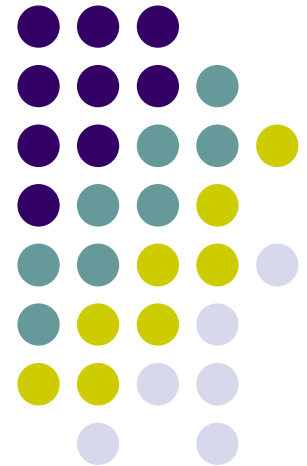
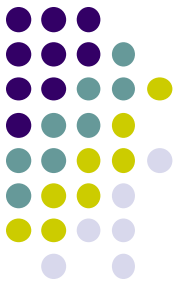


LOOGILINE PROGRAMMEERIMINE *(logic programming)*



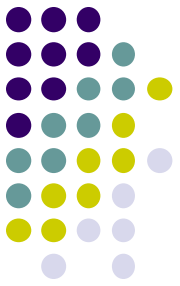
J.Vain





Kursusest üldiselt

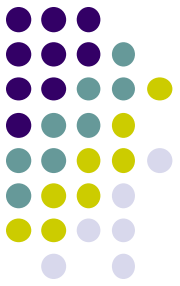
- Kood: ITI0211 6.0 4 2-2-0 H S
 - https://courses.cs.ttu.ee/pages/Loogiline_programmeerimine
 - Kontakt:
 - Õppejõud: prof. Jüri Vain
 - Konsultatsioon: N kl 16:00 – 17:00 (eelnevalt teatada)
 - E-post: juri.vain@ttu.ee
 - Telefon: 6204190
 - Ruum: ICT-419



Kursuse korraldus

- Loeng – prof. Jüri Vain
 - T 10:00-11:30 (ICT-315)
- Praktikum - Evelin Halling, Sven Nõmm
 - **IAIB51, IAIB52: E 16:00-17:30 (ICT-403)**
 - **IAIB53, IAPM31: E: 17:45-19:15 (ICT-403)**

Hindelise arvestuse nõuded



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

+

Test 1 – loengute 1 – 6 materjalile – 25 %

+

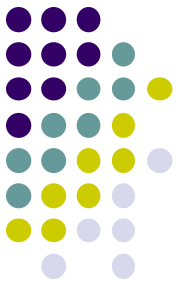
Test 2 – loengute 7– 12 materjalile – 25 %

+

Kodutöö: kabeprogramm (10%) + turniir

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

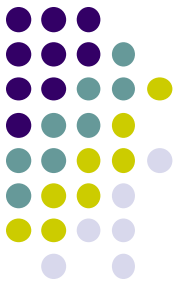
Õppeaine eesmärgid



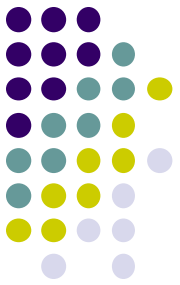
- Teadmiste formaliseeritud esitamine *Horni lausetega*
- Järelduste tuletamine teadmusbaasist kasutades *resolutsiooni* meetodit ja *unifitseerimisreegleid*
- Põhilised otsingumeetodid ja nende programmeerimine keeles *Prolog*
- Tehisintellekti ülesannete lahendamine:
 - kitsendustega tegevuste/logistika/... planeerimine,
 - keerdülesannete (male, kabe, bridž,...) lahendamine
 - inimene-masin dialoogi süsteemid jne
- *Praktiline programmeerimisoskus*

1. Sissejuhatus

1.1. Mis on loogiline programmeerimine?



<u>Programmeerimise paradigma</u>	<u>Fookus:</u>
<ul style="list-style-type: none">● Imperatiivne	KUIDAS ARVUTADA
<ul style="list-style-type: none">● OOP● Aspekt-orienteeritud	KUIDAS STRUKTUREERIDA PROGRAMMI
<ul style="list-style-type: none">● Loogiline (LP)● Funktsionaalne (FP)	MIDA ARVUTADA (kuidas (spetsifitseerida probleemi)



LP aspektid

- LP on *deklaratiivne* programmeerimisstiil;
- Programm kirjutatakse loogika keeles
- Programmi täitmine põhineb
 - loogika printsiipidel ja
 - kasutab automaattõestamise protseduure - resolutsioon, unifikatsioon;
- LP keel on Prolog (PROgramming in LOGic) ,
kuid LP \neq Prolog;



Markup languages
(only Datastructures)

More declarative Paradigms

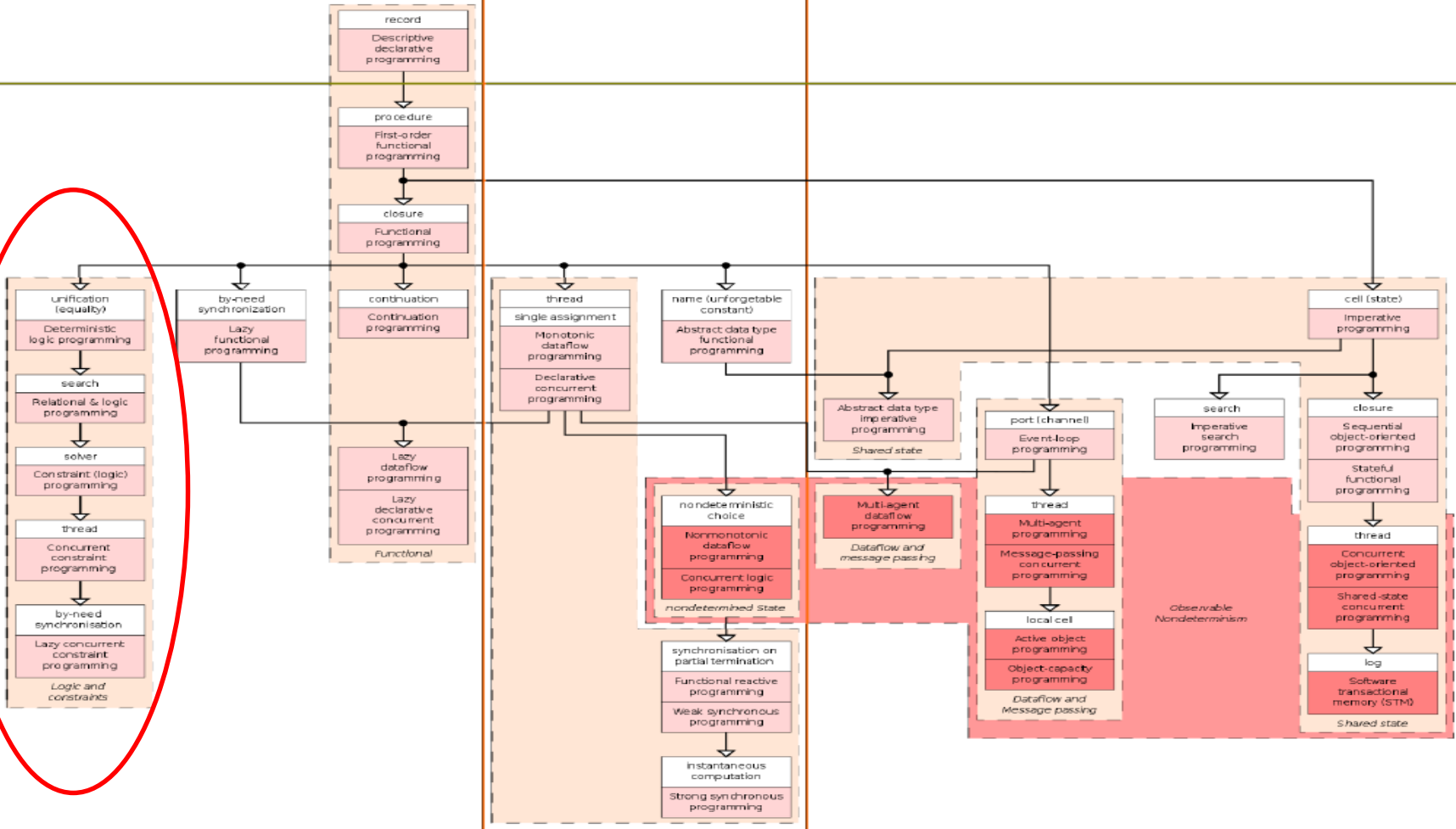
More Imperative Paradigms

Unnamed state
(sequential or concurrent)

Undeterministic state

Named state

Turing complete Languages



1.1. Miks loogiline programmeerimine? (2)



- LP sobib *tehisintellekti rakenduste* programmeerimiseks:
 - loomuliku keele analüüs (DCG grammatikareeglid)
 - ekspertsüsteemid (otsingu- ja järeldusreeglid)
 - kujundituvastus (tuvastusreeglid)
 - kitsendustega planeerimine (logistika, marsruudi otsimine)
 - rekursiivsete funktsioonide püsipunkti arvutus
 - jne

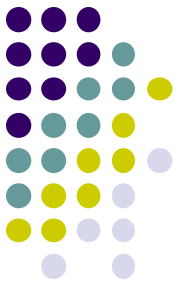




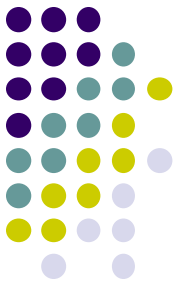
Milleks ei sobi loogiline programmeerimine?

- Ei eksisteeri programmeerimiskeelt, mis sobiks ühtviisi hästi kõikideks rakendusteks!
- LP ei sobi
 - kiireteks numbrilisteks arvutusteks n. maatriksarvutused, võrrandite lahendamine
 - OOP, kuigi OO on toetatud mõnes prologis
 - graafiliste kasutajaliideste programmeerimiseks
 - masingraafikaks ja animatsioonideks

1.1. Miks loogiline programmeerimine? (3)

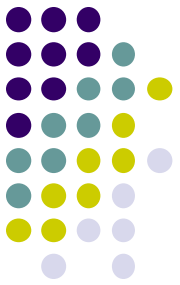


- Õpetab mõtlema *probleemikeskselt* ja esitama lahendusi *abstraktsel* kujul, konkreetse lahenduse leiab Prologi tuletusmootor
 - Programmi põhifunktsioonid:
 - reaalse maailma probleemide *abstraktne esitamine*,
 - abstraktsioonide teisendamine ja sidumine omavahel
 - abstraktsioonide põhjal arvutamine/otsuste tegemine
 - *Programeerimiskeel* peab võimaldama
 - kirjeldada ja analüüsida abstraktsioone nii inimesele, kui *arvutile* sobival kujul
 - *Deklaratiivsed* programmeerimiskeeled sobivad
 - abstraktsete objektide ja nende seoste kirjeldamiseks
 - *väldivad protseduurseid detaile*



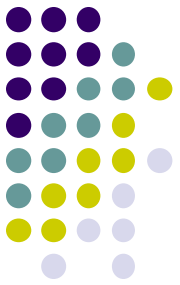
1.1. Mis on loogiline programmeerimine? (4)

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



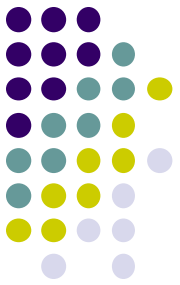
1.2 LP ajalugu

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



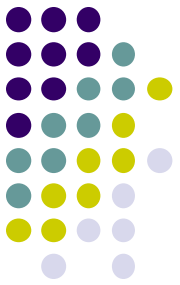
1.3 LP meetod

- Piiritleda valdkond:
 - reaalse maailma modelleeritav situatsioon (*domain, use-cases*)
 - määratleda sellega seotud põhimõisted/olemid
 - defineerida mõisteid iseloomustavad atribuudid ja nende omadused
 - defineerida seosed atribuutide vahel
 - Formaliseerida valdkonna objektid ja seosed LP keeles
 - tulemusena tekib faktide ja reeglite hulkNäide:
 - raamatukogu asub 2. korrusel, <-- fakt
 - Sokrates on kreeklane <-- fakt
 - kui keegi on kreeklane, siis on ta inimene <-- järeldusreegel
 - Formuleerida päringud teadmiste struktuuril
- Näide:
- Kas Sokrates on inimene?



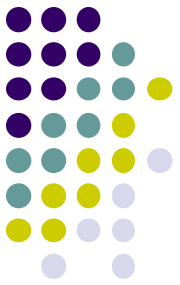
LP “õrnad” kohad

- Teadmiste esitamine on *otsingureeglite tundlik*
 - päringu tulemus oleneb otsingureeglist ja faktide järjestusest teadmusbaasis
 - tagurdamis mehhanismist (*backtracking*) arusaamine nõuab otsingumootori tundmist
- Keeruline saavutada “puhast deklaratiivsust”
- Efektiivsuse saavutamiseks peab tundma Prologi tuletusmootorit
- Praktilises programmeerimises vaja ka nn “madala taseme” käske:
 - kasutajaliidese juhtimine,
 - failisüsteemi käsud,
 - stringioperatsioonidjms.



LP edasiarendused

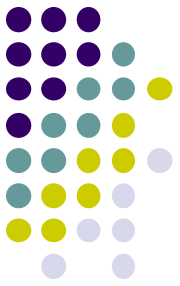
- Laiendamine teiste programmikeelte paradigmadega
 - Functional logic programming
 - <http://www.informatik.uni-kiel.de/~mh/FLP/>
 - keeled [Curry](#) and [Mercury](#).
- Efektiivsuse suurendamine
 - Concurrent prolog
 - [Curry](#), [ToonTalk](#), [Janus](#), [Alice](#)
- Probleem-orienteerituse suurendamine
 - 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

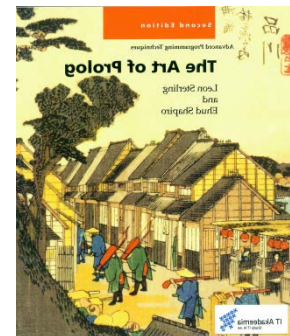
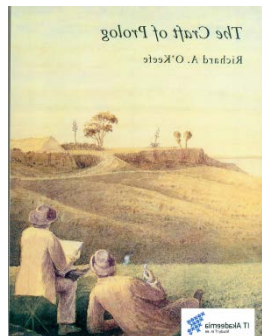
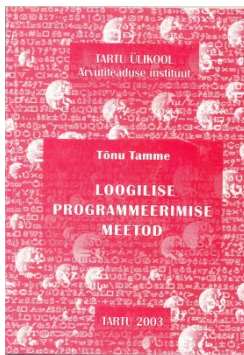
- Alusmõisteid loogikast
 - Loogikasüsteem (faktid ja reeglid)
 - Termide unifitseerimine ja võrdlemine
 - Tõestusmeetod - resolutsioon
- LP andmestruktuurid (listid, semantilised võrgud, freimid)
- LP denotatsiooniline ja operatsiooniline semantika
- Prologi süntaks ja operaatorid
- Prologi otsingumootor, otsingu juhtimine
- Kitsenduste süsteemide kirjeldamine ja lahendamine Prologis
- Rakendusnäiteid:
 - teadmusbasisid ja reisiplaani koostamine
 - loomulike keelte analüüs (lausete parsimine)
 - kujundituvastus ja keerdülesanded
- Näpunäiteid praktiliseks programmeerimiseks: integreerimine Java ja C++ga.

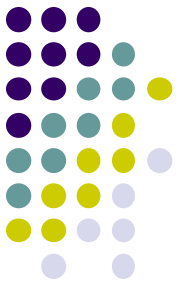
Õppematerjal



- Õpikud TTÜ raamatukogus:

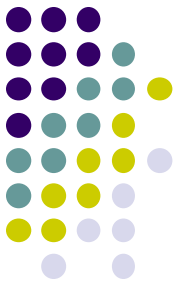
- Tõnu Tamme. Loogilise programmeerimise meetod. Tartu Ülikool 2003. (algajatele)
- R.A.O'Keefe The Craft of Prolog, MIT Press (sissejuhatav)
- L. Sterling, E. Shapiro, The Art of Prolog. (edasijõudnutele)
- I. Bratko, "Prolog Programming for Artificial Intelligence", Addison–Wesley Ltd. (rakendusprogrammerijatele)





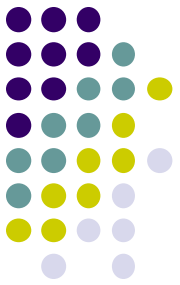
Lisamaterjal

- Ajakirjad:
 - The Journal of Logic and Algebraic Programming
 - (<http://www.informatik.uni-trier.de/~ley/db/journals/jlp/jlap.html>)
 - Theory and Practice of Logic Programming
 - (<http://www.cwi.nl/projects/alp/TPLP/>)
- SWI prologi help



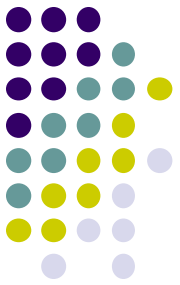
Veel kasulikke linke

- 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
- [Ischislenie Vyskazyvaniy I Logicheskoe Programmirovanie](#)
 - 2012/5/2 Vladimir Kulakov **Vene keeles!**
- The World Wide Web Virtual Library: Logic Programming
- 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?

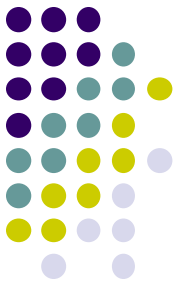
- Unix, Windows, Linux:
 - ALS (Applied Logic Systems, Inc.) Prolog compiler
 - BinProlog, BinNet Corp. See also Jinni (Java INference Engine and Networked Interactor).
 - GNU Prolog compiler - free Prolog compiler with constraint solving over finite domains.
 - IF/Prolog system. IF Computer. Unix, Windows 95/98/NT.
 - IT ProLog. IT Masters. (Unix and Windows NT).
 - 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, Linux/Solaris/Windows NT, 95, 98. Akadeemiline litsents vaba.
 - Amzi Prolog + Logic Server. (Commercial). Windows 3.x, 95, WFW, NT 3.5x, DOS, Extended-DOS. 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 3.1, Macintosh 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.

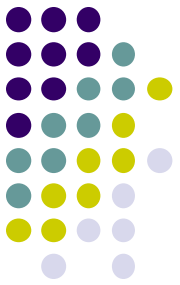


Mis prologi kasutada praktikumides?

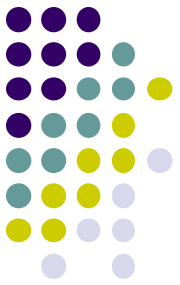
- Praktikumides ainult SWI prolog!!
 - <http://www.swi-prolog.org/>

Küsimused?





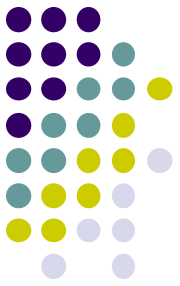
- Sissejuhatus LP põhimõistetes



Mis on loogika keel?

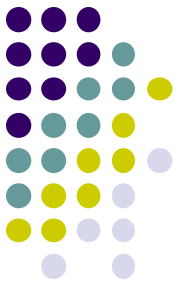
- Süntaks: reeglid valemite kirjutamiseks
- Semantika: valemitele tähenduse andmine
- Tuletusreeglid: reeglid olemasolevatest valemitest loogiliselt korrektsete järelduste tegemiseks

Kuidas rakendada loogika reegleid (1)?

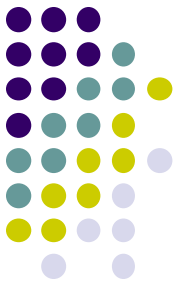


- Näite
- Olgu antud hulk fakte
 - Juhan on Liia isa
 - Kati on Liia ema
 - Liia on Juku ema
 - Ken on Karini isa
- Kes on Juku vanavanemad?
- Kes on Karini vanavanemad?

Kuidas rakendada loogika reegleid (2)?



- Faktid Prologis
isa(juhan, liia).
ema(kati, liia).
ema(liia, juku).
isa(ken, karini
- 'isa' ja 'ema' on *predikaadid*
- 'ema(liia, juku).' - atomaarne valem, mis väljendab tõest fakti.



Programm kui loogika valem

- Esitame seose 'vanavanem'

$\text{vanavanem}(X, Z) :- \text{vanem}(X, Y), \text{vanem}(Y, Z).$

$\text{vanem}(X, Y) :- \text{isa}(X, Y).$

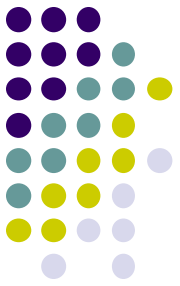
$\text{vanem}(X, Y) :- \text{ema}(X, Y).$

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

- Pannes kokku faktid ja reeglid, saame Prologi programmi:

isa(juhan, liia).

ema(kati, liia).

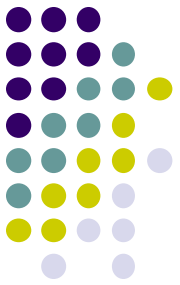
ema(liia, juku).

isa(ken, karini

vanavanem(X, Z) :- vanem(X, Y), vanem(Y, Z).

vanem(X, Y) :- father(X, Y).

vanem(X, Y):- mother(X, Y).

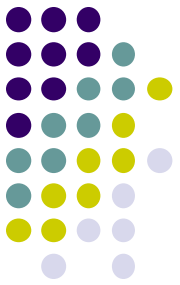


Prologi päring

- Kes on Juhani lapselapsed

?- vanavanem(juhan, Lapselaps).

- Prolog tagastab vastuse
Lapselaps = juku



Prolog programmi struktuur

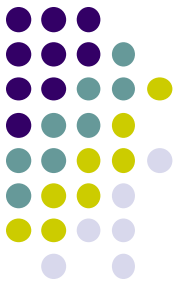
- Üldiselt on Prologi programm hulk lauseid, mis on kujul

$A \text{ :- } B_1, B_2, \dots, B_n.$

- kus A ja B -d on atomaarvalemid
- : - tähistab loogika implikatsiooni \leftarrow
- kui lause paremas pooles puuduvad eeldused, on tegemist faktiga mis kirjutatakse kujul

$A.$

Kokkuvõte



- Loogiline programm koosneb kindlat süntaktilist kuju omavatest lausetest.
- Programmi täitmine on loogiliste järelduste tuletamist aksioomidest ja tõestusreeglitest.



Uurimis- ja lõputöö teemad



- <https://www.spacecenter.ch/igluna/>