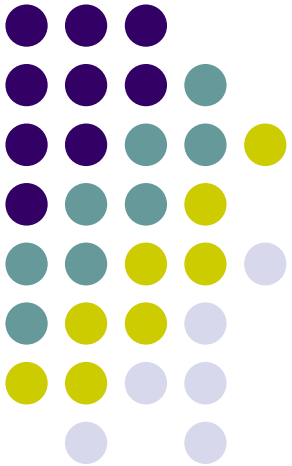
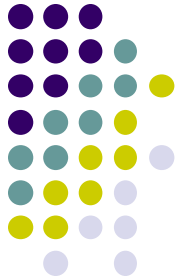
LOOGILINE PROGRAMMEERIMINE (*logic programming*)

Õppejõud: Jüri Vain

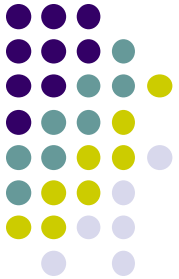


Kursusest üldiselt



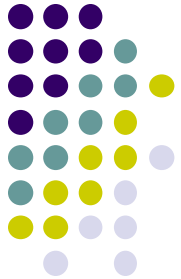
- Kood: ITI0211 6.0 4 2-2-0
  - https://courses.cs.ttu.ee/pages/Loogiline_programmeerimine
  - Moodle
  - https://echo360.org.uk/section/a81a5ca6-06ec-416e-a95f-3067e193e432/home
- Kontakt:
  - Konsultatsioon: N kl 16:00 – 17:00 (eelnevalt teatada)
  - E-post: juri.vain@taltech.ee
  - Telefon: 6204190
  - Ruum: ICT-418

Kursuse korraldus



- Loeng – Jüri Vain
  - T 10:00-11:30 (NRG-226)
- Praktikum - Evelin Halling
  - T 16:00-17:30 (ICT-401)

Hindelise arvestuse nõuded



Praktikumi ülesannete (10 tk) esitamine (tähtajaliselt) – max 40%

+

Test 1 – loengute 1 – 6 materjalile – 25 %

+

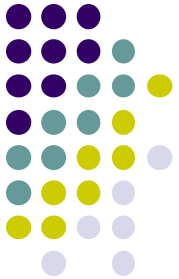
Test 2 – loengute 7– 12 materjalile – 25 %

+

Kodutöö: kabeprogramm (10%) + turniir

NB! Turniiri kohad I-III annavad koondhindele lisapalle vastavalt 3, 2 ja 1 palli.

Õppeaine eesmärgid (I)

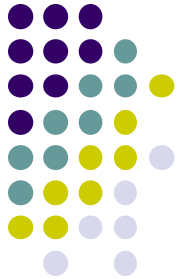


- Teoreetilised alusteadmised

  Kuidas kasutada matemaatilise loogika meetodeid probleemide **deklaratiivseks** kirjeldamiseks ja lahendamiseks?

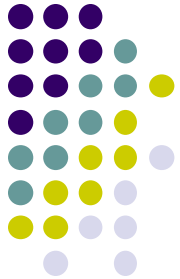
  - Loogilise programmeerimise alused:
    - Lause- ja predikaatarvutus, teadmiste esitamine loogikas
    - Horni lause
    - Resolutsioon
    - Termide unifitseerimine
    - Rekursioon
    - Listid, freimid jt andmestruktuurid

Õppeaine eesmärgid (II)



- Loogilise programmeerimise keel Prolog
  - Süntaks
  - Semantika
  - Mälu optimeerimine
  - Teekide kasutamine (kitsenduste lahendamine, semantiline veeb, stringiteisendused jpm)

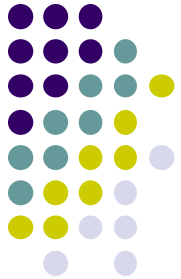
Õppeaine eesmärgid (III)



- Praktilised oskused AI rakenduste programmeerimiseks
  - Objektide ja nende omaduste/seoste esitamine
  - Teadmusbaasi koostamine: sugupuu
  - Rekursiooni programmeerimine: otsing suunatud graafil
  - Listioperatsioonide programmeerimine
  - Otsingu strateegiad: mängude programmeerimine
  - Klasside, hierarhia ja pärismisreeglite programmeerimine
  - Loomuliku keele genereerimine ja parsimine (DCG)
  - Hulgateooria, topoloogia ja algebra põhimõistete programmeerimine (ekvilentsiklassid, tüübid, meetrika)
  - Testimine, mudelkontroll, verifitseerimine

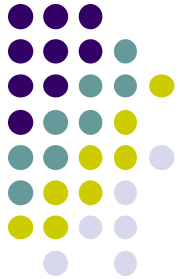
1. Sissejuhatus

1.1. Mis on loogiline programmeerimine?



Programmeerimise paradigma	Fookus:
Imperatiivne	KUIDAS ARVUTADA
OOP	KUIDAS STRUKTUREERIDA PROGRAMMI
Aspekt-orienteeritud	
Struktuurprogrammeerimine	
Loogiline (LP)	MIDA ARVUTADA (kuidas (spetsifitseerida probleemi)
Funktsionaalne (FP)	

Mis eristab LP teistest?



- LP on *deklaratiivne* programmeerimine – kirjeldame probleemi mitte lahendusalgoritmi;
- Programm kirjutatakse loogika keeles (loogika valemitena)
- Programmi täitmine
  - põhineb loogilise tõestamise printsiipidel ja
  - kasutab automaattõestamise protseduure - resolutsioon, unifitseerimine;
- LP keel on Prolog (PROgramming in LOGic) ,
- kuid LP teooria sisaldab enamat kui keel Prolog

Vt. ka LP positsioon programmeerimiskeelte maastikult:

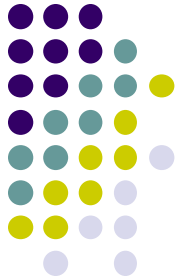
https://en.wikipedia.org/wiki/Programming_paradigm#cite_note-4

1.1. Miks loogiline programmeerimine?

- LP sobib *tehisintellekti rakenduste* programmeerimiseks:
  - loomuliku keele analüüs (DCG grammatikareeglid)
  - ekspertsüsteemid (otsingu- ja järeldusreeglid)
  - kujundituvastus (tuvastusreeglid)
  - kitsendustega planeerimine (logistika, marsruudi otsimine)
  - semantilised võrgud/ *semantic computing*
  - jpm

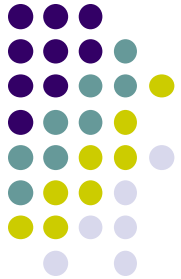
IBM Watson'i dialoogisüsteem ja tuletusmootor on Prologis

LP ei ole „hõbekuul“!



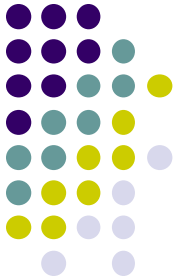
- Ei ole programmeerimiskeelt, mis sobiks ühtviisi hästi kõikide ülesannete lahendamiseks!
- LP-ga on raske saavutada efektiivsust algoritmiselt determineeritud lahendusmeetodite ja ajakriitiliste rakenduste programmeerimisel:
  - maatriksarvutused,
  - võrrandite lahendamine,
  - numbriline optimeerimine
  - graafiliste kasutajaliideste programmeerimine
  - animatsioonide programmeerimine

1.1. Miks peaks õppima loogilist programmeerimist?



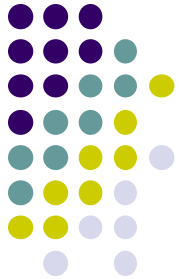
- LP õpetab mõtlema *probleemikeskselt* ja esitama lahenduskäigu vaid *abstraktsel* (kitsenduste) kujul - konkreetse lahenduse leiab Prologi tuletusmootor
- LP aitab vältida programmis protseduurilisi detaile ja sellega seotud programmeerimisvigu.
- LP tagab programmi kompaktsuse ja hea loetavuse – lihtne arendada ja muuta.
- Kui efektiivsus ei ole kriitline, siis LP sobib kiireks prototüüpimiseks.
- LP keeled arenevad koos teiste kaasaegsete programmeerimis-keeltega, sisaldavad nende konstruktsioone ja/või on nendega liidestatavad.

1.1. Mis on loogiline programmeerimine?



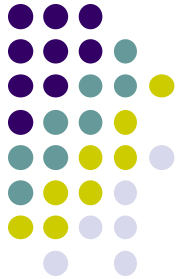
- Universaalne keel omaduste/seoste abstraktseks kirjeldamiseks on loogika.
- → LP on programmeerimine loogika keeles!
- Prolog – *programming in logic*
- LP ≠ Prolog

1.2 LP ajalugu



- Prolog (1972)
  - Alain Colmerauer, Phillipe Roussel;
- Edinburgh Prolog (1980 algus)
  - David Warren;
- 1980 - Jaapani 5. põlvkonna (Prologi) arvuti loomise projekt
- 1980 – 2021 – LP kasutatavuse parandamine/laiendamine teiste paradigmadega:
  - paralleelsus, OO, andmetüübid jm
  - palju Prologi dialekte
  - OWL

1.3 LP meetod (3 sammu)



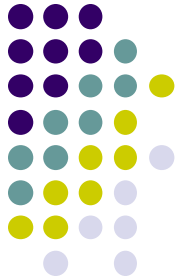
- Piiritleda valdkond (*universe of dicourse*):
  - reaalse maailma modelleeritav situatsioon (*domain, use-cases*)
  - määratleda sellega seotud põhimõisted/olemid
  - defineerida mõisteid iseloomustavad atribuudid ja nende omadused
  - defineerida seosed olemite ja nende atribuutide vahel
- Viia valdkonna ja probleemi kirjeldus Horni lausete kujule:
  - tulemusena tekib hulk Horni lauseid (faktid, reeglid ja päringud)

    Fakt:
    - Sokrates on kreeklane

    Reegel:
    - Kui keegi on kreeklane, siis on ta inimene
- Formuleerida päringud:

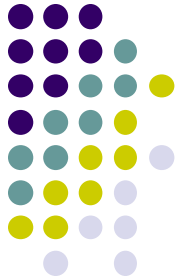
  Päring:
  - Kas Sokrates on inimene?

LP esitusvõimsus



- LP toetab meta-programmeerimist
  - Reeglid programmi dünaamiliseks muutmiseks lahendamise käigus
    - Horni lausete loomine/kustutamine
    - Predikaatmuutujad
    - Termikonstruktorid
    - Kasutaja oma programmeerimiskeele loomise võimalus Prologi baasil

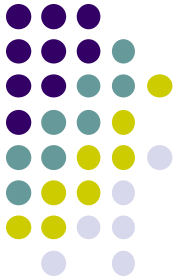
LP “õrnad” kohad



- Teadmiste esitamise ja programmi efektiivsus oleneb reeglite *kujust:*
  - päringu tulemus oleneb otsingureegli alternatiivide ja faktide järjestusest teadmusbaasis
  - tagurdamismehhanismist (*backtracking*) arusaamine nõuab otsingumootori head tundmist
- Praktiline programmeerimine ainult “puhta“ deklaratiivsusega on keeruline.
- Praktilises programmeerimises vaja ka “madala taseme” käske:
  - kasutajaliidese juhtimine,
  - failisüsteemi käsud,
  - stringioperatsioonid

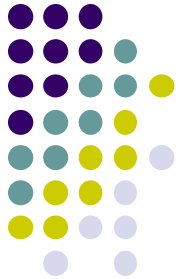
  jms.

Kuhu LP areneb?



- Laiendamine teiste programmikeelte paradigmadega
  - Functional logic programming
    - http://www.informatik.uni-kiel.de/~mh/FLP/
    - keeled Curry and Mercury.
- Efektiivsuse suurendamine
  - Concurrent prolog
    - Curry, ToonTalk, Janus, Alice
- Probleem-orienteerituse laiendamine
  - Constraint Logic Programming
    - http://en.wikipedia.org/wiki/Constraint_logic_programming
  - Semantiline veeb
    - http://hcs.science.uva.nl/projects/SWI-Prolog/articles/mn9c.pdf

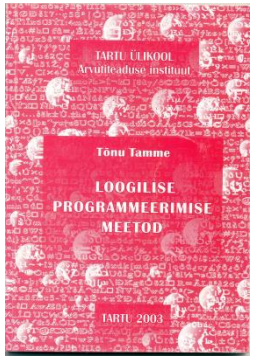
Kursuse sisu (mitte loengute järjestuses)

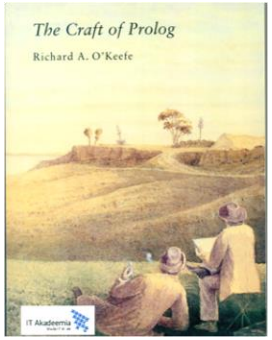


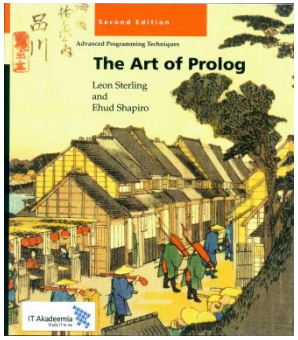
- Alusmõisteid loogikast
  - Loogikasüsteem (faktid ja reeglid)
  - Termid, unifitseerimine ja võrdlemine
  - Tõestusmeetod - resolutsioon
- Prologi andmestruktuurid (listid, semantilised võrgud, freimid)
- Prologi denotatsiooniline ja operatsiooniline semantika
- Prologi süntaks ja operaatorid
- Prologi otsingumootor, otsingu juhtimine
- Arendused: kitsenduste süsteemi kirjeldamine ja lahendamine
- Rakendusnäiteid:
  - teadmusbaasid ja reisiplaani koostamine
  - loomulike keelte analüüs (lausete parsimine)
  - kujundituvastus ja keerdülesannete lahendamine
- Näpunäiteid praktiliseks programmeerimiseks: integreerimine Java ja C++ga.

Kust leida lisamaterjali?

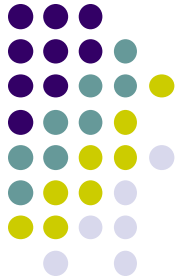
- Õpikud TTÜ raamatukogus:
  - Tõnu Tamme. Loogilise programmeerimise meetod. Tartu Ülikool 2003. (algajatele)
  - R.A.O'Keefe The Craft of Prolog, MIT Press (sissejuhatav)
  - L. Sterling, E. Shapiro, The Art of Prolog. (edasijõudnutele)
  - I. Bratko,"Prolog Programming for Artificial Intelligence", Addison–Wesley Ltd. (rakendusprogrammerijatele)







Lisamaterjal



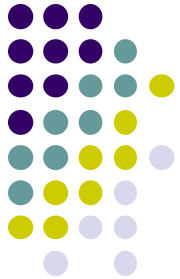
- Ajakirjad:
  - The Journal of Logic and Algebraic Programming

    https://www.journals.elsevier.com/the-journal-of-logic-and-algebraic-programming
  - Theory and Practice of Logic Programming
    - https://www.cambridge.org/core/journals/theory-and-practice-of-logic-programming
- SWI prologi help

Veel linke

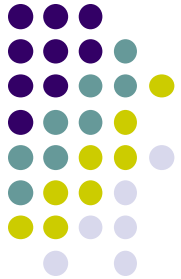
- Pengines: Web Logic Programming Made Easy
  - https://www.cambridge.org/core/journals/theory-and-practice-of-logic-programming/article/abs/pengines-web-logic-programming-made-easy/4C43EAE4DBA200D589F70168D3C3C9B0
- Peter Hancox. Prolog and Logic Programming. School of Computer Science in the University of Birmingham, UK.
  - http://www.cs.bham.ac.uk/~pjh/prolog_course/sem242.html
- The World Wide Web Virtual Library: Logic Programming
  - https://www.cs.ccu.edu.tw/~dan/logic-prog.html
- Guide to Prolog Programming
  - http://kti.mff.cuni.cz/~bartak/prolog/implementations.html
- Object-Oriented Prolog
  - http://www.cetus-links.org/oo_prolog.html
- Jonathan Bowen Logic Programming page
  - http://formalmethods.wikia.com/wiki/Logic_programming

Kuidas hankida oma Prolog?



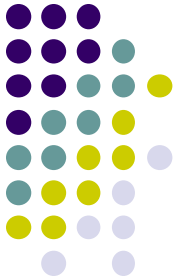
- Unix, Windows, Linux:
  - ALS (Applied Logic Systems, Inc.) Prolog compiler
  - BinProlog, BinNet Corp. See also Jinni (Java INference Engine and Networked Interactor).
  - GNU Prolog compiler - free Prolog compiler with constraint solving over finite domains.
  - IF/Prolog system. IF Computer. Unix, Windows.
  - IT ProLog. IT Masters. (Unix and Windows).
  - LPA WIN-PROLOG, MacProlog32 and Prolog++. Logic Programming Associates Ltd.
  - Quintus Prolog. For Unix and MS Windows.
  - SICStus Prolog (commercial, portable) Unix machines, Windows.
  - **SWI-Prolog**. Unix and MS Windows. Portable.
- PC Prologid:
  - YAP Prolog System (Yet Another Prolog) – kiire Prologi kompilaator,. Akadeemiline litsents vaba.
  - Amzi Prolog + Logic Server. (Commercial). Windows. Allows embedding of Prolog components in C/C++, Visual Basic, Delphi, Access, etc. http://www.amzi.com/download/freedist.htm
  - ADA Prolog (aeglane) ja ESL Prolog (hea, kiire).
  - LPA WIN-Prolog. Windows, Mac ja MS-DOS.
  - Qu-Prolog. Support symbolic computation for mathematical notations and languages such as Z.
  - Visual Prolog from the Prolog Development Center. DOS, Windows 3.1/95/98, NT, Linux.

Mis prologi kasutada praktikumides?

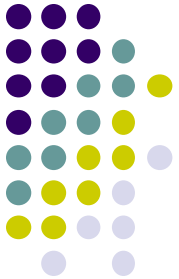


- Praktikumides ainult SWI-Prolog!!
  - http://www.swi-prolog.org/

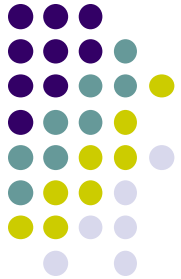
Küsimused?



Loogilise programmeerimise põhimõisted

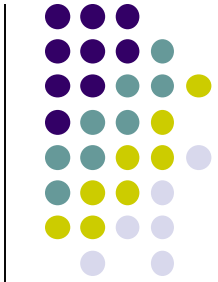


Mis on loogika keel?



- 3 komponenti:
  - Süntaks: reeglid valemite kirjutamiseks
  - Semantika: valemitele tähenduse andmine
  - Tõestusaparaat: reeglid olemasolevatest valemitest (aksioomid ja reeglid) loogiliselt korrektsete järelduste tegemiseks.

Kuidas rakendada loogika reegleid (1)?



- Näide
  - On teada hulk fakte:
    - Juhan on Liia isa
    - Kati on Liia ema
    - Liia on Juku ema
    - Ken on Karini isa
  - Võimalikud päringud sellel faktide hulgal:
    - Kes on Juku vanavanemad?
    - Kes on Karini vanavanemad?
    - …

Kuidas rakendada loogika reegleid (2)?

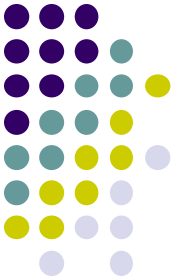
- Esitame faktid Prologis:

```
isa( juhan, liia).
ema( kati, liia).
ema( liia, juku).
isa( ken, karini).
```

kus

- ‘isa’ ja ‘ema’ on *predikaatide nimed*
- ‘juhan’, ‘liia’, ‘kati’, ‘ken’ on *termid*
- `‘ema( liia, juku).’` on predikaatvalem, mis väljendab seost termide vahel.

Programm kui loogika valem



- Esitame seose ‘vanavanem’ Prologis järeldusreeglite kujul:

```
vanavanem(X, Z) :- vanem(X, Y), vanem(Y, Z).
vanem(X, Y) :- isa(X, Y).
vanem(X, Y) :- ema(X, Y).
```

Reegli üldkuju:

järeldus :- eeldus(ed)

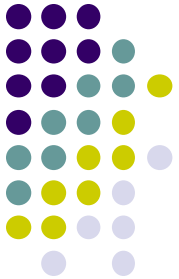
See formaliseerib tingimusliku lause:

„**Mistahes** X,Y,Z korral,

kui X on Y vanem **ja** Y on Z vanem,

siis X on Z vanavanem“.

Programm kui loogika valem



- Konjugeerides faktid ja reeglid, saame loogilise programmi:

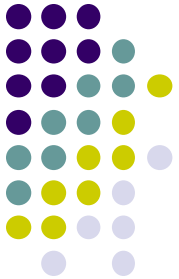
```
isa( juhan, liia).
ema( kati, liia).
ema( liia, juku).
isa( ken, karini).
vanavanem(X, Z) :- vanem(X, Y), vanem(Y, Z).
vanem(X, Y) :- isa(X, Y) ; ema(X, Y).
```

implikatsioon

konjunktsioon

disjunktsioon

Prologi päring



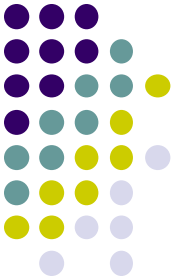
- Kes on Juhani lapselapsed

```
?- vanavanem(juhan, Lapselaps).
```

- Prolog tagastab vastuse

```
Lapselaps = juku
```

Prolog programmi struktuur



- Prologi programm koosneb lausetest kujul

```
A :- B1, B2, …, Bn.
```

  - kus A ja B-d on atomaarvalemid (predikaadid)
  - Sümbol „ :- “ tähistab loogilist implikatsiooni.
  - Kui lause tõesus ei sõltu eeldustest, on tegemist faktiga, mis kirjutatakse ilma tingimusosata kujul

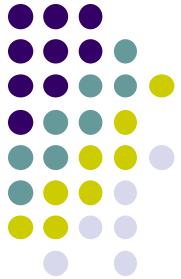
```
A.
```

Näited:

`ilus_ilm.` on 0-kohaline fakt, sest tal puuduvad argumendid.

`ilus_ilm(päike, soe).` on 2 argumendiga fakt

Kokkuvõte



- Loogiline programm koosneb implikatiivsetest lausetest ehk Horni lausetest, kus
  - päringu lauses puudub implikatsiooni järeldusosa
  - faktides puudub implikatsiooni tingimusosa
  - reeglites esinevad mõlemad – tingimus- ja järeldusosa
  - Ühe reegli erinevaid alternatiive tuleb mõista kui loogilise „või“-ga seotud Horni lauseid.

- Programmi täitmine on olemasolevatest Horni lausetest uute lausete tuletamine.