlang)
- Mindenkinek önállóan kell kódolnia (programoznia)!
- Hatékony (időlimit!), jól dokumentált („kommentezett”) programok
- A két programhoz közös, 5–10 oldalas fejlesztői dokumentáció (TXT, HTML, PDF, PS; de nem DOC vagy RTF)
- Kiadás legkésőbb a 6. héten, a honlapon, letölthető keretprogrammal
- Beadás a 12. héten; elektronikus úton (ld. honlap)
- A beadáskor és a pontozáskor külön-külön tesztsorozatot használunk (nehézségben hasonlókat, de nem azonosakat)
- Azok a programok, amelyek megoldják a tesztesetek 80%-át *létraversenyen* vesznek részt (hatékonyság, gyorsaság plusz pontokért)

Deklaratív programozás: félévközi követelmények (folyt.)

Nagy házi feladat (folyt.)

- A beadási határidőig többször is beadható, csak az utolsót értékeljük
- Pontozása mindkét fő nyelvből:
  - helyes (azaz jó eredményt időkorláton belül adó) futás esetén a 10 teszteset mindegyikére 0,5-0,5 pont, összesen max. 5 pont
  - a dokumentációra, a kód olvashatóságára, kommentezettségére max. 2,5 pont
  - tehát nyelvenként összesen max. 7,5 pont szerezhető
- A NHF súlya az osztályzatban: 15% (a 100 pontból 15)
- A megajánlott jegy előfeltétele, hogy a hallgató nagy házi feladata mindkét fő nyelvből bejusson a létraversenybe (minimum 80%-os teljesítmény)

Deklaratív programozás: félévközi követelmények (folyt.)

Kis házi feladatok (KHF)

- 3 feladat Prologból, 3 Erlangból, 1 Céklából
- Beadás elektronikus úton (ld. honlap)
- Egy KHF beadása érvényes, ha minden tesztesetre lefut
- Kötelező a KHF-ek legalább 50%-ának érvényes beadása, és legalább egy érvényes KHF beadása Prologból is és Erlangból is.
- Minden feladat jó megoldásáért 1-1 jutalompont jár

Deklaratív programozás: félévközi követelmények (folyt.)

Gyakorlatok

- Kéthetente 2 órás gyakorlatok (páros heteken, beosztás hamarosan)
- Időpontok:
  1. Szeptember 16.
  2. Szeptember 30.
  3. Október 14.
  4. Október 28.
  5. November 11.
  6. December 9.

  (November 25. oktatási szünet)
- Laptop használata megengedett
- Kötelező részvétel a gyakorlatok 70 %-án (pontosabban n gyakorlat esetén legalább $\lfloor 0.7n \rfloor$ gyakorlaton)
- További Prolog gyakorlási lehetőség az ETS rendszerben (gyakorló feladatok, lásd honlap)

Deklaratív programozás: félévközi követelmények (folyt.)

Nagyzárthelyi, pótzárthelyi (NZH, PZH, PPZH)

- A zárthelyi kötelező, semmilyen jegyzet, segédlet nem használható!
- 40%-os szabály (nyelvenként a maximális részpontszám 40%-a kell az eredményességhez)
- NZH: 2011. október 26. 8:00, PZH: 2011. november 23. 8:00
- A PPZH-ra indokolt esetben a pótlási időszakban egyetlen alkalommal adunk lehetőséget
- Az NZH anyaga az addig előadott tananyag
- A PZH, ill. a PPZH anyaga azonos az NZH anyagával
- A zárthelyi súlya az osztályzatban: 15% (a 100 pontból 15)

Az aláírás megszerzésének feltételei (összefoglalás)

- Részvétel a gyakorlatok legalább 70%-án
- Zárthelyi sikeres megírása, azaz mindkét fő nyelvből legalább 40%-os eredmény elérése
- A 7 kis házi közül legalább 3 érvényes beadása úgy, hogy mindkét fő nyelvből legalább egy érvényes kis házi van

Deklaratív programozás: vizsga

- Feltétel: aláírás a jelen félévben vagy korábban
- A vizsga szóbeli, felkészülés írásban
- Prolog, Erlang: több kisebb feladat (programírás, -elemzés) kétszer 35 pontért
- A vizsgán szerezhető max. 70 ponthoz adjuk hozzá a félévközi munkával szerzett pontokat: ZH: max. 15 pont, NHF: max. 15 pont, továbbá a pluszpontokat (KHF, létraverseny)
- A vizsgán semmilyen jegyzet, segédlet nem használható, de lehet segítséget kérni
- Nyelvenként a max. részpontszám 40%-a kell az eredményességhez
- Elővizsga a pótlási héten – minden, a tárgyból vizsgára bocsátható hallgató jelentkezhet
- Megajánlott vizsgajegy
  - Alapfeltételek: aláírás; NHF „megvédése” az elővizsgán
  - Jó (4): a nagy házi feladat mindkét fő nyelvből bejut a létraversenybe
  - Jeles (5): legalább 40%-os eredmény a létraversenyen, mindkét fő nyelvből

Tartalom

Bevezető példa: adott értékű kifejezés előállítása

- A feladat: írjunk Prolog programot a következő feladvány megoldására:
  - Adott számokból a négy alapművelet (+, -, *, /) segítségével építsünk egy megadott értékű kifejezést!
  - A számok nem „tapaszthatók” össze hosszabb számokká
  - Mindegyik adott számot pontosan egyszer kell felhasználni, sorrendjük tetszőleges lehet
  - Nem minden alapműveletet kell felhasználni, egyfajta alapművelet többször is előfordulhat
  - Zárójelek tetszőlegesen használhatók
- Példák a fenti szabályoknak megfelelő, az 1, 3, 4, 6 számokból felépített kifejezésekre: $1 + 6 * (3 + 4)$, $(1 + 3)/4 + 6$
- Viszonylag nehéz megtalálni egy olyan kifejezést, amely az 1, 3, 4, 6 számokból áll, és értéke 24

Aritmetikai kifejezések kezelése Prologban – ellenőrzés

Írjunk egy `kif` nevű egyargumentumú Prolog eljárást!

- `kif(X)` ellenőrzi, hogy `X` egy olyan kifejezés-e, amely számokból a négy alapművelet (+, -, *, /) segítségével épül fel.

```
% kif(K): K számokból, a négy alapművelettel képzett kifejezés.
kif(K) :-   number(K).
kif(X+Y) :- kif(X), kif(Y).          kif(X-Y) :- kif(X), kif(Y).
kif(X*Y) :- kif(X), kif(Y).          kif(X/Y) :- kif(X), kif(Y).
```

- Betöltése: `| ?- compile(file).` vagy `| ?- consult(file).`
- Futtatás nyomkövetés nélkül és nyomkövetéssel (`consult`-ot követően):

```
| ?- kif(alma).
no
| ?- kif(1+2).
yes
| ?-
```

```
| ?- trace, kif(alma).
% The debugger will first creep
        1      1 Call: kif(alma) ?
        2      2 Call: number(alma) ?
        2      2 Fail: number(alma) ?
        1      1 Fail: kif(alma) ?
no
| ?-
```

Aritmetikai kifejezések ellenőrzése – továbbfejlesztett változat

- A `kif` Prolog eljárás segédeljárást használó változata (javasolt formázással):

```
% kif2(K): K számokból, a négy alapművelettel képzett kifejezés.
kif2(Kif) :-
        number(Kif).
kif2(Kif) :-
        alap4(X, Y, Kif),
        kif2(X),
        kif2(Y).
```

- Az `alap4` segédeljárás:

```
% alap4(X, Y, Kif): A Kif kifejezés az X és Y kifejezésekből
%                    a négy alapművelet egyikével áll elő.
alap4(X, Y, X+Y).
alap4(X, Y, X-Y).
alap4(X, Y, X*Y).
alap4(X, Y, X/Y).
```

Aritmetikai kifejezés levéllistájának előállítása

- A `kif_levelek` eljárás ellenőrzi a kifejezést és előállítja levéllistáját

```
% kif_levelek(Kif, L): A számokból alapműveletekkel felépülő Kif
%                       kifejezés leveleiben levő számok listája L.
kif_levelek(Kif, L) :-
        number(Kif), L = [Kif].  % L egyelemű, Kif-ből álló lista
kif_levelek(Kif, L) :-
        alap4(K1, K2, Kif),
        kif_levelek(K1, LX),
        kif_levelek(K2, LY),
        append(LX, LY, L).
```

```
| ?- kif_levelek(2/3-4*(5+6), L).     ⟶ L = [2,3,4,5,6]
```

- Az `append` egy beépített eljárás, fejkommentje és példafutása

```
% append(L1, L2, L3): Az L1 és L2 listák összefűzése az L3 lista.
```

```
| ?- append([1,2], [3,4], L).        ⟶ L = [1,2,3,4]
```

Az append eljárás többirányú használata

- Az append eljárás a fejkommentje által leírt *relációt* valósítja meg, több választ is adhat (új válasz kérése a ; karakterrel)

```
% append(L1, L2, L3): Az L1 és L2 listák összefűzése az L3 lista.

| ?- append(L, [3], [1,2,3]).    % [1,2,3] utolsó eleme-e 3,
L = [1,2] ? ;                    % és milyen L lista van előtte?
no                               % nincs TÖBB válasz
| ?- append([1,2], L, [1,2,3]). % [1,2,3,4] prefixuma-e [1,2]?
L = [3] ? ;  no
| ?- append(L1, L2, [1,2,3]).   % [1,2,3] hogyan bontható két részre?
L1 = [], L2 = [1,2,3] ? ;
L1 = [1], L2 = [2,3] ? ;
L1 = [1,2], L2 = [3] ? ;
L1 = [1,2,3], L2 = [] ? ; no
| ?- append(L, [2], L2).
L = [], L2 = [2] ? ;
L = [_A], L2 = [_A,2] ? ;
L = [_A,_B], L2 = [_A,_B,2] ? ; % végtelen sok válasz, problémás ...
...
```

Adott levéllistájú aritmetikai kifejezések előállítása

- A `kif_levelek` eljárás sajnos nem használható „visszafelé”, végtelen ciklusba esik, lásd pl. `| ?- kif_levelek(Kif, [1]).`
- Ez javítható a hívások átrendezésével és új feltételek beszúrásával

```
% kif_levelek(+Kif, -L):
% Kif levéllistája L.
kif_levelek(Kif, L) :-
        number(Kif),
        L = [Kif].
kif_levelek(Kif, L) :-
        alap4(K1, K2, Kif),

        kif_levelek(K1, L1),
        kif_levelek(K2, L2),
        append(L1, L2, L).
```

```
% levelek_kif(+L, -Kif):
% Kif levéllistája L.
levelek_kif(L, Kif) :-
        L = [Kif],
        number(Kif).
levelek_kif(L, Kif) :-
        append(L1, L2, L),
        L1 \= [], L2 \= [],
        % L1, L2 nem-üres listák
        levelek_kif(L1, K1),
        levelek_kif(L2, K2),
        alap4(K1, K2, Kif).
```

```
| ?- levelek_kif([1,3,4], K).
K = 1+(3+4) ? ; K = 1-(3+4) ? ; K = 1*(3+4) ? ; K = 1/(3+4) ? ;
K = 1+(3-4) ? ; K = 1-(3-4) ? ; K = 1*(3-4) ? ; K = 1/(3-4) ? ; ...
```

Adott értékű kifejezés előállítása

- Bevezető példánk megoldásához szükséges további nyelvi elemek
  - A `lists` könyvtárban található `permutation` eljárás:

    ```
    % permutation(L, PL): PL az L lista permutációja.
    ```

  - Az `=:=` (`=\=`) beépített aritmetikai eljárás mindkét argumentumában aritmetikai kifejezést vár, azokat kiértékeli, és csakkor sikerül, ha az értékek aritmetikailag megegyeznek (különböznek), pl.

    ```
    | ?- 4+2 =\= 3*2. ⟶ no          | ?-  2.0 =:= 2. ⟶ yes
    | ?- 8/3 =:= 2.666666666666666. ⟶ no
    ```

- A példa „generál és ellenőriz” (generate-and-test) stílusú megoldása:

```
% levelek_ertek_kif(L, Ertek, Kif): Kif az L listabeli számokból
% a négy alapművelet segítségével felépített olyan kifejezés,
% amelynek értéke Ertek.
levelek_ertek_kif(L, Ertek, Kif) :-
        permutation(L, PL), levelek_kif(PL, Kif), Kif =:= Ertek.

| ?- levelek_ertek_kif([1,3,4], 11, Kif).
Kif = 3*4-1 ? ; Kif = 4*3-1 ? ; no
```

Adott értékű kifejezés előállítása – a teljes kód

```
:- use_module(library(lists), [permutation/2]). % importálás

% levelek_ertek_kif(L, Ertek, Kif): Kif az L listabeli számokból
% a négy alapművelettel felépített, Ertek értékű kifejezés.
levelek_ertek_kif(L, Ertek, Kif) :-
        permutation(L, PL), levelek_kif(PL, Kif), Kif =:= Ertek.

% levelek_kif(L, Kif): Az alapműveletekkel felépített Kif levéllistája L.
levelek_kif(L, Kif) :-
        L = [Kif], number(Kif).
levelek_kif(L, Kif) :-
        append(L1, L2, L),
        L1 \= [], L2 \= [], levelek_kif(L1, K1), levelek_kif(L2, K2),
        alap4(K1, K2, Kif).

% alap4(X, Y, Kif): Kif X-ből és Y-ból a 4 alapművelet egyikével áll elő.
alap4(X, Y, X+Y).
alap4(X, Y, X-Y).
alap4(X, Y, X*Y).
alap4(X, Y, X/Y) :- Y =\= 0.      % a 0-val való osztás kiküszöbölése
```