Vilniaus universitetas
Matematikos ir informatikos fakultetas
Programų sistemų katedra

Vytautas Čyras

ŽINIŲ VAIZDAVIMAS

Paskaitų konspektas

Vilnius
2007

Turinys

Vytautas Čyras. Žinių vaizdavimas. Elektroninė knyga, 81 p. Prieiga internete http://www.mif.vu.lt/katedros/se/veikla/konspektas-ziniu-vaizdavimas.pdf. 2007.05.21

Pratarmė

Žinių vaizdavimą (ŽV) autorius supranta ir pristato kaip *dirbtinio intelekto* (DI) sudėtinę dalį. Tokiu būdu ŽV yra atskleidžiamas kaip *informatikos* (angl. *computer science*) šaka.

Šios mokymo priemonės turinys formuojasi dėstant vieno semestro kursą „Žinių vaizdavimas“, kuris eina po kurso „Dirbtinis intelektas“.

Mokymo priemonė yra skirta, visų pirma, informatikos specialybės studentams ir doktorantams. Tačiau ji tinka ir pasirinkusiems kitas specialybes, pavyzdžiui, Informacijos sistemų vadybą. Medžiagą siekiama dėstyti taip, kad galėtų suprasti ir skaitytojas, neturintis gilių informatikos žinių.

Autorius siekia pristatyti žinių vaizdavimo „dvasią“. Autorius laikosi nuostatos, kad ŽV dvasia ir apskritai mokslo šakos arba dalykinės veiklos dvasia nėra kokia nors mistinė sąvoka. Čia galima vadovautis analogija su terminu „Konstitucijos dvasia“, kuris buvo pavartotas Lietuvos Konstitucinio Teismo praktikoje, ir yra apibūdinamas, kaip visa tai, kas glūdi už Konstitucijos teksto. Tokiu būdu supaprastintai dalyko dvasią galima suprasti, kaip visa tai, kas slypi už teksto. Tačiau problemos esmę sudaro tai, kaip tą dvasią pavaizduoti. Per mokymo priemonėje pateikiamą medžiagą autorius ir siekia atskleisti ŽV dvasią.

Autorius gina požiūrį, kad DI tarnauja, pirmiausia, žmoniškajam pažinimui, dalykinių sričių supratimui ir asmenybės tobulinimui, o tik paskui intelektualizuotų sistemų kūrimui.

Nilsas Nilsonas [Nilsson 1982] savo knygą apie DI pavadino „Dirbtinio intelekto principai“. Tokiu būdu dvasią galima suprasti ir kaip principų visumą.

Savo požiūryje į DI ir ŽV autorius akcentuoja *pažinimą*. Tai nėra atsitiktinė aplinkybė, o esminis priežastinis ryšys. Žodžių *žinios* ir *pažinimas* bendra šaknis bei kilmė charakterizuoja autoriaus požiūrio ir giluminį pradą, ir siekį.

Žinių vaizdavimo kaip ir dirbtinio intelekto sąvoka yra siejama su žodžiu „intelektualus“. Kalbėdami apie žmogaus kuriamus artefaktus ir sistemas siūlome sakyti „intelektualizuota“, pvz., „intelektualizuota programų sistema“ (toliau – intelektualizuota sistema). Pastarąją autorius supranta kaip tokią programų sistemą, kurioje naudojami DI principai. Į intelektualizuotas sistemas autorius žiūri per programų sistemų inžinerijos ir DI prizmę. Tokiu būdu yra pagrindžiama, kodėl ŽV yra pristatomas per informatiką ir DI.

Autorius vadovaujasi matematiniu metodu, tačiau siekia atsižvelgti į metodus, naudojamus socialiniuose ir humanitariniuose moksluose. Todėl siekiama pateikti fundamentalią reikšmę turinčią medžiagą. ŽV ir DI kelrodžiu galėtų būti *visasupantis intelektas* (angl. *ambient intelligence*, AmI) kaip informacinės visuomenės technologijų ateities vizija. Šia vizija vadovaujasi Europos Sąjungos (ES) institucijos, tame tarpe Informacinės visuomenės generalinis direktoratas. ES 6-oji ir 7-oji bendroji mokslo ir technologijų plėtros programa yra sudaromos vadovaujantis visasupančio intelekto vizija.

Mokymo priemonės turinį sudaro šie dalykai. Procedūrinis ir deklaratyvusis žinių vaizdavimas. Žinių vaizdavimo būdai: (1) loginis – predikatų logika; (2) procedūrinis – produkcijų taisyklės, procedūros; (3) tinklai – semantiniai tinklai, koncepciniai grafai, Petri tinklai; (4) struktūrizuotas pavaizdavimas – freimai ir objektai. Ontologija kaip išreikština konceptualizacija. Tipinės (*generic*) DI užduotys (angl. *tasks*): numatymas, valdymas, diagnozė, nurodymas.

Rengiant tekstą talkino studentai, kurie klausėsi paskaitų ir konspektavo. Indrė Lukauskaitė ir Marek Meškevič buvo pirmieji, užrašę konspektą Microsoft Word programa. Toliau talkino Vera Afanasjeva ir kiti. Šis konspektas iš viso nebūtų sukurtas, jeigu ne Justo Arasimavičiaus kūrybinis darbas konspektuojant paskaitų metu ir iš rankraščio perkeliant į failą.

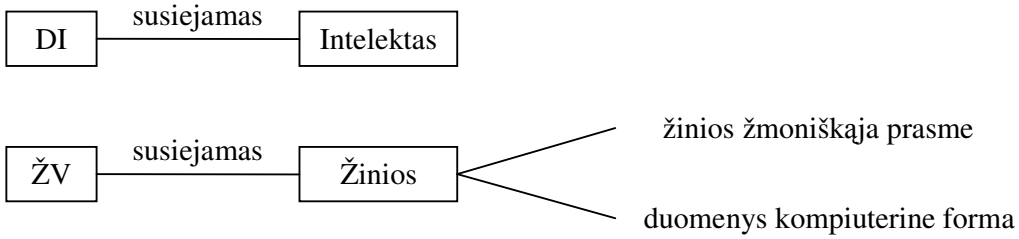
Atsakomybė už netikslumus ir klaidas tenka autoriui.

1. Žinių vaizdavimas kaip dirbtinio intelekto šaka

Šioje mokymo priemonėje autorius laikosi požiūrio, kad *žinių vaizdavimas* (toliau – ŽV) yra *dirbtinio intelekto* (toliau – DI, angl. *artificial intelligence*, AI) šaka. Egzistuoja ir yra priimtini ir kitokie požiūriai. Taip yra laikomasi požiūrio, kad DI yra *informatikos* (angl. *computer science*) šaka. Čia taip pat egzistuoja ir yra priimtini ir kitokie požiūriai bei paradigmos.

Būtų galima sąvokai „žinių vaizdavimas“ turinį suteikti remiantis gramatiniu žodžių „žinios“ ir „vaizdavimas“ aiškinimu. Tačiau šioje mokymo priemonėje to nedarysime; tai galėtų būti atskiras tyrimas. Mokymo priemonėje tikslinga vadovautis tokia ŽV samprata, kuri priimta rinkoje paplitusiuose vadovėliuose, pavyzdžiui, [Brachman, Levesque 2004; Russell, Norvig 2003; Stefik 1995]. Moksliniam tyrimui skirtoje monografijoje būtų galima propaguoti kokią nors specialią ŽV sampratą, pavyzdžiui, ŽV kaip pažinimas. Tačiau tai išeitų už šios mokymo priemonės ribų.

Analogiškai būtų galima ir sąvokai „dirbtinis intelektas“ turinį priskirti remiantis žodžių „dirbtinis“ ir „intelektas“ aiškinimu. Tačiau ir čia vadovaujamės tokia DI samprata, kokia yra atskleidžiama plačiai naudojamuose vadovėliuose, pavyzdžiui, [Nilsson 1982; Nilsson 1998; Russell, Norvig 2003; Luger 2005].



Pav. 1.1 DI yra siejamas su žodžiu intelektas. ŽV – žinios.

Žinių vaizdavimas – žinių vertimas duomenimis. Duomenys jau nebėra mistinė sąvoka ir apie juos jau galima kalbėti.

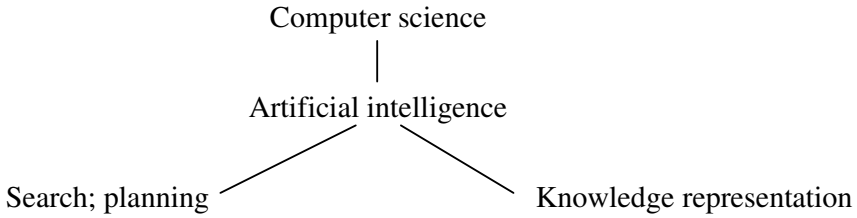
DI sampratai yra būdinga, kad ji kinta. DI riba plečiasi užgriebdama vis naujas sritis. Kas vakar buvo priskiriama žmoniškojo intelekto prerogatyvai, šiandien gali būti priskiriama DI. Šia savybe gali būti paaiškinamas toks posakis: „Mes užsiiminėjome dirbtiniu DI patys to nežinodami, kad tai įeina DI“.

Ir informatika, ir DI sparčiai vystosi. Todėl kiekvienam dešimtmečiui yra būdinga tam tikra samprata. Yra išskiriamos tam tikros *programavimo eros*. Analogiškai yra ir su DI bei ŽV. Todėl yra pateisinama, kad skirtingi autoriai akcentuoja skirtingus požiūrius į DI ir ŽV.

Vadovaujantis [Nilsson 1982] *dirbtinio intelekto sistema* yra suprantama kaip *produkcijų sistema*, kuri, savo ruožtu, yra trejetas: *globali duomenų bazė*, *produkcijų aibė* ir *valdymo sistema*. Dalykinės srities žinios, gali būti klasifikuojamos pagal tai, kur vaizduojamos. Žinios, vaizduojamos globalioje duomenų bazėje, vadinamos *deklaratyviosios žinios,* pvz., faktai. Žinios, pavaizduotos produkcijų aibėje, – tai *procedūrinės žinios*. Žinios, pavaizduotos valdymo strategijoje, – tai *valdymo žinios*.

DI sampratą apsprendžia riba tarp žmoniškojo intelekto (angl. *human intelligence*) ir mašininio intelekto (angl. *machine intelligence*). DI sampratos esmę sudaro tai, kad ši riba vis plečiasi. Kas vakar priklausė žmoniškojo intelekto sričiai, šiandien jau gali būti priskiriama DI. Šios ribos plėtimasis sąlygoja ir *intelektualios sistemos* (angl. *intelligent system*) sampratą.

Žinių vaizdavimo vietą informatikoje atspindi, pavyzdžiui, klasifikacija, pavaizduota Pav. 1.2. Tai labai neformali klasifikacija.



Pav. 1.2 Žinių vaizdavimo vieta informatikoje

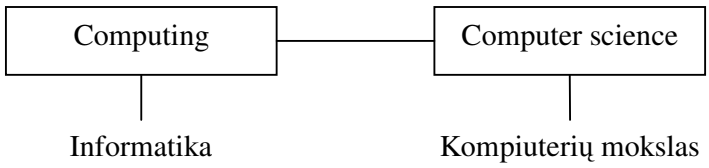
Yra ir kitų daugiau ar mažiau pripažintų informatikos, DI ir ŽV klasifikacijų. Leidykla „Kluwer Academic Publishers“ (pastaraisiais metais įėjusi į Spriger Verlag sudėtį) pateikia tokią DI klasifikaciją (toliau visur pabraukta autoriaus):

- AI languages (PROLOG ir kt.).
- Cognitive science (tyrimo objektas yra tai, kaip pažįsta žmogus).
- Computer vision (rastriniai vaizdai, matricos, pikseliai).
- Expert systems (ekspertinės sistemos).
- Genetic programming and Algorithms (genetinis programavimas ir algoritmai).
- Knowledge rbvepresentation (žinių vaizdavimas).
- Machine learning (nagrinėja algoritmus, kurie patys automatiškai tobulėja).
- Machine translation (tekstų vertimas, nepažodinis, bet su semantika).
- Natural languages (natūrali kalba).
- Robots and automation (pastovūs algoritmai robotams).

Klasifikacija yra vienas iš žinių vaizdavimo metodų. Informatikos, DI iš ŽV klasifikacijų yra ir daugiau. Tai mokslo institucijų nustatyti mokslo šakų klasifikatoriai ir kitokios klasifikacijos. Kitas DI klasifikacijos pavyzdys:

0. Bendrasis DI (filosofiniai pagrindai, kognityvinis modeliavimas) [General AI (Philosophical foundations, Cognitive simulation)]
1. Problemų sprendimas, planavimas, paieška [Problem Solving, Planning, Search]
2. Teoremų įrodymas, loginis išvedimas; euristinis, miglotumų ir tikėtinumų samprotavimas [Theorem Proving, Inference; Heuristic, Uncertainty, Fuzzy, and Probabilistic Reasoning]
3. Žinių vaizdavimas, žiniomis grindžiamos sistemos [Knowledge Representation, Knowledge-Based Systems]
4. Šablonų (signalų, garsų-šnekos, vaizdų) atpažinimas [Pattern Recognition (Signal, Audio-Speach, Optical-Vision)]
5. Natūralios kalbos apdorojimas (supratimas, generavimas) [Natural Language Processing (Understanding, Generation)]
6. Mašininis mokymas, adaptyviosios sistemos, kūrybingumas [Machine learning, Adaptive systems, Creativity]
7. Neuroniniai tinklai; evoliuciniai skaičiavimai (genetinis programavimas ir kt.) [Neural Networks; Evolutionary Computation (Genetic Programming, etc.)]
8. Išskirstytas DI, intelektualizuoti programiniai agentai, daugiaagentinės sistemos [Distributed AI, Intelligent Agents, Multiagent Systems]
9. Robotika [Robotics]
10. Kita [Miscellaneous]

Kaip genetinių algoritmo pavyzdys paprastai pateikiamas keliaujančio pirklio (komovijažieriaus) uždavinio sprendimas ne eksponentinio perrinkimo būdu, o analogijos su chromosomų kryžminimu būdu.



Pav. 1.3 Informatikos ir kompiuterių mokslų sąsaja.

Galime pateikti informatikos klasifikavimo pavyzdį, remiantis ACM Computing Clasification Systems(1998) http://www.acm.org/class/1998/overview.html . Šiame pavyzdyje detaliau aprašysime tik žinių vaizdavimo klasifikavimą.

A. General Literature
B. Hardware
C. Computer Systems Organization
D. Software
E. Data
F. Theory of Computation
G. Mathematics of Computing
H. Information Systems
I. Computing Methodologies
    I.0. General
    I.1. Symbolic and Algebraic manipulation
    I.2. Artificial intelligence
        I.2.0. General
        I.2.1. Applications and Expert Systems
        I.2.2. Automatic Programming
        I.2.3. Deduction and Theorem Proving
        I.2.4. Knowledge Representation Formalisms and Methods
            - Frames and scripts
            - Modal logic
            - Predicate logic
            - Relation systems
            - Representation languages
            - Representations (procedural and rule-based)
            - Semantic networks
            - Temporal logic
        I.2.5. Programming Languages and Software
        I.2.6. Learning
        I.2.7. Natural Language Processing
        I.2.8. Problem Solving, Control Methods, and Search
        I.2.9. Robotics
        I.2.10. Vision and Scene Understanding
        I.2.11. Distributed Artificial Intelligence
    I.3. Computer Graphics
    I.4. Image Processing And Computer Vision
    I.5. Pattern Recognition

I.6. Simulation And Modeling
I.7. Document And Text Processing
J. Computer Applications
K. Computing Milieux

Klasifikavimą kaip abstrahavimąsi ir sąvokų sukūrimą autorius priskiria žmoniškojo intelekto sričiai. Tačiau kaip jau minėjome, plečiantis DI ribai yra užgriebiamos atskiros žmoniškojo intelekto sritys. Klasifikavimas gali būti siejamas su *duomenų gavyba* (angl. *data mining*). Duomenų gavyba tradiciškai yra siejama ir su DI, o ne tik su duomenų bazių valdymo sistemomis ar statistika.

Žinių vaizdavimo būdus (t.y. kryptis, paradigmas) galima skirstyti į dvi grupes:

1. **procedūrinis** (angl. *procedural*) žinių vaizdavimas. Algoritmai ir programos yra šio vaizdavimo pavyzdžiai (pavaizdavimas plačiąja prasme);
2. **deklaratyvus** (angl. *declarative*) žinių vaizdavimas. Labai apibendrintai ir negriežtai apibūdinsime kaip vaizdavimą duomenimis.
    a. **loginis**
        i. Teiginių logika
        ii. Predikatų logika
        iii. Deontinė logika. Pavyzdys: Prieš teiginį A yra vienas iš dviejų deontinių operatorių O arba P. O(A) reiškia „privalo būti“ A, angliškai „it is obligatory that A”. P(A) reikia „gali būti A”, angliškai „it is permitted that A“.
        iv. Atmetimo logika (defeasibility logic)
    b. **procedūrinis** (siaurąja prasme)
        i. (Produkcijų sistemos) A,B,C → D, E
    c. **tinklinis** (angl. *network representation schema*)
        i. Semantiniai tinklai (angl. *semantic networks*) – grafas, kurio viršūnėms ir briaunoms suteikiama tam tikra reikšmė.
        ii. Koncepciniai grafai (angl. *conceptual graphs*) – grafas, kuris atspindi dalykinės srities **sąvokas**.
    d. **struktūrinis** (angl. *structure knowledge representation*)
        i. Freimai (angl. *frames*)
        ii. Objektai (angl. *objects*)

Abstraktus, supaprastintas pasaulio vaizdas, kuriuo norima pavaizduoti tikslą, kuris yra svarbus.

Supaprastintai skirtumą tarp procedūrinio ir deklaratyvaus žinių vaizdavimo galima charakterizuoti ir prisimenant skirtumą tarp programos ir jos specifikacijos. Procedūrinis vaizdavimas (kaip ir programa) atsako į klausimą „kaip“ (tiksliau, kaip sprendžiamas tam tikras uždavinys). Deklaratyvus vaizdavimas (kaip ir programos specifikacija) atsako į klausimą „kas“ (kas yra vaizduojama, koks uždavinys yra sprendžiamas), bet neatsako į klausimą, kaip jį išpręsti.

Žinių vaizdavimo kurse pagrindinis dėmesys yra skiriamas deklaratyvaus žinių vaizdavimo būdams.

2. Duomenys, informacija ir žinios

Informatikoje mes giname tokį požiūrį į žinių vaizdavimą kaip dalykinę veiklą ir procesą:

Žinių vaizdavimas – tai žinių vertimas duomenimis.

[Stefik 1995] knygoje nagrinėjamos žiniomis grindžiamos sistemos. Kartu išsamiai, remiantis anglų kalbos aiškinamuoju žodynu yra aptariama žinių sąvoka.

Sistemų teorijos specialistai [Bellinger, Castro, Mills 2004] apžvelgia sąvokas „duomenys“, „informacija“, „žinios“ ir „išmintis“. Jie cituoja sistemų teorijos specialistą ir organizacijų kaitos profesorių Russell Ackoff [Ackoff 1985], kurio nuomone žmogiškaus protas gali būti klasifikuotas į penkias kategorijas:

1. *duomenys*. Tai simboliai;
2. *informacija*. Tai duomenys, kurie yra apdorojami ir yra naudingi.; atsako į klausimus „kas“, „ką“, „kur“ ir „kada“;
3. *žinios*. Tai duomenų ir informacijos pritaikymas. Atsako į klausimą „kaip“;
4. *supratimas*. Tai „kodėl“ pripažinimas;
5. *išmintis*. Tai įvertintas supratimas.

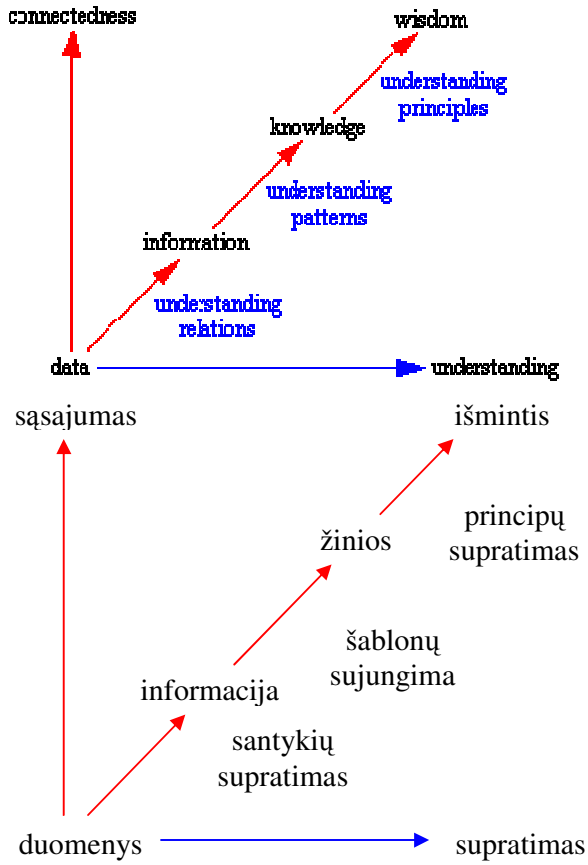
Ackoff nuomone, pirmosios keturios kategorijos siejamos su praeitimi – kas yra arba buvo žinoma. Tik penktoji kategorija – išmintis – yra siejama su ateitimi, nes apima įžvalgumą ir sumanymą.

[Bellinger, Castro, Mills 2004] išvysto aukščiau pateiktus Ackoff apibrėžimus ir pateikia savo modelį (žr.Pav. 2.1):

1. *duomenys* yra neapdoroti. Pavyzdžiui, kompiuterinės skaičiuoklės lentelė (angl. *spreadsheet*);
2. *informacija*. Tai duomenys, kuriems yra suteikta reikšmė. Kitaip sakant, tai interpretuoti duomenys;
3. *žinios*. Tai informacijos visuma, skirta tam tikriems tikslams. Kai asmuo iškala informaciją pats jos nesuprasdamas, tai tokia veika nėra laikoma žinių įsisavinimu. Dauguma taikomųjų programų naudoja kompiuteryje saugomas žinias.
4. *supratimas* pasižymi interpoliavimu ir tikimybiškumu. Supratimas yra kognityvinis ir analitinis procesas. Šiame procese iš vienų žinių yra gaunamos naujos žinios. Skirtumas tarp supratimo ir žinių yra palyginamas skirtumui tarp išmokimo ir įsiminimo. DI sistemos pasižymi supratimu, bet „dirbtinio“, o ne žmogiskojo intelekto prasme. DI sistemos gali sintezuoti naujas žinias iš anksčiau sukauptos informacijos ir žinių;
5. *išmintis* pasižymi extrapoliavimu, o taip pat determinuotumo bei tikimybiškumo nebuvimu. Išmintis iššaukia kitus sąmonės lygmenis, atskiru atveju, žmonėms skirtas normines sistemas, pavyzdžiui, moralės kodeksą. Skirtingai nuo keturių ankstesnių sąvokų išmintis kelia klausimus, į kuriuos nėra lengvų atsakymų arba išvo nėra įmanoma atsakyti. Išmintimi grindžiamas filosofinis tyrimas. Išmintimi yra sprendžiama, kas yra gerai ir blogai, teisinga ir neteisinga. Gene Bellinger laikosi nuomonės, kad kompiuteriai niekada neturės išminties. Išmintis būdinga tik žmogui. Išminčiai būtina siela, kuri glūdi ne tik prote, bet ir širdyje. Siela yra tai, ko kompiuteriai niekada neturės.

Keturias sąvokas – duomenis, informaciją, žinias ir išmintį – jie vaizduoja semantinio grafo viršūnėmis, o penktą – supratimą – jie vaizduoja kaip briaunas (t.y. perėjimus) tarp šių

viršūnių. Kadangi šių perėjimų yra trys, tai jiems priskiriami papildomi atributai, vaizduojantys supratimo lygmenį.



Pav. 2.1 Trys supratimo lygmenys – ryšio supratimas, šablono supratimas ir principų supratimas – kaip perėjimai tarp duomenų, informacijos, žinių ir išminties [Bellinger, Castro, Mills 2004] http://www.systems-thinking.org/dikw/dikw.htm.

Turime trijų lygmenų supratimą:

1. ryšio supratimas kaip perėjimas nuo duomenų prie informacijos;
2. šablono supratimas kaip perėjimas nuo informacijos prie žinių;
3. principų supratimas kaip perėjimas nuo žinių prie išminties.

Duomenys vaizduoja faktą arba teiginį be ryšio su kitais dalykais. Pavyzdžiui:

Lauke lyja.

Duomenys nėra interpretuojami. Pavyzdžiui:

Microsoft Excel ląstelėje parašytas skaičius 17. Šiam skaitmeniui yra nesuteikta jokia prasmė. Jeigu pasakytume, kad tai 17 kilometrų, 17 mylių, 17 000 Lt, 17 lt ir pan. – tai jau būtų informacija.

Informacija yra interpreti duomenys. Reikšminga gali būti tik informacija, o ne duomenys, nes tik suteikus prasmę, galima lyginti reikšmingumą kieno nors atžvilgiu.

Informacija įkūnija tam tikro sąryšio supratimą, pavyzdžiui tarp priežasties ir pasekmės. Pavyzdžiui:

Temperatūra nukrito iki 15 laipsnių ir lauke pradėjo lyti.

Žinios – tai informacijos ir naujų faktų išvedimas. Žinios vaizduoja numatymo šabloną. Šis šablonas sujungia, kas yra duota, su tuo, kas atsitiks. Pavyzdžiui:

Jeigu oro drėgnumas didelis ir oro temperatūra stipriai nukrinta,
tai atmosferoje negali išsilaikyti drėgmė,
ir todėl lauke lyja.

Išmintis įkūnija supratimą fundamentalių principų, kurie yra įkūnyti žiniose. Pavyzdžiui:

Lauke lyja, nes yra toks reiškinys kaip lietus. Lietus kaip gamtos reiškinys apima supratimą apie garavimą, oro sroves, temperatūros pokyčius, garų kondensaciją, lietų ir kt.

Išmintis pasižymi ekstrapoliavimu.

Mūsų nuomone aukščiau pateikti modeliai labiau aktualūs bibliotekininkystės, informacijos ir komunikacijos moksluose (angl. *library, information and communication sciences*). Pastaruosiuose esminė sąvoka yra informacija. O informatikoje duomenų sąvoka yra svarbesnė negu informacijos.

3. Žmogiškasis pažinimas ir žiniomis grindžiamos sistemos

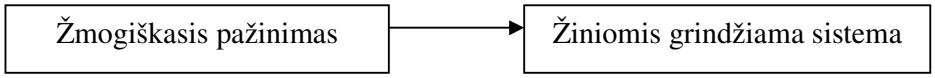
Mes žinių vaizdavimą siejame su dviem sąvokomis, atliekančidviejų tipų siekiais:

1. *žmogiškuoju pažinimu*. Visų pirma, tai žmogui būdingo smalsumo patenkinimas. Antra, pažinimas kaip mokslinė, projektinė veikla, tarp kurios rezultatų yra ataskaita (angl. report) apie tam tikroje dalykinėje srityje atskleistus faktus ir dėsnius;
2. žiniomis grindžiamomis sistemomis.

Į šias dvi sąvokas galime žiūrėti kaip į žmogaus dalykinės veiklos motyvus (arba tikslus):

1. *pažinti* tam tikrą reiškinį (kuris ir sudarytų dalykinę sritį);
2. *sukurti* žiniomis grindžiamą sistemą.

Atrodytų, kad tai alternatyvūs, vienas nuo kito nepriklausomi motyvai. Tačiau labiau įsigilinus galima pastebėti, kad antrajam tikslui pasiekti yra būtinas pirmasis tikslas. Norint sukurti žiniomis grindžiamą sistemą tam tikroje dalykinėje srityje yra būtina šią dalykinę sritį pažinti. Tokiu būdu turime tokį modelį:



Pav. 3.1 Dvi interpretacijos. A) žmogiškasis pažinimas yra autoriui svarbesnis motyvas negu žiniomis grindžiamų sistemos. B) žmogiškasis pažinimas yra būtinas kaip žinių išgavimo stadija kuriant žiniomis grindžiamą sistemą.

Žiniomis grindžiama sistema (ŽGS) (angl. *knowledge-based system*, KBS) – tai kompiuterinė sistema, kuri naudoja žinias. Terminą *žiniomis grindžiama sistema* [Stefik 1995] trumpina iki **žinių sistema** (angl. *knowledge system*). Toks sutrumpinimas yra visiškai priimtinas vienos knygos rėmuose. Tačiau bendru atveju, pavartotas neišskiriant tam tikro konteksto, gali būti suprastas kaip žinių iš tam tikros dalykinės srities, o ne jos apdorojimo būdų, sistema. Čia mes darome analogiją su terminais **informacinė sistema** ir **informacijos sistema**.

Autoriui teko dalyvauti diskusijose, ką šios sąvokos žymi. Informacinė sistema yra informatikoje prigijusi sąvoka. Ji siejama su tuo, *kaip* duomenys yra apdorojami.

Iš informatikos pozicijos mes terminą „informacijos sistema“ aiškintume gramatiškai. Jis žymi, *kas* yra apdorojama. Čia informacija yra kaip apdorojimo abjektas. Čia kalbama, visų pirma, apie informaciją. Informacijos sistema – tai sistema informacijos iš tam tikros dalykinės srities. Tai sistematizuota, struktūrizuota informacija. Terminą „informacijos sistema“ mes sietume su bibliotekininkystės, informacijos ir komunikacijos mokslais. Pavyzdžiui, netgi knygų klasifikavimo pagal mokslo šakas klasifikatorių УДК (Универсальная десятичная классификация) mes galėtume priskirti prie informacijos sistemų.

Žiniomis grindžiama sistema yra atskiras atvejis informacinės sistemos:

Žiniomis grindžiama sistema *is-a* informacinė sistema.

Žiniomis grindžiamos sistemo gyvavimo ciklo modelis būtų specializuotas atvejis informacinės sistemos gyvavimo siklo modelio atžvilgiu. Informacinės sistemos gyvavimo ciklo pirmoji stadija yra reikalavimų analizė. Žiniomis grindžiamų sistemų pirmoji gyvavimo stadija yra **žinių išgavimas** (angl. *knowledge acquisition*), pvz., [Stefik 1995, p. 217]. O žinių išgavimo veikloje yra reikalaujama pažinti dalykinę sritį. Tiesa, **žinių inžinierius** neprivalo dalykinės srities pažinti tokiu dideliu mastu ir taip giliai kaip dalykinės srities **ekspertas**.

Tokiu būdu parodėme, kad žiniomis grindžiamos sistemos sukūrimui yra būtina pažinti dalykinę sritį. Koks šio pažinimo mastas yra jau kitas klausimas.

[Buchanan et al. 1990] rašyti apie žiniomis grindžiamas sistemas pradeda pastebėjimu, kad žinios turi būti prieinamos tiems, kam jos yra reikalingos. Žiniomis grindžiamų sistemų paskirtis yra apibūdinama šitaip: kaip laiku ir tiksliai pateikti žinias, naudingas tam tikrai užduočiai (angl. *task*). Kitaip sakant, kaip profesinę patirtį tam tikroje srityje pateikti sprendimų priėmėjams kada jiems reikia ir kur reikia. Yra įprasta naudoti terminą „ekspertinė sistema“ dėl tokių priežasčių:

1. ŽGS demonstruoja kompetenciją duotoje užduotyje tokiu lygiu, kaip ir duotos dalykinės srities ekspertai;
2. Žinios, esančios ŽGS, yra išgautos iš ekspertų;
3. ŽGS išvedimo metodai yra pagrįsti uždavinių sprendimo technika ir strategijomis, kurias naudoja ekspertai.

Kadangi šios trys charakteristikos yra tik siūlomojo, o ne privalomojo pobūdžio, t.y. nėra nei būtinos, nei pakankamos intelektualioms sistemoms, tai [Buchanan et al. 1990] suteikia pirmenybę terminui žiniomis grindžiama sistema. Terminas ŽGS žymi dirbtinio intelekto taikymą uždavinių sprendimo (angl. *problem-solving*) ir sprendimų priėmimo (angl. *decision-making*) užduotims spręsti.

ŽGS architektūroje yra išskiriamos dvi sudėtinės dalys:

1. samprotavimo (angl. *reasoning*) (arba užduoties sprendimo) procesas;
2. žinių bazė (angl. *knowledge base*)

KBS skirta spręsti tik *tam tikro* tipo užduotims. Ankstyvajame dirbtinio intelekto tyrimų laikotarpyje mokslininkai tikėjosi, kad DI tyrimų rezultatus vainikuos bendras užduočių sprendėjas (angl. *general problem solver*) [Ernst, Newell 1969]. Deja tokio labai universalaus sprendėjo iš principo neįmanoma sukurti. Matematinėje logikoje yra nagrinėjamos algoritmiškai neišprendžiamomis problemos.

Užduočių tipus, kurių sprendimui taikomas DI, klasifikavo [Chandrasekaran et al. 1992]. [Valente 1995, p. 136] išskiria tokias tipiniai (*generic*) DI užduotis: numatymas, valdymas, diagnozė, nurodymas.

4. Žinių vaizdavimo klasifikavimas į procedūrinį ir deklaratyvų

4.1. Procedūrinis žinių vaizdavimas

Programavimas algoritmine kalba yra siejamas su procedūriniu žinių pavaizdavimu. Išnagrinėkime tokį pavyzdį.

Paimkime kvadratinę lygtį pavidalo $ax^2 + bx + c = 0$, jos šaknys yra

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Šaknis x_1 ir x_2 apskaičiuoja šios procedūros:

```
{* Procedūra apskaičiuoja šaknį x1 *}
procedure saknis1 (a, b, c: real; var x1: real);
      x1 := ( -b + sqrt ( b*b - 4*a*c ) ) / ( 2*a)
end

{* Procedūra apskaičiuoja šaknį x2 *}
procedure saknis2 (a, b, c: real; var x2: real);
      x2 := ( -b - sqrt ( b*b - 4*a*c ) ) / ( 2*a)
end
```

Kvadratinę lygtį su koeficientais AA, BB, CC, galime išspręsti su programa, kuri nuosekliai iškviečia x_1 ir x_2 apskaičiavimo procedūras:

```
read(AA, BB, CC);
call saknis1(AA, BB, CC, XX1);
call saknis2(AA, BB, CC, XX2);
writeln(XX1, XX2)
```

Šios lygties šaknų apskaičiavimui toliau pateikiama efektyvesnė procedūra `saknys`. Pastaroji šaknies traukimo veiksmą atlieka tik vieną kartą, o ne du kartus, kaip kviečiant abi procedūras `saknis1` ir `saknis2` nuosekliai.

```
{* Procedūra apskaičiuoja abi šaknis - x1 ir x2 *}
procedure saknys ( a, b, c: real; var x1, x2: real );
      var kint: real;
begin
      kint := sqrt(b*b - 4*a*c);
      x1 := ( -b + kint ) / ( 2*a );
      x2 := ( -b - kint ) / ( 2*a );
end;
```

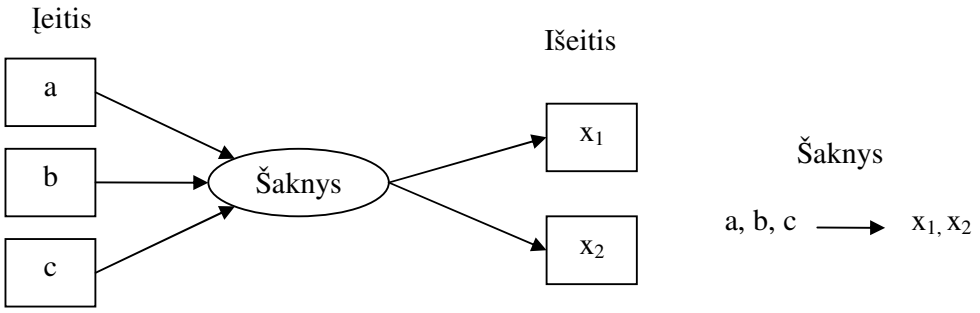
Pirmuoju atveju koncepcija yra lengvesnė, o antruoju vykdymo laikas – trumpesnis.

Kompiuterių programos yra procedūrinio žinių vaizdavimo pavyzdys. Programos pagrindas – algoritmas, tai galima laikyti viena iš žinių vaizdavimo formų.

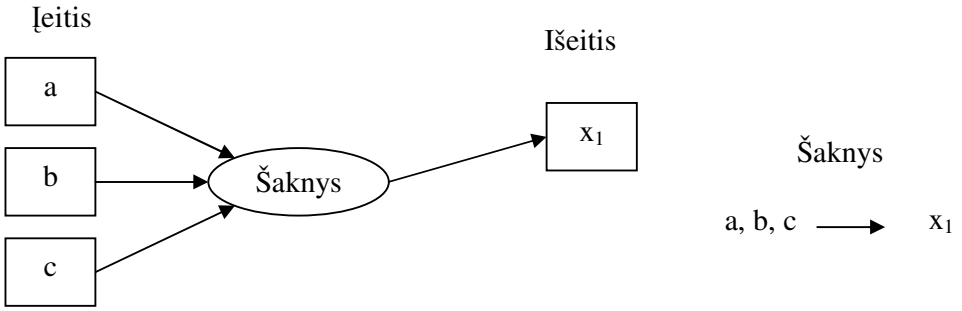
Žinios – tai kas reglamentuota. Dalykinė sritis yra lengviau suvokiama ir valdoma, kai yra hierarchija.

4.2. Deklaratyvus žinių vaizdavimas

Tą patį kvadratinės lygties šaknų suradimo uždavinį deklaratyviai pavaizduojame Pav. 4.1 ir Pav. 4.2



Pav. 4.1 lygties a•x2 + b•x + c = 0 šaknų radimo programos įeities ir išeities pavaizdavimas



Pav. 4.2 Kvadratinės lygties a•x2 + b•x + c = 0 šaknies radimo programos įeities ir išeities pavaizdavimas

Deklaratyvus pavaizdavimas neatsako į klausimą „kaip“. Atsakymas į šį klausimą lieka suprogramuotame modulyje.

Čia IN – įeitis (angl. *input*), t.y. lygties $a \cdot x^2 + b \cdot x + c = 0$ koeficientai a, b ir c. OUT – išeitis (angl. *output*), t.y. šios kvadratinės lygties šaknys x1 ir x2.

Prisiminkime pavyzdį iš dirbtinio intelekto kurso [Čyras 2006]. Čia nagrinėjome producijų sistemą, sudarytą iš trijų produkcijų:

π_1: F, B $\rightarrow$ Z
π_2: C, D $\rightarrow$ F
π_3: A $\rightarrow$ D

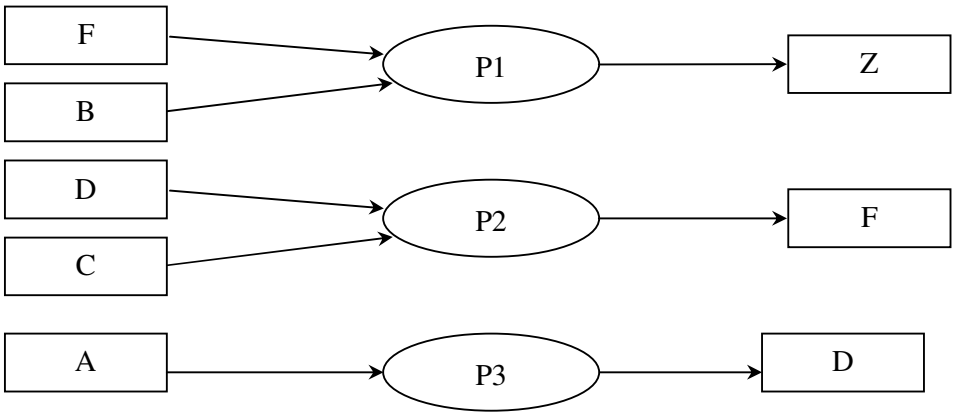
Pav. 4.3 Produkcijų sistema, sudaryta iš tijų produkcijų.

Šias tris produkcijas mes siejame su trimis procedūromis:

procedure P1; Z := f1(B,F)	procedure P2; F := f2(C,D)	procedure P3; D := f3(A)

Pav. 4.4 Susietos produkcijų sistemos.

Čia kalbame apie *funkcinę priklausomybę* (angl. *functional dependency*, sinonimas *data dependency*).



Pav. 4.5 Trijų procedūrų P1, P2 ir P3 iš (1.4) funkcinių priklausomybių tarp įeities ir išeities pavaizdavimas.

Šiame pavyzdyje alfabetas yra visų išeities ir įeities objektų sąjunga {A, B, C, D, F, Z}.
Programos P3 įeitis ir išeitis:

in(P3) = {A}. Sutrumpintai in P3 = {A}
out(P3) = {D}. Sutrumpintai out P3 = {D}

Programos P2 įeitis ir išeitis:

in(P2) = {D, C}
out(P2) = {F}

Programos P3 įeitis ir išeitis:

in(P1) = {F, B}
out(P1) = {Z}

Kokia yra procedūrų P1 ir P2 nuoseklios kompozicijos (paprastai kalbant, sekos) „P1;P2“ įeitis ir išeitis?

1.1 apibrėžimas. Dviejų procedūrų sekos įeitis ir išeitis apibrėžiama tokiomis formulėmis:

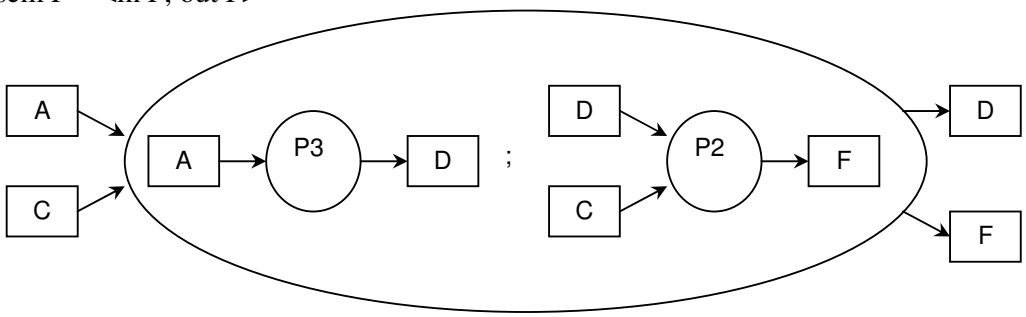
$in(P_1, P_2) = in\ P_1 \cup (in\ P_2 / out\ P_1)$
$out(P_1, P_2) = out\ P_1 \cup out\ P_2$

Paaiškinsime pradėdami nuo išeities.
Dviejų programų sekos išeitis yra lygi šių programų išeičių sąjungai.
Dviejų programų įeitis yra lygi sąjungai, kurią sudaro P_1 įeitis ir ta P_2 dalis, kuri neįeina į P1 išeitį.

1.2 apibrėžimas. Programos P *semantika* vadiname porą, kurią sudaro šios programos įeitis ir išeitis <in P, out P>.

Programos P semantimą žymime sem(P) arba sutrumpintai sem P. Tokiu būdu pagal apibrėžimą turime:

sem P = <in P, out P>



Pav. 4.6 Dviejų procedūrų P3 ir P2 iš (1.4) įeities ir išeities pavaizdavimas

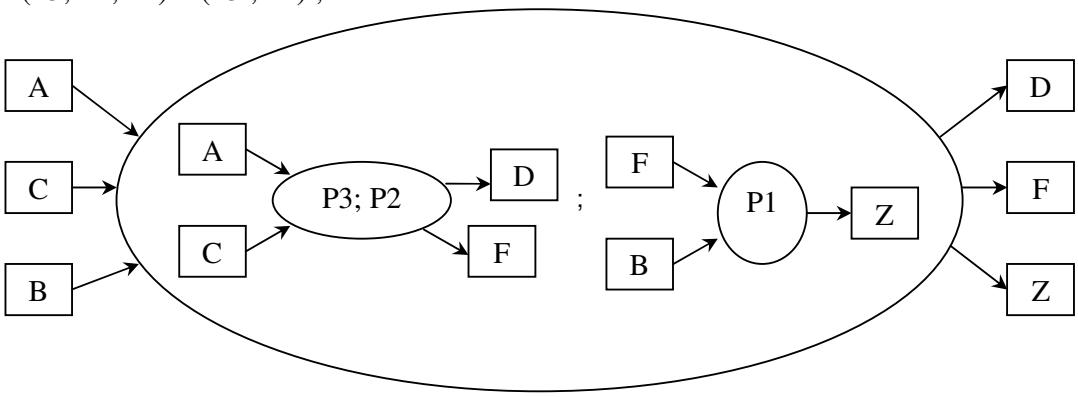
Taip galime pavaizduoti grafiškai bei užrašyti:

$$\{A, C\} \rightarrow \left[\begin{array}{l} \text{call P3 ; (tai ekvivalentu D:= f1(A))} \\ \text{call P2 (tai ekvivalentu F:= f2(D, C))} \end{array}\right] \rightarrow \{D, F\}$$

Programų nuosekli kompoziciją žymime „;“. Tai operacija tarp programų.

Trijų programų nuosekli kompozicija yra apibrėžiama rekurentiškai, t.y., kaip nuosekli kompozicija pirmųjų dviejų programų, o paskui dar nuosekli kompozicija su trečiąja programa:

(P3; P2; P1) = (P3 ; P2) ; P1



Pav. 4.7 Trijų procedūrų P3, P2 ir P1 iš (1.4) įeities ir išeities pavaizdavimas per „P3;P2“ ir P1 nuoseklią kompoziciją.

„P3; P2; P1“ įvestis:

$in((P3;P2);P1) = in(P3;P2) \cup (in(P1)/out(P3;P2)) = \{A,B\} \cup \{\{B,F\}/\{D,F\}\} = \{A,C,B\}$

„P3; P2; P1“ išvestis:

$out((P3;P2);P1) = out(P3;P2) \cup out(P1) = \{\{D,F\} \cup \{Z\}\} = \{D,F,Z\}$

Operacijų tarp aibių terminais įrodėme:

in((P3;P2);P1) = {A,C,B}
out((P3;P2);P1) = {D,F,Z}

Šiame etape žinias traktuojame, kaip porą <įvestis, išvestis>:

$in(P_1;P_2;P_3) = in(P_1;P_2) \cup (in(P_3)/out(P_1;P_2)) =$
$= in(P_1) \cup (in(P_2)/out(P_1)) \cup (in(P_3)/(out(P_1) \cup out(P_2)))$

$out(P_1;P_2;P_3) = out(P_1;P_2) \cup out(P_3) = out(P_1) \cup out(P_2) \cup out(P_3)$

Bendruoju atveju:

1.3 teorema. Programų P_1, P_2, ..., P_n nuoseklios kompozicijos įeitis ir išvestis yra lygi:

$$in(P_1;P_2;...;P_n) = \bigcup_{i=1}^{n} (in(P_i)/\bigcup_{j=1}^{i-1} out(P_j))$$

$$out(P_1;P_2;...;P_n) = \bigcup_{i=1}^{n} out(P_i)$$

Čia Y žymi aibių sąjungą ∪.

Tai įrodyti galima matematinės indukcijos keliu, tačiau tą padaryti paliksime skaitytojui. Programų P1, P2, ..., P_n nuoseklios kompozicijos išvestis yra lygi atskirų programų išvesčių sąjungai. O įvestis lygi sąjungai, kur iš P_i įeities atimamos P1, P2, ..., P_i išvestys.

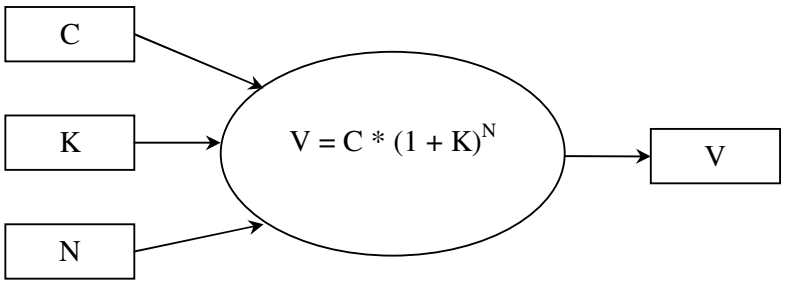
Nuoseklios kompozicijos esmė yra tame, kad anksčiau iškviestos programos pagamina rezultatus vėliau kviečiamos programoms. Kitais žodžiais, ankstesni sekos nariai „maitina“ vėlesnius.

4.3. Įeities – išeities vaizdavimo pavyzdys

Programos įeities-išeities pavaizdavimui paimkime kitą pavyzdį. Nagrinėkime klasikinę kapitalo formulę:

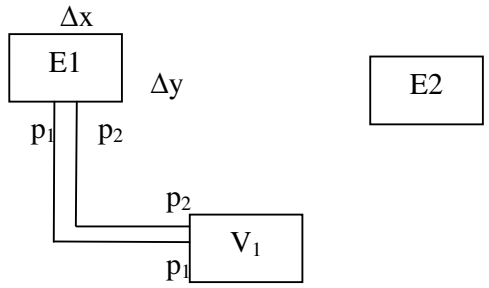
$$V = C * (1 + K)^N$$

kur V yra gaunama vertė, įnešus į banką pinigus C ir palaikius N metų, kai palūkanų norma K (pavyzdžiui, K=0,03, kai metinės palūkanos yra 3 procentai).



Pav. 4.8 Programos, kuri skaičiuoja kapitalo C vertę V po N metų esant palūkonoms K, įeities ir išeities pavaizdavimas.

Panagrinėkime kitą pavyzdį. Turime tokią situaciją – reikia trasuoti spausdinto montažo plokštę (angl. *printed circuit board*). Toks uždavinys yra būdingas lustų projektavime projektavimo automatizavimo sistemomis (angl. *computer aided design*, CAD). Plokštumoje yra dėstomi ir trasuojami elektronikos elementai: mikroschemos, tranzistoriai, varžos ir kt. Trasavimo dalykinėje srityje yra nustatyti tam tikri reikalavimai, pavyzdžiui, grandinės negali kirstis. Užduotis: nustatyti, kaip sujungti tarpusavyje *portus*, t.y. kaip išvedžioti grandines tarp duotų portų porų.



Pav. 4.9 Procedūrinių ir deklaratyvių žinių atskyrimas spaudinto montažo plokščių projektavime trasavime. Deklaratyvios žinių pavyzdys yra jungiamų portų poros. Procedūrinės žinios apima trasavimo algoritmus. Jungiamų portų poros: E1p1 → V1p1, E1p2 → V1p

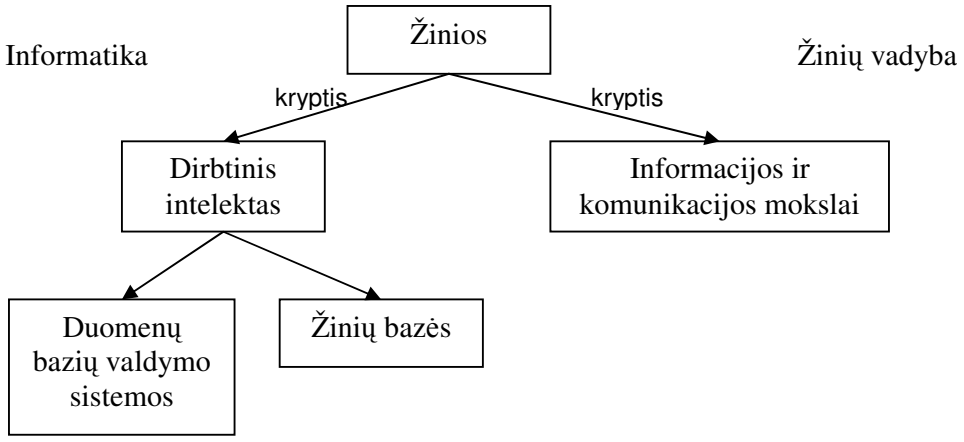
Čia E1, E2, V1 – elementai, kurie dedami ant spausdinto montažo plokštės; Δx ir Δy – tų elementų matmenys; p_1, p_2 – trasuojami elementų portai.

Šias žinias vaizduosime deklaratyviai. Pvz., atstumas tarp kontaktų (normatyvas – konstanta), storis, elementų matmenys E1.Δx, E1.Δy ir kt.

Konstantos vaizduojamos *deklaratyviai*. Jos išsaugomos duomenų bazėje. Deklaratyvus vaizdavimas siejamas su duomenų baze. Trasavimo algoritmai yra siejami su procedūriniu žinių vaizdavimu.

Deklaratyvus ir procedūrinis žinių pavaizdavimas labai susiję. Griežtos ribos tarp jų nėra.

5. Sąvoka „žinios“



Pav. 5.1 Žinių užimama vieta Informatikos ir žinių vadybos kontekste.

Šį grafa galima pavadinti semantiniu tinklu. Žinių vaizdavimas – tai žinių vertimas duomenimis. Žinios skirstomos į dvi pagrindines šakas t.y. dirbtinį intelektą bei informacijos komunikacijos mokslus. Detaliau nagrinėsime informatikos kryptį, dirbtinio intelekto šakas. Duomenų bazių valdymo sistemos siejamos su sąvoka išrinkimas (retrieval). Žinių bazės siejamos su sąvokomis: išplaukimas, sąlygojimas (logical entailment), samprotavimas (reasoning). Tuo tarpu sąvoka samprotavimas (reasoning) yra siejama su (loginių) išvedimu (inference).

Jeigu programa išrenka informaciją (information retrevial), tai jos sudėtingumas gali būti polinominis arba kvadratinis. Pavyzdžiui, turime atlyginimų lentelę:

Pavardė	Atlyginimas
Adomaitis	1000
Adukas	900
Birmantas	1200
Budraitis	800
Caraitis	1000
...	...
Žabaitis	1800

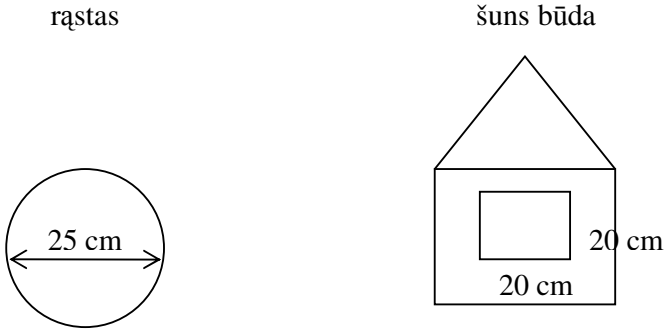
Pav. 5.2 Atlyginimų lentelė.

Turėdami Pav. 5.2 pateiktą duomenų bazę ir norėdami surasti, kokį atlyginimą gauna konkretus asmuo, pateikiame užklausą, pavyzdžiui, „Budraitis = ?“. Jei masyvas yra surūšiuotas, tai informacijos radimo sudėtingumas O(ln(N)). Jei masyvas yra nesurūšiuotas, tokiu atveju yra naudojamas perrinkimas. Perrenkant nesurūšiuotą masyvą – sudėtingumas yra O(N). Kur yra žinių bazės, ten sudėtingumas siekia $O(2^{n)}$. Pavyzdžiui, keliaujančio pirklio uždavinys, kai turime N miestų:

Būna atvejų, kai viršijamas eksponentinis sudėtingumas. Pavyzdžiui:

Turime aibę faktų $\{s_1, s_2, s_3...s_N\} \vdash P$. Pateikiame užklausą, norėdami sužinoti, ar iš pateiktų faktų išvedama P. Šito uždavinio realizavimas viršija eksponentinį sudėtingumą.

Būna atvejų, kai susiduriama su neįsprendžiamomis problemomis. Tuomet keičiamas problemos formulavimas. Neišsprendžiamų problemų pavyzdys:



Pav. 5.3 Neišsprendžiama problema

Reikia patalpinti 25 cm skersmens rąstą į 20 cm pločio kvadratinį įėjimą turinčios šuns būdos. Matematiškai rąstas netelpa. Tačiau jeigu pakeisime problemos formulavimą ir žiūrėsime į rąstą kaip į lygiagrečių plokščių rinkinį (kurias gautume supjausčius rąstą išilgai), kurių plačiausia 25 cm, o kitos mažesnės, tai tas plokštes galima prakišti į būdą, nes skersai tilps 20 * sqrt(2) = 28,28 cm.

Problemos yra formuluojamos, o uždaviniai yra sprendžiami.

Galima iškelti sau klausimą: „kas yra geriau: veržtis pro uždaras duris ar veržtis pro atviras duris?“. Veržtis pro atviras duris, akivaizdžiai beprasmiška. Veržtis pro uždaras duris, lyg irgi beprasmiška. Tačiau, tai yra prasmingiau, nes galbūt pavyks arba bus bent jau pastangos tikslo link.

6. Logikos vaidmuo samprotavime

Šis skyrius pradedamas pagal Brachman, Levesque knygos įvadą.

Samprotavimo pavyzdys logikoje:

Teiginiai:

1. pacientas Jonas yra alergiškas medikamentui M;
2. kas yra alergiškas medikamentui M, yra alergiškas ir medikamentui M‘;

Šie du teiginiai sąlygoja, kad medikamentą M‘ draudžiama skirti Jonui.

Logikoje tai įrodoma taip:

Duota:

1. Alerg (Jonas,M)
2. $\forall$x(Alerg (x,M) $\rightarrow$ Alerg(x, M‘))

Įrodyti: Alerg (Jonas, M‘)

Įrodymas

1. žingsnis:

$\forall$x (Alerg (x, M) $\rightarrow$ Alerg (x, M‘)) $\forall$x (P(x))/P(a) (vietoj kintamojo įrašomos
Alerg (Jonas, M) $\rightarrow$ Alerg (Jonas M‘) konstanta)

2 .žingsnis

Alerg (Jonas, M) $\rightarrow$ Alerg (Jonas, M‘) taikome *modus ponens* taisyklę.
Alerg (Jonas, M)
Alerg(Jonas,M‘)

Naudojantis predikatų logikos taisyklėmis įrodėme matematiškai, kad medikamentą M‘ negalima skirti Jonui.

Logika nagrinėja tik pačią struktūrą, o turinio neliečia. Bendresne prasme predikatai suprantami kaip teiginiai su parametrais t.y. tvirtinamojo pobūdžio sakiniai, kuriose konkretizuoti požymiai, o ne objektai. Reikia nurodyti tik objektų kitimo aibę. Išnagrinėsime kitą pavyzdį:

Sokratas yra žmogus;
visi žmonės yra mirtingi;
Sokratas yra nemirtingas.

Perrašome teiginius formulėmis:

Duota:

1. žmogus(Sokratas);
2. $\forall$x(žmogus (x)) $\rightarrow$ mirtingas(x))

Įrodyti: $\neg$ mirtingas(Dzeusas)

Įrodymas:

1. žingsnis:

$\forall x(\text{žmogus}(x)) \rightarrow \text{mirtingas}(x)$ $\forall x(P(x))/P(a)$ (vietoj kintamojo įrašomos
Žmogus(Sokratas) → mirtingas(Sokratas) konstanta).

2. žingsnis:

Žmogus(Sokratas) → mirtingas(Sokratas) taikome *modus ponens* taisyklę.
žmogus(Sokratas)
mirtingas(Sokratas)

Gavome, kad Sokratas yra mirtingas.

Kitas uždavinys:

1. Dzeusas yra Dievas.
2. Dievai yra nemirtingi.

Įrodyti: Dzeusas yra nemirtingas

Įrodymas:

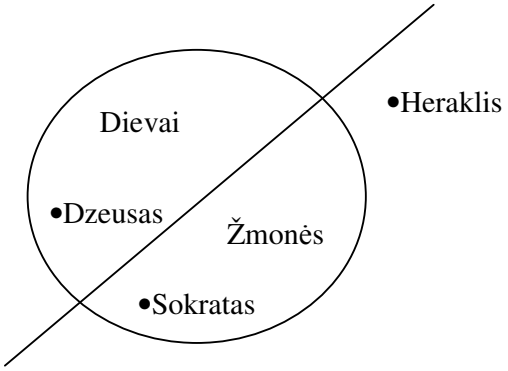
1.žingsnis:

$\forall x(\text{Dievas}(x)) \rightarrow \neg\, \text{mirtingas}(x)$ $\forall x(P(x))/P(a)$ (vietoj kintamojo įrašoma
Dievas(Dzeusas) → ¬ (mirtingas(Dzeusas)) konstanta).

2.žingsnis:

Dievas(Dzeusas) → ¬ (mirtingas(Dzeusas)) taikome *modus ponens* taisyklę.
Dievas(Dzeusas)
¬ (mirtingas(Dzeusas))

Šiame uždavinyje Dievas Dzeusas yra nemirtingas.
Neturėdami teiginio Dievas(Sokratas), išvesti, kad jis yra nemirtingas negalime.

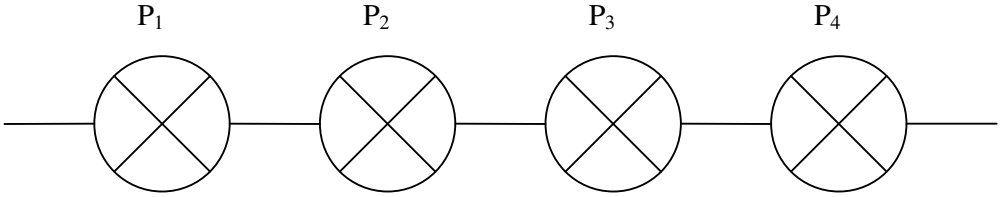


Pav. 6.1 Objektų kitimo aibės: "Dievai" ir "Žmonės"

Atviras žinių modeliavimo klausimas:
kokiai aibei priskirti legendos herojų Heraklį – ar dievams, ar žmonėms?
Logika – nėra vaistas, jinai pati savaime neišsprendžia problemų.

$S \rightarrow p$

Tegu yra 4 lempučių girliandą, tačiau jinai nedega:
$p_1 \,\&\, p_2 \,\&\, p_3 \,\&\, p_4 = \text{false}$
Norime gauti $\exists p_i = \text{false}$, bet mes nežinome kuri būtent. Šiuo atveju logika niekuo nepadės. Diagnozės problema, tačiau diagnozė yra nedeterminuota. Galbūt $p_1 = \text{false}$ arba $p_2 = \text{false}$ arba $p_1 \,\&\, p_2 = \text{false}$ ir t.t.



Pav. 6.2 Keturių lempučių grilianda

7. Žinių vaizdavimo būdų apžvalga

Deklaratyvaus žinių vaizdavimo būdus yra priimta skirstyti [Sowa 2000; Brachman, Levesque 2004] į keturias grupes:

1. loginis žinių vaizdavimas;
2. procedūrinis žinių vaizdavimas (siaurąja prasme);
3. tinklinis žinių vaizdavimas;
4. struktūrinis žinių vaizdavimas.

Toliau ir apžvelgsime aukščiau išvardintas žinių vaizdavimo būdų grupes.

7.1. Loginis žinių vaizdavimas

Loginio žinių vaizdavimo būduose remiamasi:

1. teiginių logika, kitaip dar teiginių skaičiavimu (angl. proposition calculus);
2. predikatų logika, kitaip dar predikatų skaičiavimu (angl. predicate calculus).

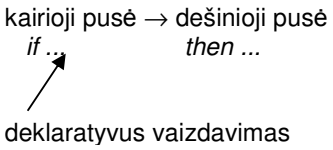
Apibendrintai yra sakoma, kad loginio žinių vaizdavimo būdai remiasi *pirmos eilės* logika. Ir teiginių logika, ir predikatų logika paprastai yra nagrinėjamos matematinės logikos kurse.

7.2. Procedūrinis žinių vaizdavimas (siaurąja prasme)

Procedūrinis žinių vaizdavimas siaurąja prasme supaprastintai yra suprantamas kaip produkcijų sistema.

Produkcijų sistemos pavyzdys:

$$\pi_1 : E, C \to Z$$
$$\pi_2 : B, D \to E$$
$$\pi_3 : A \to D$$



Logikoje: antecedentas → konsekventas. Siejama su implikacijos operacija.

7.3. Tinklinis žinių vaizdavimas

Anglų kalbos terminas yra *network representation schema*, sutrumpintai tiesiog *network representation*. Tinklinio žinių vaizdavimo būdai apima:

1. semantinius tinklus (angl. semantic networks);
2. koncepcinius grafus (angl. conceptual graphs).

Abi šios sąvokos yra giminingos. Yra autorių, kurie kalbėdami apie tą patį naudoja arba pirmąjį, arba antrąjį terminą.

Semantinis tinklas yra grafas, kuriuo vaizduojamos klasifikavimo žinios apie objektus ir jų savybes. Viršūnėmis vaizduojamos sąvokos, o briaunomis santykiai.

Yra priimta išskirti šių rūšių santykius, vadinamus universaliaisiais:

1. is-a;
2. instance-of;
3. part-of.

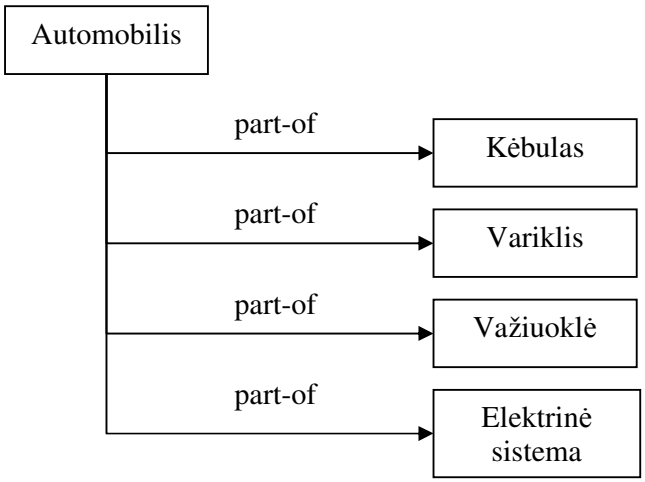
Tai bendrieji, nuo dalykinės srities nepriklausomi santykiai.

Dalykinės srities žinių visuma remiasi *konceptualizacija*. Pastaroji apima objektus, sąvokas ir kitas esybes, kurios priimta laikyti egzistuojančiomis dalykinėje srityje, ir sąryšius tarp jų [Genesereth, Nilsson 1987]. Konceptualizacija yra abstraktus, supaprastintas pasaulio vaizdas, kurį norima pavaizduoti tam tikru tikslu.

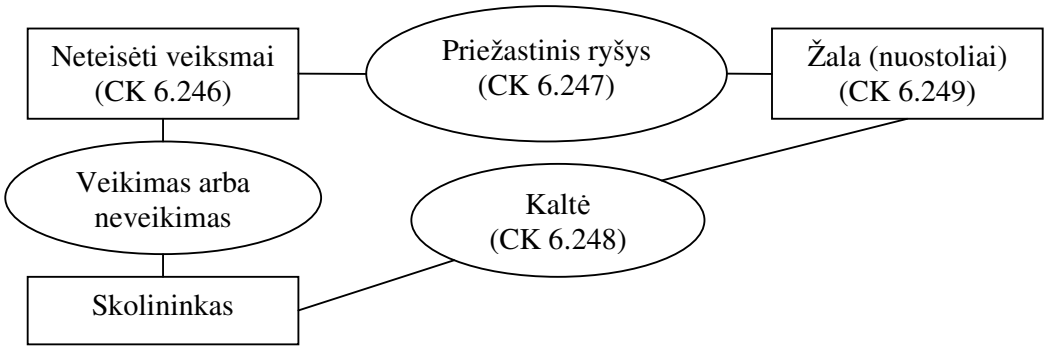
Informatikoje terminu **ontologija** yra vadinama tam tikros dalykinės srities sąvokų visumos specifikavimas išreikštiniu pavidalu (angl. *explicit specification of a conceptualization*) [Gruber 1993]. Terminas ontologija yra paimtas iš filosofijos, kur ontologija yra suprantama kaip būties teorija, o epistemologija yra pažinimo teorija. Supaprastintai galima sakyti, kad filosofijoje ontologija atsako į klausimą „kas yra“, o epistemologija – „kaip mes pažįstame“.

Ontologija galima apibūdinti kaip sąvokų sąvadą. Tradiciškai sąvokų atributai neįeina į ontologijas. Pavyzdys:

Žemiau esantis grafas vaizduoja teiginį, jog automobilių sudėtinės dalys yra kėbulas, variklis, važiuoklė ir elektrinė sistema



Pav. 7.1 Automobilių sudėtinės dalys yra kėbulas, variklis, važiuoklė ir elektrinė sistema



Pav. 7.2 Sąvokos „Civilinė atsakomybė“semantinis grafas pagal Lietuvos Respublikos civilinio kodekso 6.245 straipsnį.

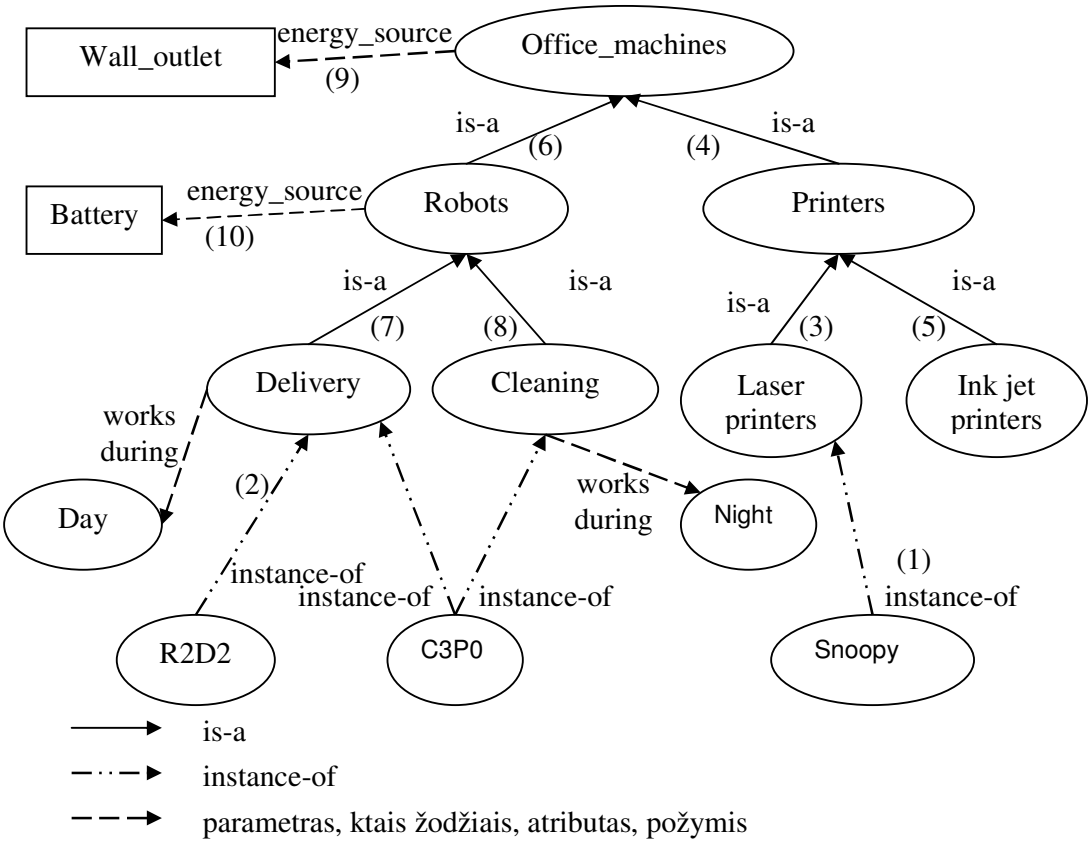
CK 6.245 Civilinė atsakomybė:

1. neteisėti veiksmai CK 6.246 straipsnis;
2. kaltė CK 6.248 straipsnis;
3. priežastinis ryšys CK 6.247 straipsnis;
4. žala CK 6.249 straipsnis.

8. Semantinis tinklas

Vienas iš pavaizdavimo būdų – semantinis tinklas. Semantinio tinklo sąvoka atsirado XIX a. Tai logikos užrašymo forma. Semantiniai tinklai neturi tikslaus apibrėžimo, kaip, pavyzdžiui grafai. Semantinį tinklą galima apibūdinti kaip grafą, kurio sąvokos vaizduoja dalykinės srities sąvokas, o briaunos vaizduoja santykius. Semantinis tinklas – grafas, kurio viršūnėms ir briaunoms suteikiamos semantikos. Pagrindinė šių tinklų motyvacija – neformali struktūra. Semantinius tinklus verta naudoti tada, kai yra paveldėjimas.

Paimkime pavyzdį iš [Nilsson 1998 p. 308-313] 18.3 poskyrio „Žinių vaizdavimas tinklais“. Jame yra pateikiama tokia biuro technikos klasifikacija (žr.Pav. 8.1.):



Pav. 8.1 Semantinis tinklas, vaizduojantis biuro technikos klasifikaciją [Nilsson 1998, p. 308-313]

Paaiškinimai Pav. 8.1.: biuro įrenginiai yra, pirma, robotai (rūšių Delivery ir Cleaning, Delivery egzempliorius – R2D2 bei Delivery robot ir Cleaning robot bendras egzempliorius C3P0) ir , antra, spausdintuvai (rūšių Laser ir InkJet, lazerinio spausdintuvo egzempliorius yra Snoopy).

Kiekvienas biuro įrenginys yra prijungiamas prie elektros srovės, t.y. turi savybę Wall-outlet. Biuro įrenginio Robot energijos šaltinis yra Battery, robotų rūšis Delivery robots turi savybę – dirba dieną (Day), cleaning robots turi savybę dirba naktį (Night). Tačiau kol kas į šias savybes dėmesio nekreipkime ir nagrinėkime be jų. Prieš tai dar atkreipsime dėmesį į tai,

kad loginis išvedimas pas mus realizuotas – kaip paveldėjimas. Turėdami pavyzdį (C3P0) kur yra dvigubas paveldėjimas susiduriame su problema, kaip realizuoti šio objekto savybes.

Paaiškinkime numeriukais pažymėtus sąryšius:

(1) Laser_printer (Snoopy)

(2) Delivery_robot (R2D2)

(3) $\forall x$ [Laser_printer (x) $\Rightarrow$ Printer(x)]

(4) $\forall x$ [Printer (x) $\Rightarrow$ Office_machine(x)]

(5) $\forall x$ [Ink_jet_printer (x) $\Rightarrow$ Printer(x)]

(6) $\forall x$ [Robot (x) $\Rightarrow$ Office_machine(x)]

(7) $\forall x$ [Delivery_robot (x) $\Rightarrow$ Robot(x)]

(8) $\forall x$ [Cleaning_robot (x) $\Rightarrow$ Robot(x)]

(9) $\forall x$ [Office_machine (x) $\Rightarrow$ Energy_source(x) = Wall_outlet]

(10) $\forall x$ [Robot (x) $\Rightarrow$ Energy_source(x) = Battery]

Duotoje dalykinėje srityje yra teisingi tokie klasifikavimo teiginiai ($\Rightarrow$ žymi implikaciją):

$\forall x$[Laser printer(x) $\Rightarrow$ Printer(x)], t.y. bet kuris lazerinis spausdintuvas yra spausdintuvas.

$\forall x$[Printer(x) $\Rightarrow$ Office_machine(x)], t.y. bet kuris spausdintuvas yra biuro įrenginys.

$\forall x$[Office_machine(x) $\Rightarrow$ [Energy_source(x)=Wall_outlet]], t.y. kiekvienas biuro įrenginys yra maitinamas iš rozetės.

Remiantis šiais teiginiais logiškai yra išvedama:

Energy_source(Snoopy) = Wall_outlet

t.y. spausdintuvo Snoopy energijos šaltinis yra rozetė.

Mūsų pavyzdyje iškyla problema nusakant objekto C3P0 funkciją, kada jis dirba. Nes pagal mūsų schemą jis paveldi funkcijas, kad dirba ir dieną, ir naktį.

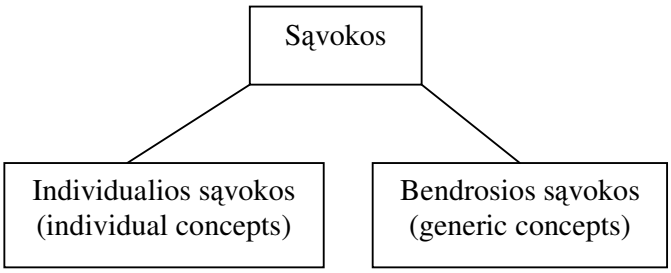
Tokia problema sprendžiama taip:

1. nustatomi prioritetai;
2. nustatomos reikšmės pagal nutylėjimą (angl. *by default*).

Semantinis tinklas – tai grafas, kurio briaunoms suteikta prasmė.

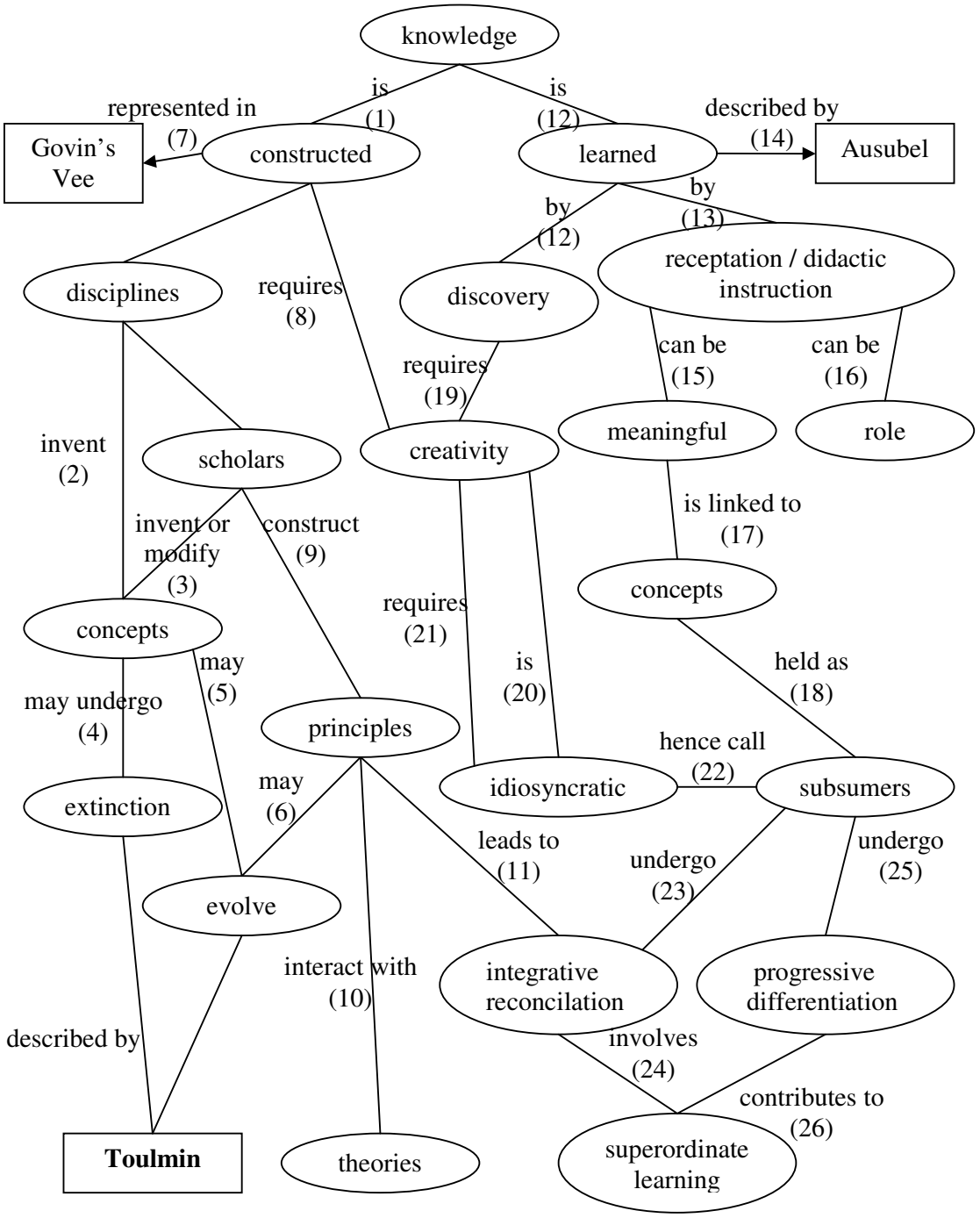
Semantinis grafas G = <V, E>, kur $E \subset V \times V$.

Šiame grafe viršūnės atspindi sąvokas, o briaunoms yra suteikta tam tikra prasmė.



Pav. 8.2 Sąvokų klasifikavimas į individualias ir bendrąsias

Semantinio grafo sąvoka įvesta XIXa. norint pavaizduoti tekstinės eilutės prasmę. Paimkime pavyzdį iš [Geldenhuys 1999, p. 13] (žr.Pav. 8.3 žemiau).



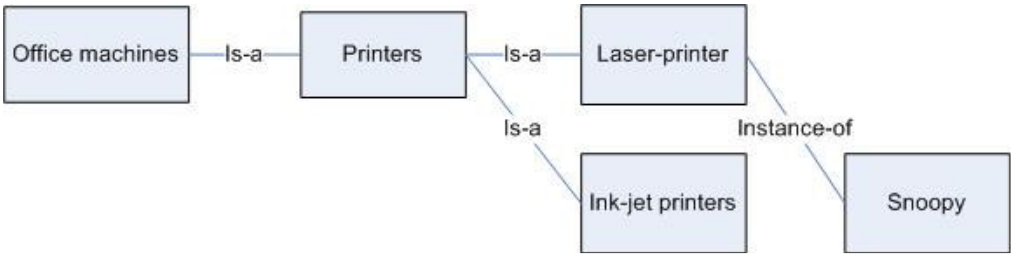
Pav. 8.3 Semantinio tinklo pavyzdys iš [Geldenhuys 1999].

Semantinam tinklui Pav. 8.3. yra suteikta tokia prasmė. Jį galima taip interpretuoti kaip šių sakinių aibę:

1. Knowledge is constructed by disciplines with scholars.
2. Knowledge is learned by discovery.
3. Scholars invent or modify concepts combined in principles.
4. Concepts may undergo extinction described by Toulmin.
5. Concepts may envelope.
6. Principles may envelope.

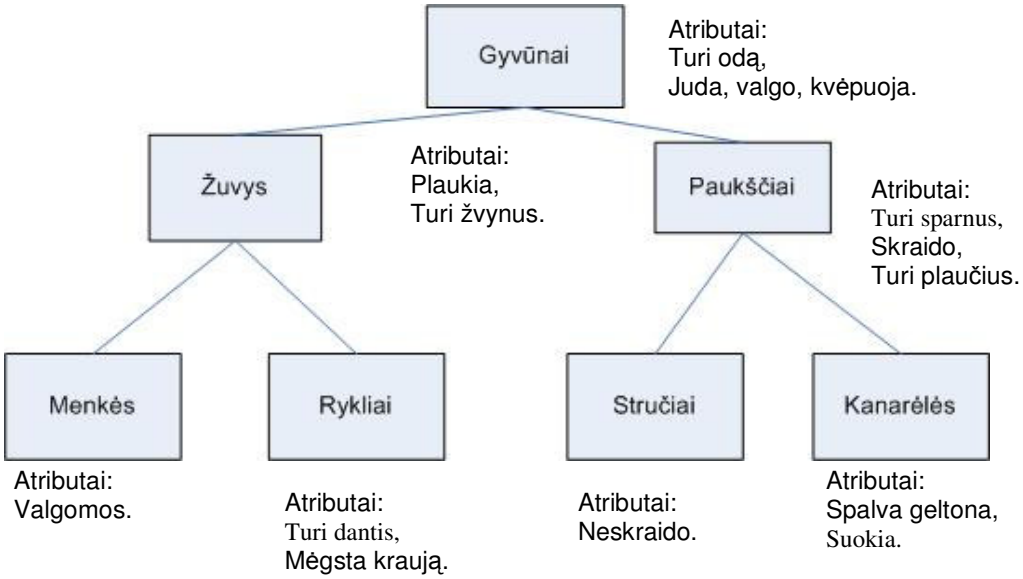
7. Constructe knowledge ir represented in Gorvins‘s Vee
8. Knowledge constructed requires creativity.
9. Scholars construct principles.
10. Principles interact with theories.
11. Principles lead to integrative reconcilation.
12. Knowledge is learned by discovery.
13. Knowledge is learned by receptation / didactic instruction
14. Knowledge is learned described by Ausubel.
15. Receptation / didactic instruction can be meaningful.
16. Receptation / didactic instruction can be role.
17. Meaningful is linked to concepts.
18. Concepts hel as subsumers.
19. Discovery requires creativity.
20. Creativity is idiosyncratic.
21. Idiosyncratic requires creativity.
22. Subsumers hence call idiosyncatic.
23. Subsumers undergo integrative recincilation.
24. Integrative recincilation involves superordinate learning.
25. Subsumers undergo progressive differentiation.
26. Progressive differentiation contributes to superordinate learning.

Grįžkime prie pavyzdžio apie biuro mašinas Pav. 8.1.



Pav. 8.4 Semantinis tinklo iš Pav. 8.1. fragmentas, iliustruojantis bendrines sąvokas ir individualią sąvoką Snoopy.

Is-a yra santykis tarp bendrinių sąvokų.
Instance-of yra santykis tarp individualios sąvokos ir bendrinės sąvokos.
Bandykime šiek tiek šias sąvokas paaiškinti.

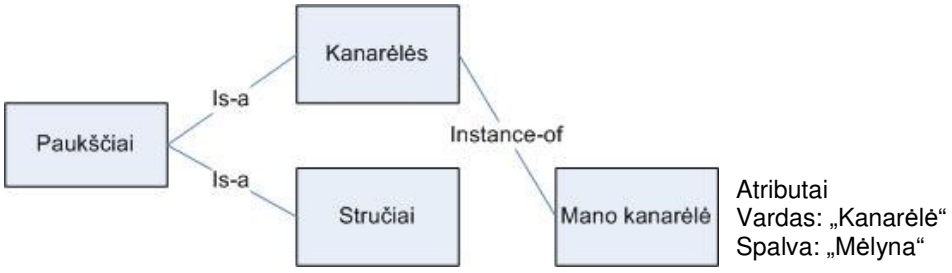


Pav. 8.5 Gyvūnų klasifikavimo pavyzdys.

Pastebėkime, kad šiame grafe pavaizduotos klasės ir jų sąryšiai. Programuotojai klasę rašytų vienaskaita, pavyzdžiui, gyvūnas, žuvis, paukštis. Kalbant apie žmonių supratimą, kaip iš šiuo atveju, naudojama daugiskaita.

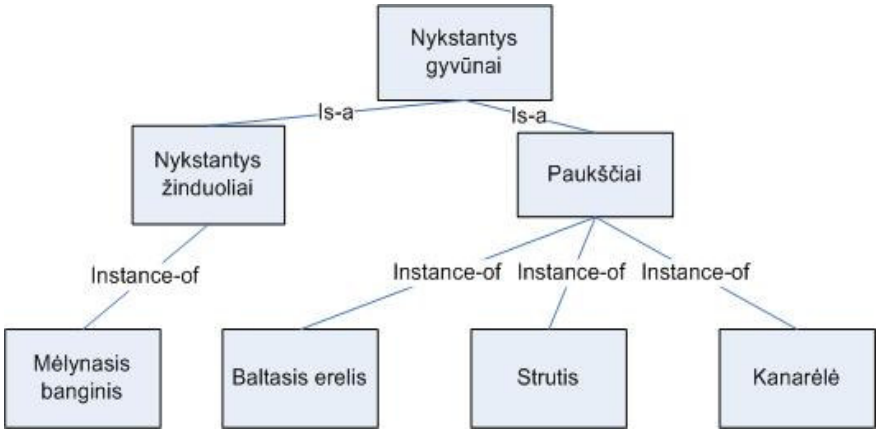
Iškelkime klausimą: kas yra kanarėlė – egzempliorius ar rūšis.

Pirmiausia galime galvoti, kad kanarėlė yra rūšis (is-a) (Pav. 8.6)



Pav. 8.6 „Kanarėlės“ kaip bendrinė sąvoka, „Mano kanarėlė“ – individuali sąvoka.

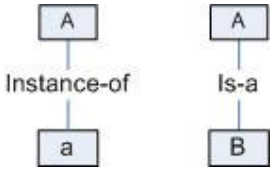
Tačiau toks pat teisingas galėtų būti ir kitas Pav. 8.5 interpretacijos variantas (žr.Pav. 8.7), kuriame „Kanarėlės“ yra individuali sąvoka.



Pav. 8.7. „Kanarėlė“ kaip individuali sąvoka. Ryšys instance-of.

Šis pavaizdavimas leidžia medžio (grafo) lapus traktuoti kaip individualias sąvokas. Tradiciškai galima būtu naudoti is-a ryšį, tačiau šiuo atveju negalima, nes kalbama apie nykstančias rūšis t.y. individus. Tačiau vien iš sąvokų neaišku ar tai individuali ar bendrinė. sąvoka. Ta pati sąvoka vienam kontekste gali būti suprasta kaip individuali, kitam kaip bendrinė.

Iš teksto taip paprastai neištrauksime žinių, reikia žinoti kontekstą. Pavyzdžiui, ar banginiai yra žuvys.



Pav. 8.8 Galima tokia interpretacija: A – bendrinė sąvoka, a – individuali sąvoka, B – individuali sąvoka, kur $a \in A$ ir $B \subset A$.

Pirmuoju atveju turime:
$a \in A$
Antruoju atveju:
$B \subset A$

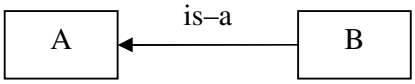
Nors gramatiškai abiem atvejais skaitysime vienodai:

a yra A (angliškai a is A) ir B yra A (angliškai B is A)

8.1. Ryšio is-a interpretacijos

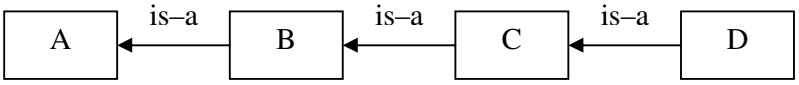
Ryšys is-a gali turėti tokias interpretacijas.

1. Poaibis ir viršaibis (subset/superset). Tokią interpretaciją galima paaiškinti: A ir B yra sąvokos. Tačiau neaišku, ar tai bendrinės ar individualiosios sąvokos.



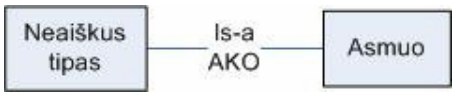
Pav. 8.9 is-a ryšys.

2. Apibendrinimas ir specializacija (generalization/specialization). Suprantame taip, kad elementas A turi atributus. Elementas B turi atributus Traktuosime kad šiems predikatams galiota tokia specializacija:$\forall$ x[B(x) $\Rightarrow$ A(x)].



Pav. 8.10 is-a ryšys tarp keletos elemantų.

3. AKO (A-kind-of).

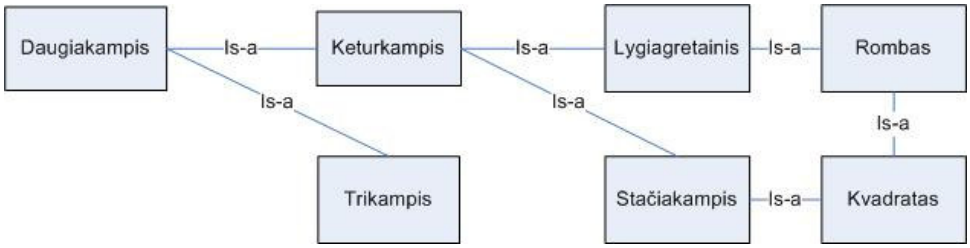


Pav. 8.11 is-a AKO ryšys

Šioje schemoje nenorime suteikti „sąvokos“ („klasės“) vaidmens atributams. Naudojama buitinė sąvoka „Neaiškus tipas“. Nesuteikiant klasės reikšmės yra paliekama galimybė paveldėti. Pavyzdys:

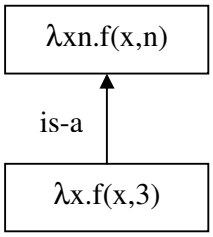
Į degalinę užsukęs asmuo iš pradžių gali būti laikomas neaiškiu tipu. Negalima nusakyti tikslo, dėl kurio jis apsilankė degalinėje: ar apsipirkti, ar apsižvalgyti, ar vogti. Jeigu išsirinkęs prekę (-es) paduoda kreditinę kortelę – vadinasi jis yra pirkėjas.

4. Conceptual containment (koncepcinis apėmimas, apribojimas). Konceptualus apėmimas – iš viršaus į apačia. Apribojimas – iš apačios į viršų.



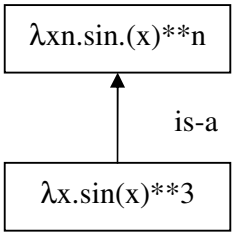
Pav. 8.12 is-a ryšys tarp geometrinių figūrų

Kitas pavyzdys galėtų būti λ abstrakcija. Pavyzdžiui, turime funkciją nuo dviejų argumentų λxn.f(x,n). Jos specializacija yra funkcija nuo vieno argumento λx.f(x,3). Pastaroji gaunama, kai n reikšmę sukonkretiname n=3. Bendrąja prasme atrodo taip:



Pav. 8.13 is-a ryšys su lambda abstrakcija

Pateikiame konkretų pavyzdį, kuriame skaičiuojama funkcija sin(x)**3:



Pav. 8.14 is-a ryšys funkcijoje sin(x)**3

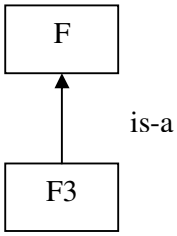
Programavimas algoritmine kalba bendruoju atveju atrodo taip:

```
Procedure F(x,n)
     return sin(x)**n;
```

Kai mus domina tik atvejis, kai n = 3, tai pseudokodą galime optimizuoti kėlimą laipsniu keičiant sandauga:

```
Procedure F3(x)
     return sin(x)* sin(x)* sin(x);
```

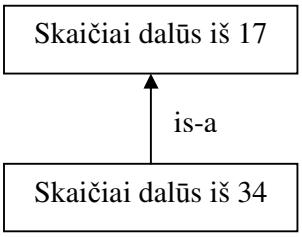
Galima šių pseudokodų sąryši apibrėžti taip:



Pav. 8.15 Tarp F3 ir F yra is–a ryšys

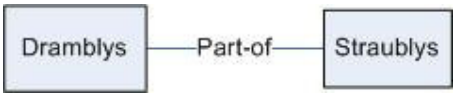
Dar vienas pavyzdys – dalūs skaičiai. Bendrąja prasme sąvoka „skaičiai dalūs iš 17“ neegzistuoja.

Žiūrime kaip į klases, kuriose yra objektai ir skirstome pagal atributus.



Pav. 8.16 is-a dalių skaičių ryšys

5. Sloto reikšmės tipas.



Sąvybės:
Cilindras: ilgis: 1,3 m.
Spindulys: 0,1 m.

Pav. 8.17 part –of ryšys

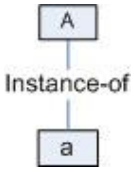
6. Aibės ir universalios aibės, kaip sąvokos sąryšis.

Pav. 8.18 Sąvokų ryšys

8.2. Ryšio instance-of interpretacijos

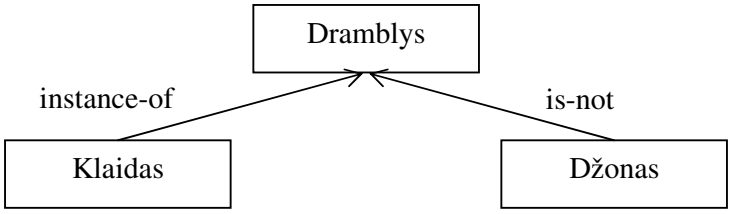
Ryšis instance-of gali turėti tokias interpretacijas.

1. Elemento ir aibės ryšis (angl. *set membership*) a∈ A.



Pav. 8.19 instance-of ryšys

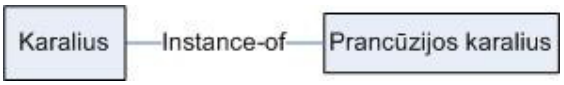
2. Predikatas.



Pav. 8.20 instance or ir is–not ryšiai

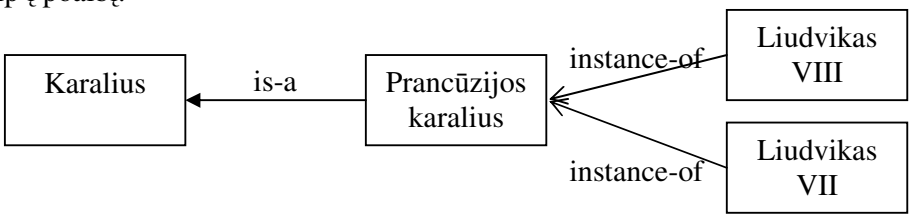
Dramblys(„Klaidas“) = true. Dramblys („Džonas“) = false.
Konstanta „Klaidas“ sąlygoja predikato reikšmę true, o Džonas – false.

3. Koncepcinis aprėpimas, apribojimas (angl. *conceptual containment*).
Pavyzdžiui, „Prancūzijos karalius yra karalius“.



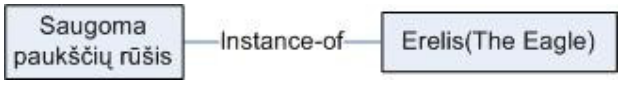
Pav. 8.21 instance–of ryšys

Tačiau prisimename, jo viskas priklauso nuo konteksto, taigi galim pažiūrėti į Prancūzijos karalių kaip į poaibį:



Pav. 8.22 Ryšių is-a ir instance-of priklausomybė nuokonteksto.

4. Abstrakcija (abstraction).
Bendrą sąvoką detalizuojanti sąvoka:



Pav. 8.23 savokų ryšys instance-of

9. Klasifikavimo vaizdavimas

Pradėsime nuo pavyzdžio, demonstruojančio klasifikavimą specifinėje dalykinėje srityje – mokesčių teisėje. Mokesčių administravimo įstatymo [MAĮ] 13 straipsnis nustato mokesčių sąrašą Lietuvoje. Siekdami iliustruoti šią dalykinę sritį, sąrašą pateikiame nesutrumpintą.

13 straipsnis. Mokesčiai

Pagal šį Įstatymą administruojami mokesčiai:

1. pridėtinės vertės mokestis;
2. akcizas;
3. gyventojų pajamų mokestis;
4. nekilnojamojo turto mokestis;
5. žemės mokestis;
6. mokestis už valstybinius gamtos išteklius;
7. naftos ir dujų išteklių mokestis;
8. mokestis už aplinkos teršimą;
9. konsulinis mokestis;
10. žyminis mokestis;
11. (neteko galios nuo 2005 m. liepos 1 d.);
12. paveldimo turto mokestis;
13. privalomojo sveikatos draudimo įmokos;
14. įmokos į Garantinį fondą;
15. valstybės rinkliava;
16. loterijų ir azartinių lošimų mokestis;
17. mokesčiai už pramoninės nuosavybės objektų registravimą;
18. pelno mokestis;
19. valstybinio socialinio draudimo įmokos;
20. pertekliaus mokestis cukraus sektoriuje;
21. gamybos mokestis cukraus sektoriuje;
22. (neteko galios nuo 2007 m. balandžio 26 d.);
23. muitai;
24. atskaitymai nuo pajamų pagal Lietuvos Respublikos miškų įstatymą;
25. mokestis už valstybės turto naudojimą patikėjimo teise;
26. socialinis mokestis;
27. cukraus pramonės restruktūrizavimo laikinasis mokestis;
28. papildomos baltojo cukraus gamybos kvotos ir pridėtinės izogliukozės gamybos kvotos vienkartinio išsipirkimo mokestis.

Knygoje „Mokesčių teisė“ [Marcijonas, Sudavičius 2003, p. 18] rašoma: „Pagal mokesčių mokėtojų ypatybes skiriami **juridinių asmenų**, **fizinių asmenų (gyventojų) ir bendri mokesčiai**. Juridiniai asmenys moka pelno, fiziniai asmenys (gyventojai) – pajamų mokestį; bendriems mokesčiams priklauso žyminis mokestis, valstybės rinkliava, nuompinigiai (užmokestis) už valstybinę žemę bei valstybinio fondo vandens telkinius ir t. t.“.

Turime aibės mokesčiai suskaidymą. Suskaidymas susideda iš trijų aibių. Mokesčiai:

1. juridinių asmenų mokesčiai;
   mokestis.mokėtojo_tipas = {“juridinis asmuo”}. JA=1 & FA=0
2. fizinių asmenų mokesčiai;
   mokestis.mokėtojo_tipas = {“fizinis asmuo”}. JA=0 & FA=1
3. bendri mokesčiai.

mokestis.mokėtojo_tipas={“juridinis asmuo”, “fizinis asmuo”}. JA=1&FA=1.

Čia **type** mokestis.mokėtojo_tipas=**subset_of**{“juridinis asmuo”,“fizinis asmuo”}. 2 bitai: JA ir FA.

9.1. Algebrinis požiūris į klasifikavimą

Tegu turime aibę $A=\{a_1, a_2, ...\}$.

4.1 apibrėžimas. Aibės A suskaidymu (angl. *partition*) $A=\{A_1,A_2,\ldots,A_m\}$ yra vadinama aibės A poaibių aibė tokia, kad:

1. poaibiai nesikerta (angl. disjoint): $A_i \cap A_j=\varnothing$, kai $i \neq j$;
2. poaibių aibė yra išsemianti (angl. *exhaustive*): $\forall a \in A$; $\exists i \in \{1,2,...,m\}$ toks, kad $a \in A_i$. Kitais žodžiais, kiekvienam aibės A elementui a egzistuoja toks poaibis $A_i \in A$, kurio elementu yra a.

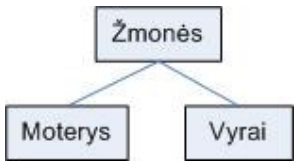
4.2 pavyzdys. Aibės žmonės suskaidymas {vyrai, moterys}. Skaidoma pagal atributą žmogus.lytis. Į suskaidymo elementą vyrai (kaip aibės žmonės poaibį) patenka tokie elementai žmogus $\in$ žmonės, kurių atributas lytis turi reikšmę =“vyras”, t.y.

žmogus.lytis=“vyras”.

Į suskaidymo elementą moterys patenka tie elementai, kurių

žmogus.lytis=“moteris”.

Kitaip tai galima pavaizduoti:



Pav. 9.1 Klasifikavimo pavyzdys: žmonės = vyrai $\cup$ moterys arba žmonės = {vyrai, moterys}.

Kaip jau buvo minėta tai yra skaidymas pagal atributą (kriterijų) lytis, t.y. žmogus.lytis = („vyras“ arba „moteris“). Galima sakyti, kad žmogus turi savybę lytis.

Aibė paprastai yra suskaidoma pagal kokį nors ekvivalentiškumo santykį.

4.3 apibrėžimas. Santykis ~ tarp aibės A elementų yra ekvivalentiškumo santykis, tada ir tik tada jis yra:

1. tranzityvus: iš x~y ir y~z seka x~z, kur $x \neq y$, $y \neq z$;
2. refleksyvus: x~x kiekvienam $x \in A$.

4.4 apibrėžimas. Aibės A suskaidymas pagal joje apibrėžtą ekvivalentiškumo santykį ~ vadinamas *faktoraibe* ir žymimas A/~.

4.5 pavyzdys. Apibrėžkime ekvivalentiškumo santykį natūraliųjų skaičių aibėje N, $A=N=\{0,1,2,3,4...\}$; x~y jeigu dalijant iš 3 jų liekanos yra lygios, t.y. (x mod 3) = (y mod 3).

Pagal šį santykį aibės A suskaidymas yra sudarytas iš trijų aibių: {0,3,6,9,...}, {1,4,7,10,...} ir {2,5,8,11,...}. T.y.:

A/~ = { {0,3,6,9,...}, {1,4,7,10,...}, {2,5,8,11,...} }

Pabrėžiame, kad aibės suskaidymo elementai savo ruožtu yra aibės, t.y. aibės A kaip alfabeto poaibiai.

9.2. Santykio sąvoka

Santykis (rus. *отношение*, angl. *relation*) yra griežta matematinė sąvoka.

4.6 apibrėžimas. *Binarinis santykis* R tarp aibės A elementų yra Dekarto sandaugos A×A poaibis, t.y. R⊂A×A

Binarinis santykis gali būti suprantamas kaip dvivietis predikatas r(x,y), kur r(x,y) = true, kai x ir y yra santykyje R, ir r(x,y) = false, kai x ir y nėra santykyje R.

Dar kitaip binarinį santykį galėtume apibrėžti, kaip lentelę turinčią du stulpelius, pvz.: lentele pavaizduokime 4 pavyzdyje pateikiamą santykį:

(x mod 3) = (y mod 3) = 0	(x mod 3) = (y mod 3) = 1	(x mod 3) = (y mod 3) = 2
0~0	1~1	2~2
0~3	1~4	2~5
0~6	1~7	2~8
0~9	1~10	2~11
0~12	1~13	2~14
...	...	...
3~0	4~1	5~2
3~3	4~4	5~5
3~6	4~7	5~8
3~9	4~10	5~11
3~12	4~13	5~14
...	...	...
6~0	7~1	8~2
6~3	7~4	8~5
6~6	7~7	8~8
6~9	7~10	8~11
6~12	7~13	8~14
...	...	...

Mūsų nagrinėjamame algebriniame pavyzdyje, santykis apibrėžiamas tarp sveikų skaičių, todėl šį santykį nusakanti lentelė yra begalinė.

Taip pat norėtume išskirti santykio ir sąryšio sąvokas. Todėl norėtume priminti, kad santykis – lentelė, Dekarto sandaugos poaibis. Pagal mūsų nagrinėjamą pavyzdį: {(0,0), (0,3), (0,6),..., (3,0), (3,3), (3,6),..., ..., (1,1), (1,4), (1,7),..., (4,1), (4,4), (4,7),..., ..., (2,2), (2,5), (2,8),..., (5,2), (5,5), (5,8),...}. O sąryšis realizuojamas santykio sąvokos pagalba.

Žodžiais santykį dar galėtume apibrėžti taip: su skaičiai yra santykyje, jeigu jie tenkina kažkokią salygą, pagal kuria yra sudaroma lentelė.

Bandykime įrodyti, kad mūsų nagrinėjamas santykis yra ekvivalentumo santykis. Įrodinėsime pagal apibrėžimą.

Pirmiausia įrodykime tranzityvumą:

x~y, kai (x mod 3) = (y mod 3), tą galėtume perrašyti šiek tiek kitaip: $x=3*k_1+r$ ir $y=3*k_2+r$. y~z, kai (y mod 3) = (z mod 3), analogiškai gautume: $y=3*k_2+r$ ir $z=3*k_3+r$. Kadangi liekanos lygios tai reiškia, kad galima rašyti (x mod 3) = (z mod 3), ką ir reikėjo įrodyti.

Aukščiau apibrėžtas santykis ~ yra **ekvivalentiškumo santykis**.

Dabar įrodykime refleksyvumą. Refleksyvumas yra akivaizdus, nes (x mod 3)=(x mod 3).

Kitaip ekvivalentumą galima apibrėžti kaip predikatą Equiv(x,y). Equiv(x,y) = true, kai yra tenkinama kažkokia sąlyga. Mūsų nagrinėjamu atveju x dalinant iš trijų ir y dalinant iš trijų gaunama ta pati liekana. Equiv(x,y) = false, kai ekvivalentumo sąlyga yra nepatenkinta. Mūsų nagrinėjamu atveju x dalinant iš trijų ir y dalinant iš trijų gaunamos skirtingos liekanos.

Paaiškinsime faktoraibės sąvoką šiame pavyzdyje.

Aibės A = N = {0, 1, 2, 3, 4,…}, suskaidymas pagal mūsų apibrėžtą santykį ~ yra:

$\mathtt{A} = A/\sim = \{A_1, A_2, A_3\}$, kur

$A_1 = \{0, 3, 6, 9,...\}$, $A_2 = \{1, 4, 7, 10,...\}$, $A_3 = \{2, 5, 8, 11,...\}$

A_1, A_2 ir A_3 yra faktoraibės $\mathtt{A}$ elementai, t.y. $A_1 \in \mathtt{A}$, $A_2 \in \mathtt{A}$, $A_3 \in \mathtt{A}$. Savo ruožtu A_1, A_2 ir A_3 yra natūraliųjų skaičių aibės poaibiai, t.y. $A_1 \subset A$, $A_2 \subset A$, $A_3 \subset A$. Bendru atveju $A/\sim=\{A_1,A_2,A_3,...A_N\}$, kur $A_i \in A/\sim$ ir $A_i \subset A$.

Konceptualaus modeliavimo prasme nagrinėjamą pavyzdį galima pavaizduoti Pav. 9.2.:

Pav. 9.2 Natūraliųjų skaičių aibės suskaidymas pagal ekvivalentiškumo santykį. Rezultatas yra faktoraibė.

Santykis gali būti ne tik dvivietis jis gali būti ir trivietis, ir daugiavietis.

4.7 pavyzdys

Santykis Skrydis(Kas_skrenda, Iš_kur, Į_kur, Kada) gali būti nustatomas lentele Pav. 9.3.:

Kas_skrenda	*Iš_kur*	*Į_kur*	*Kada*
Shankar	New York	New Delhi	2006-03-30
...	...	...	...

Pav. 9.3 Lentelė, kuri nustato santykį Skrydis(Kas_skrenda, Iš_kur, Į_kur, Kada).

4.8 teorema. Santykis tarp objektų x ir y yra ekvivalentiškumo santykis, jeigu šių objektų atributo atr reikšmės yra lygios, t. y. x.atr=y.atr.

4.9 pavyzdys. Paimkime spausdintuvų atributą spausdinimo_tipas. Tegu jo reikšmė yra viena iš aibės {“lazerinis”, “rašalinis”, “adatinis”, “kitoks”}. Apibrėžiame x~y, jeigu

x.spausdinimo_tipas = y.spausdinimo_tipas

Tada organizacijoje esančių spausdintuvų aibę $\{sp_1, sp_2, \ldots, sp_n\}$ suskaidome į poaibius lazeriniai-spausdintuvai, rašaliniai_spausdintuvai, adatiniai_spausdintuvai ir kitokie_spausdintuvai.

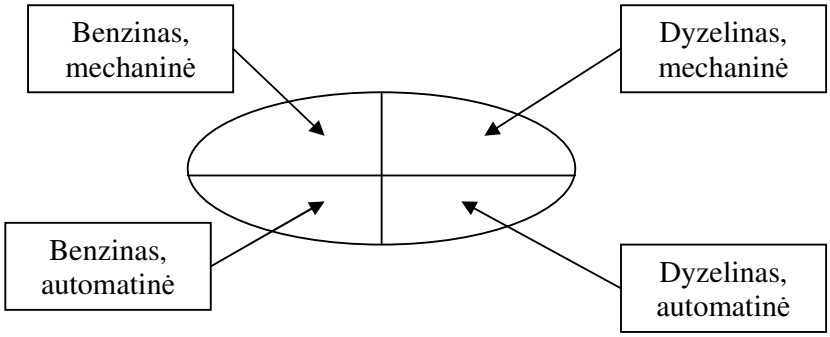
4.10 pavyzdys. Automobilius suskaidome pagal atributą spalva. Kiek spalvų tiek ir poaibių: raudoni, geltoni, žali ir t.t.

Konceptualizuojant dalykinę sritį kyla suskaidymo į atspalvius klausimas. Kokie juodos, baltos ir kitų spalvų atspalviai. Pvz., „asfalto“ spalvos arba „tamsiai pilka“?

Gyvenimiškesnių suskaidymų pavyzdžiai yra, pvz., pagal tokius atributus:
kuras = {“benzinas”, “dyzelinas”}
greičių_dėžė = {“mechaninė”, “automatinė”}

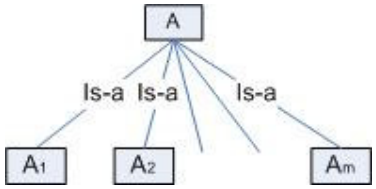
Ekvivalentiškumo santykį gauname kaip konjunkciją tatrp keleto atributų, pvz.:
(x.kuras = y.kuras) & (x.greičių_dėžė = y.greičių_dėžė).

Pavyzdžiui, pagal tokius santykius automobiliai klasifikuojami, tinklapiuose, skirtuose naudotų automobilių pardavimui. Atitinkamas suskaidymas yra pavaizduotas Pav. 9.4.

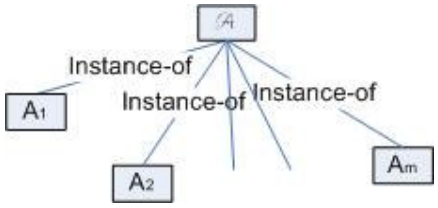


Pav. 9.4 Automobilių aibės suskaidymas pagal ekvivalentiškumo santykį x~y ⇔ (x.kuras = y.kuras) & (x.greičių_dėžė = y.greičių_dėžė).

Suskaidymą $A=\{A_1,A_2,\ldots,A_m\}$ kur $A_i \subset A$ galime pavaizduoti Pav. 9.5.:



Pav. 9.5 Suskaidymo $A=\{A_1,A_2,\ldots,A_m\}$ kur $A_i \subset A$ pavaizdavimas. Viršūnėje aibė A.



Pav. 9.6 Kitoks suskaidymo $A=\{A_1,A_2,\ldots,A_m\}$, kur $A_i \subset A$ ir tokiu būdu $A_i \in A$ pavaizdavimas. Viršūnėje yra suskaidymas A.

9.3. Tiesinės erdvės faktorizavimas pagal tiesinį poerdvį

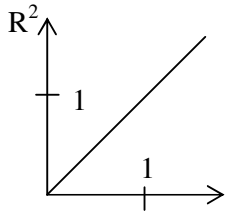
Abstrakčiojoje algebroje kaip matematikos šakoje yra žinomas tiesinės erdvės suskaidymas į faktoraibę pagal tiesinį poerdvį. Toks skaidymas vadinamas *faktorizavimu*.

Pateikime pavyzdį nagrinėdami dvimatę tiesinę erdvę R^2 ir paimdami jos tiesinį poerdvį I. Jeigu $x \in I$, ir $y \in I$, tai $x+y \in I$, ir $t*x \in I$, kur t yra realus skaičius. Tiesinis perdvis gali būti imamas tokio pavidalo: $I = \{t*v,\ t \in R\}$, kur t – poaibis dvimatėje erdvėje, o v – bazinis vektorius.

Analogiškai gaunamas dvimatį poaibį trimatėje erdvėje virš dviejų bazinių vektorių v_1 ir v_2, t.y. plokštuma:

$I = \{t_1*v_1 + t_2* v_2$, kur t_1 ir t_2 yra realūs skaičiai, t.y. $t_1, t_2 \in R\}$

Imkime pavyzdį iš dvimatės erdvės:



Pav. 9.7 Tiesinis poaibis $I = \{t*(1,1),\ t \in R\}$ dvimatėje Euklidinėje erdvėje R^2

Dvimatėje Euklidinėje erdvėje R^2 tiesinį poerdvį I sudaro, pavyzdžiui, 45 laipsnių kampu einanti tiesė, t.y. aibė vektorių, turinčių pavidalą (t,t), kur t yra realus skaičius. Įrodysime, kad tokia aibė yra tiesinis poerdvis.

Tegu $I = \{t*(1,1),\ t \in R\}$. Jeigu $x \in I$ ir $y \in I$, tai $x = t_1*(1,1) = (t_1, t_1)$ ir $y = t_2*(1,1) = (t_2,t_2)$. Tada:

1. $x+y = (t_1, t_1) + (t_2, t_2) = (t_1+t_2, t1+t_2) = (t_1+t_2)*(1,1) \in I$.
2. $k*x = k*(t_1, t_1) = (k*t_1, k*t_1) \in I$.

Įrodymas, kad I yra tiesinis poerdvis yra baigtas.

Tiesinėje erdvėje ekvivalentiškumo santykis x~y apibrėžiamas šitaip.

4.11 apibrėžimas.Vektoriai x ir y yra ekvivalentiški tada ir tik tada kai jų skirtumas priklauso tiesiniam poerdviui:

$x \sim y \Leftrightarrow (x-y) \in I$

Įrodymas, kad aukščiau apibrėžtas santykis yra ekvivalentiškumo santykis, t.y. kad tenkina tranzityvumo ir refleksyvumo aksiomas.

Tranzytivumas

Žinome:

$x = (x_1, x_2)$

$y = (y_1, y_2)$

$z = (z_1, z_2)$

$x \sim y$ ir $x - y = (t_1, t_1) \Rightarrow x = (y_1 + t_1, y_2 + t_1)$

Taip pat žinome:

$y \sim z$ ir $y - z = (t_2, t_2) \Rightarrow y = (z_1 + t_2, z_2 + t_2)$

Gauname:

$x = (y_1 + t_1, y_2 + t_1) \Rightarrow x = (z_1 + t_2 + t_1, z_2 + t_2 + t_1) \Rightarrow x = z + (t_2 + t_1, t_2 + t_1)$
$x - z = (t_2 + t_1, t_2 + t_1) \in I$

Vadinasi, santykis $x \sim y \Leftrightarrow (x-y) \in I$ yra tranzytivus.

I – tiesinis poaibis, kuris pasižymi tokiomis savybėmis:

1. jei $x, y \in I$, tai $x + y \in I$
2. jei $\alpha, x \in I$, kur $\alpha \in R$, tai $\alpha x \in I$

Tuo atveju, kai $\alpha = 0$, tai $(0,0) \in I$

Galima apibrėžti teigiamus, neigiamus ir nulinius elementus:

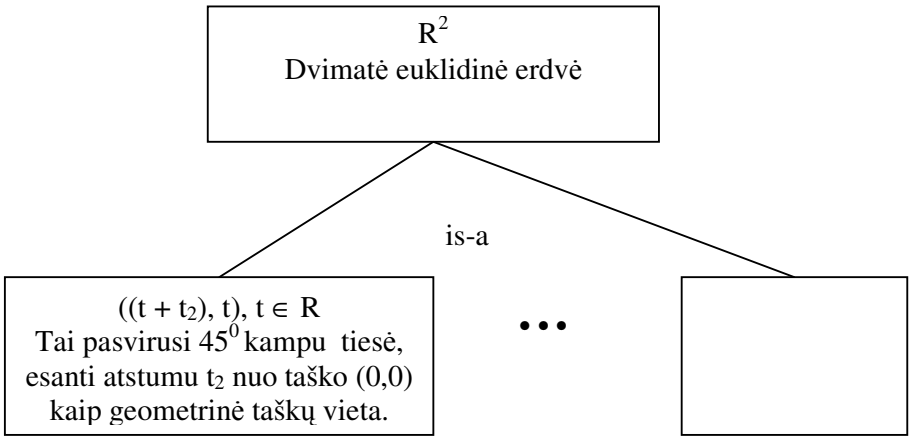
$(t_2' + I) \oplus (t_2'' + I) = t_2' + t_2'' + I$

Nulinio elemento pavyzdys:

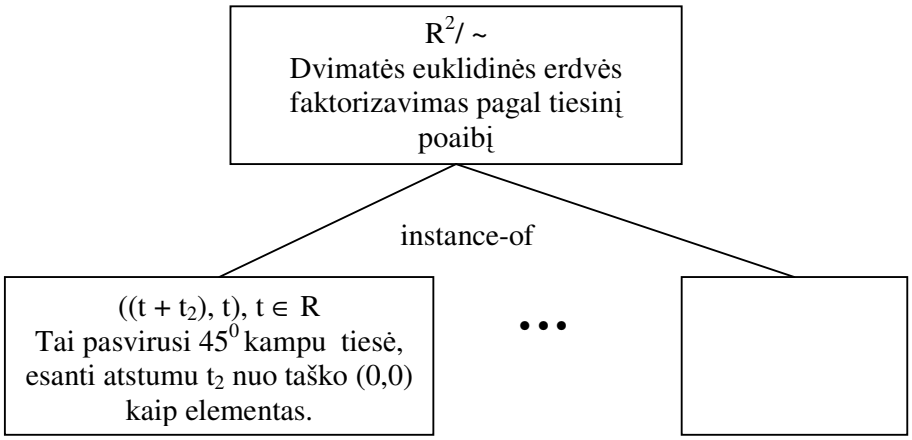
$(t_2 + I) \oplus (0 + I) = t_2 + I$

Izomorfiškas Euklidinių erdvių sąryšis:

$R^n / \sim \approx R^{n-1}$

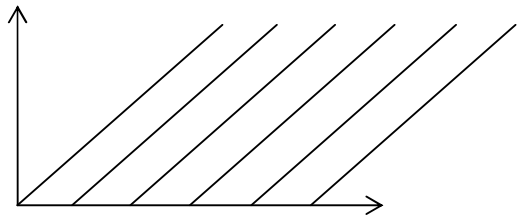


Pav. 9.8 Dvimatės euklidinės erdvės skaidymas į geometrines taškų vietas pagal t_2



Pav. 9.9 Dvimatės euklidinės erdvės skaidymas į elementus pagal t_2

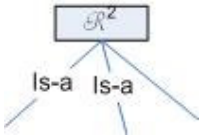
Suskaidykime R^2 pagal tiesinį poaibį `I`. Į dvimatę plokštumą žiūrime kaip į 45 laipsnių kampu pasvirusių tiesių sąjungą:



Pav. 9.10 Dvimatėje Euklidinėje erdvėje R2, 45 laipsnių kampu pasvirusių tiesių sąjunga

$R^2 = \bigcup_{t_2 \in -\infty \ldots +\infty} \{(t + t_2, t) \in R\}$, kur Y žymi aibių sąjungą.

Tai galime pavaizduoti tokiu semantiniu tinklu:



$\{(t + t_2, t), t \in R\}$

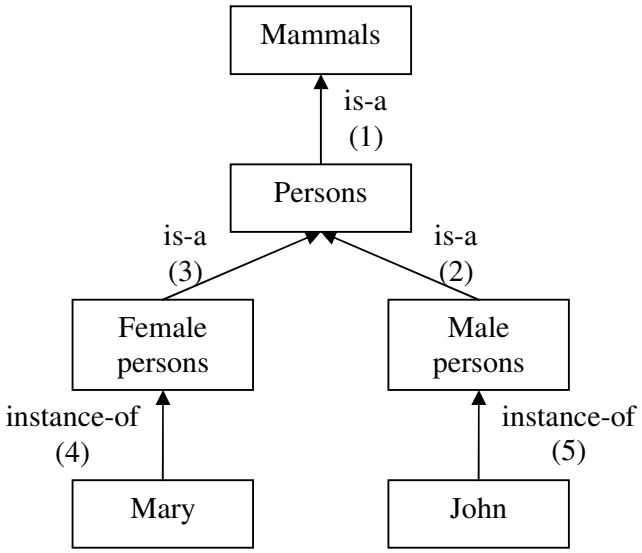
Pav. 9.11 Dvimatėje Euklidinėje erdvėje R^2 is-a ryšys su (t+t2;t) $t \in R$

Analogiškai $R^2/\sim = \{\{(t + t_2, t), t \in R\}\ t_2 \in (-\infty, +\infty)\}$. Tačiau semantiniame tinkle vietoje is-a turėsime instance-of.

10. Semantiniai tinklai pagal Russell ir Norvig

Apie semantinius tinklus yra rašoma ir [Russell, Norvig 2003, p. 350-352] 10 skyriuje „Žinių vaizdavimas“ („Knowledge representation“).

Semantinio tinklo sąvoka pasiūlė 1909 metais Pierce. Nuo to laiko vyksta debatai, ar semantiniai tinklai yra tam tikra logikos forma, ar kažkas daugiau. Mūsų nuomone, semantiniai tinklai nėra vien tik tai logikos forma. Jeigu tai būtų tik logikos forma, tai būtų labai griežta semantika. Semantinių tinklų jėga yra tame, kad nėra formalios semantikos. Kiekvienas pavaizdavimui gali priduoti tokią prasmę, kokia jis nori. Kyla rizika, kad skirtingi žmonės supras skirtingai tą patį pavaizdavimą, bet tokia ir yra kaina. Kartais netgi sąmoningai daroma, kad netgi skirtingi žmonės suprastų skirtingai, pavyzdžiui teisės srityje tą patį dalyką prokuroras vertina vienaip, advokatas kitaip, teisėjas gali ar netgi privalo turėti savo nuomonę. Jei grafui suteiktume tik vieną vienintelę reikšmę – nukirstume šaką ant kurios sėdime. Filosofiškai žiūrint, į kokį nors teigiamą dalyką, iš kitos puses, galima žiūrėti kaip į neigiamą. Pavyzdžiui, jeigu turime labai daug benzino naudojantį automobilį, tai galime žiūrėti ir teigiamai, nes tam tikromis aplinkybėmis, tai gali suteikti finansinę naudą. Arba tiesiog turi tiesiog kitų teigiamų savybių: yra stabilesnis, erdvesnis.



Pav. 10.1 Semantinis tinklas iš [Russell, Norvig 2003, p. 350-352].

(1) $\forall x$ Person(x) $\Rightarrow$ Mammal (x);
(2) $\forall x$ Female_person(x) $\Rightarrow$ Person (x);
(3) $\forall x$ Male_person(x) $\Rightarrow$ Person (x);
(4) Female_person (Mary). Tai faktas. Todėl šis predikatas lygus true.
(5) Male_person(John). Tai faktas. Todėl šis predikatas lygus true.

Šias taisykles galima užrašyti ir ne predikatų, o aibių ir elementų forma (kaip padaryta Russell, Norvig knygoje). Pavyzdžiui:

$\forall x$ ($x \in$ Persons $\Rightarrow x \in$ Mammals).

Taigi grafe stačiakampiais mes žymėjome kategorijos vardą (sąvokas), o Russell, Norvig knygoje, pažymėta apskritimais, pasakant jog tai aibės (intensionalai) t.y. suteikiant šiokią tokią semantiką.

Pavyzdžiui, jeigu turime lentelė iš dviejų laukų, tai visi potencialūs (visi įmanomi) įrašai bus **intensionalas** (kaip Dekarto sandauga), o – visi realiai duomenų bazėje esantys įrašai – **ekstencionalas** (griežtas poaibis).

Jei turime galvoje esybę, tai naudojame vienaskaitą ir sakome žinduolis, moteris, vyras, žmogus. Vienaskaitą naudojame ir predikato varde, pvz., Person(x). Jeigu naudojame daugiskaitą, tai intuityviai suvokiame, kad kalbame apie klasę, pvz., žinduoliai.

Į šį rafinį atvaizdavimą galima įdėti daugiau prasmės (semantikos) negu, kad aprašysime vien tik logikos formulėmis. Pavyzdžiui galima nuspalvinti skirtingomis spalvomis arba paryškinti tam tikrus elementus, pakeisti dydį.

10.1. Paveldėjimas

Savybę, kad žmogus turi dvi kojas, yra pavaizduota Pav. 10.2. Čia matome, kad kategorija skirta ne tik, kad turėti aibę (matematikoje aibės elementai neturi jokių savybių kaip kvapas skonis ir pan.). Čia aibės elementams, **kaip objektams**, suteikiamos savybės.

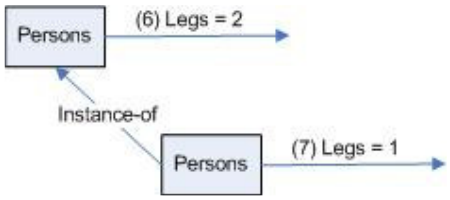
Pav. 10.2 Person turi dvi kojas

5.2

Mes galime išvesti, kad ir individualios sąvokos turės tą pačią sąvybę. Pavyzdžiui, individuali sąvoka John turės sąvybę „kojos yra dvi“, todėl, kad John to atributo neturi, tuomet matome, kad ir Male persons to atributo neturi, kylam aukštyn ir ties Persons – randame.

(6) $\forall x$ Person(x) $\Rightarrow$ Legs(x,2)

Tačiau Pav. 10.3 mes norime pavaizduoti, kad John turi vieną koją:



Pav. 10.3 John turi vieną koją

Turime predikatą:

(7) Legs(John,1)

Toks nemonotoninis išvedimas semantiniame tinkle arba žinių bazėje gali sukelti sunkumų, nes gali būti išvesti prieštaringi teiginiai.

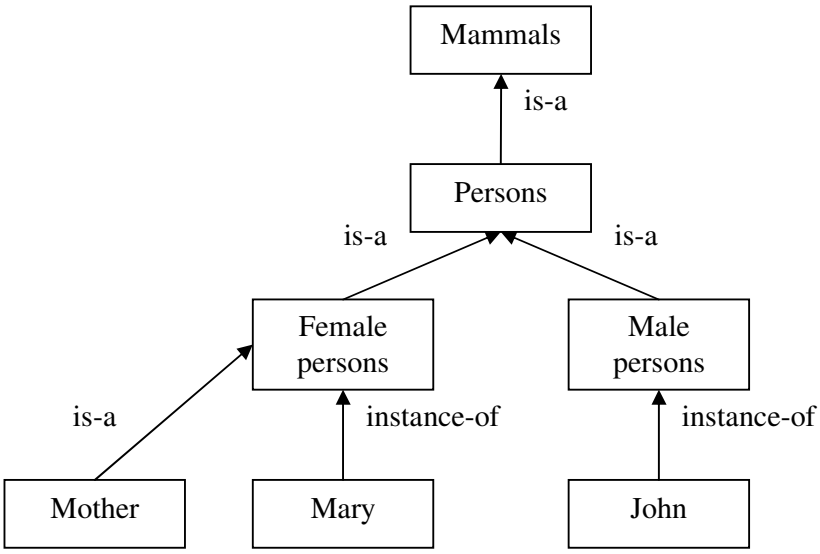
Aišku (6) galime užrašyti ir taip:

$\forall x$ (x $\in$ Persons & x $\neq$ John $\Rightarrow$ Legs(x,2)

Tačiau tokio tipo kalba būtų labai sudėtinga dėl išimčių. Tokia kalba būtų nepatogi duomenų bazėjei.

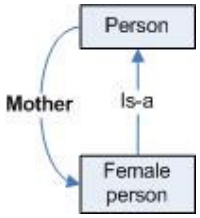
10.2. Santykiai ir ryšiai

Ar būtų galima pasaulį modeliuoti taip:



Pav. 10.4 Semantinis tinklas

Deja, čia yra iš principo neteisingas modeliavimas. Čia iš principo yra ne poaibis, kaip gali atrodyti iš pirmo žvilgsnio, o santykis tarp asmens ir jo motinos, tai yra dviejų konkrečių asmenų.



Pav. 10.5 Santykis tarp asmens ir jo motinos

Santykį mes suprantame kaip lentelę.

Taigi žodžiu mama yra žymimas ryšys. Tuo pačiu žodžiu žymime santykį mama, kai kalbame apie Dekarto sandaugą ir ją atitinkančią lentelę.

Asmuo	Jo mama
John	Catherine
...	...

Pav. 10.6 Santykis tarp asmens ir jo motinos pavaizduotas Dekarto sandauga.

Pavaizduosime, kad Catherine yra John mama.

(8) Female_person(Catherine)
(9) Mother(John, Catherine)

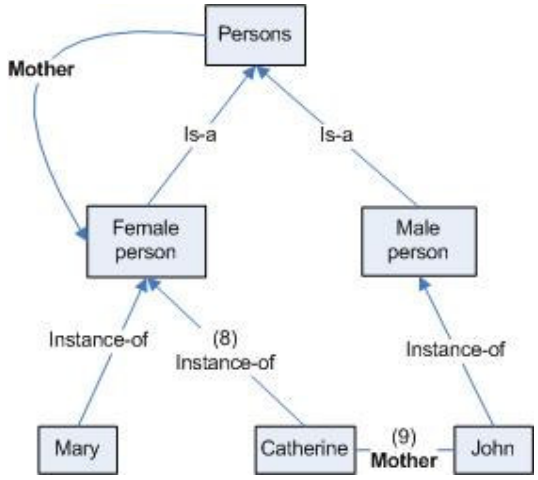
Dviejų laukų lentelės įrašą lengvai galima atvaizduoti dviviečiu predikatu.

Asmuo	Jo mama
John	Catherine

John	Ruth

Pav. 10.7 Dviejų laukų lentelės įrašas atvaizduotas dviviečiu predikatu.

Problema iškiltų, jeigu objektui John būtų priskirtos dvi mamos. Pavyzdžiui: mūsų santykis būtų nusakomas tokia lentele (žr.Pav. 10.5):



Pav. 10.8 Santykis tarp asmens ir jo šeimos narių.

Primename, kad santykiai gali būti 1:1, 1:N ir N:M.

Santykio kryptį pasirinkome remdamiesi tuo, kuris lentelės laukas yra raktas, t.y. pagal kurį lauką surūšiuota.

Šį santykį suprantame taip: egzistuoja predikatas. O kurios poros priklauso šiam predikatui, reikia išvardinti (santykio ekstencionalas).

Peršasi mintis, jog santykio intensionalas yra Dekarto sandauga. Persons X Female persons. Tačiau, kaip jau pastebėjome, čia, kaip ir grafe, neatsispindi tai, jog asmuo gali turėti ne daugiau negu vieną motiną.

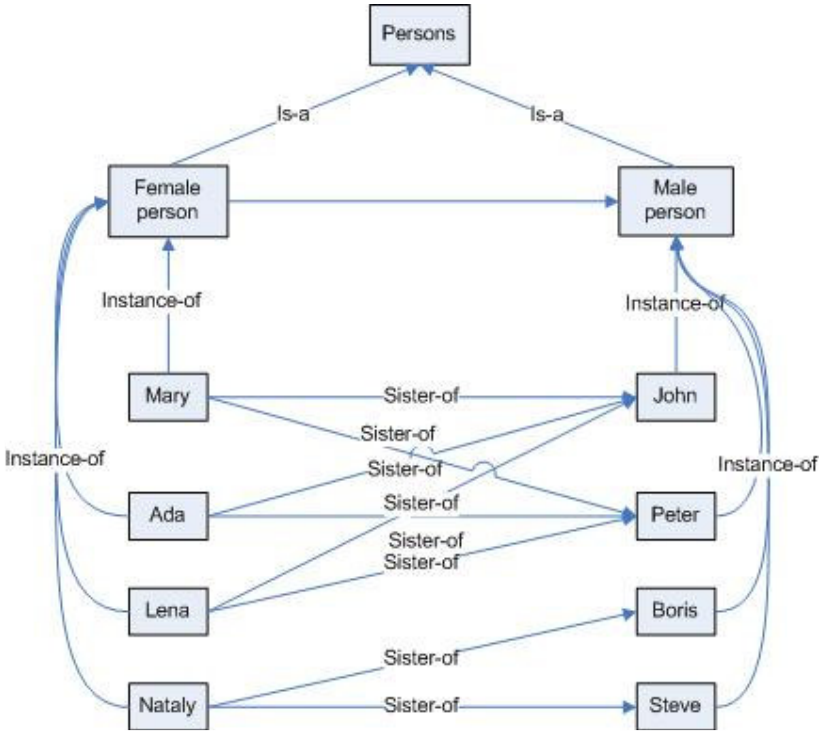
(9‘)$\forall x \in$ Persons $\Rightarrow$($\forall$y Has mother (x,y) $\Rightarrow$ y $\in$ Female_persons).

(9’’) Asmuo turi vieną mamą arba nė vienos.

10.3. Atvirkštinis santykis

Įveskime dar vieną santykį, būtent, Sister-of, tarp kategorijų Female_person ir Male_person. Priedo papildykime semantinį tinklą faktais, kad turime tris seseris Mary, Ada ir Lena bei jų du brolius John ir Peter. Taip pat vieną moterį Nataly bei du jos brolius Boris ir Steve.

(10) Female_person(Ada)
(11) Female_person(Lena)
(12) Female_person(Nataly)
(13) Male_person(Peter)
(14) Male_person(Boris)
(15) Male_person(Steve)
(16) Sister-of(Mary, John)
(17) Sister-of(Mary, Peter)
(18) Sister-of(Ada, John)
(19) Sister-of(Ada, Peter)
(20) Sister-of(Lena, John)
(21) Sister-of(Lena, Peter)
(22) Sister-of(Nataly, Boris)
(23) Sister-of(Nataly, Steve)



Pav. 10.9 Trys seserys Mary, Ada ir Lena turi du brolius John ir Peter. Nataly turi du brolius Boris ir Steve.

Santykis Sister-of apibrėžtas tarp kategorijų Moteris (Female_person) ir Vyras (Male_person). Tokiu būdu šiame semantiniame tinkle nėra vaizduojama situacija, kai dvi moterys yra seserys.

Sister-of santykis gali būti nustatytas Pav. 10.10.

Female_person	*Male_person*
Mary	John
Mary	Peter
Ada	John
Ada	Peter
Lena	John

Lena	Peter
Nataly	Boris
Nataly	Steve

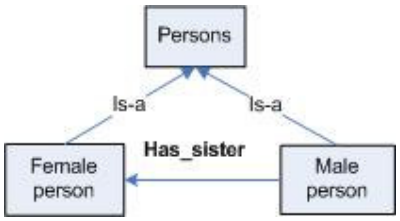
Pav. 10.10 Trys seserys Mary, Ada ir Lena turi du brolius John ir Peter. Nataly turi du brolius Boris ir Steve.

Savo žinių bazę mes įsivaizduosime, kaip nemonotoninę, nes galima išvesti prieštaringus teiginius. Todėl, kai atsiranda išimtys, pvz., Legs(John,1), susitarsime, kad speciali taisyklė nugali bendrąją taisyklę. Tačiau nuo tokio tipo susitarimų nukenčia išvedimo efektyvumas.

Susitarimas, kad speciali taisyklė nugali bendrąją taisyklę, taip pat yra taisyklė. Ją vadinsime metataisykle.

Toliau įvedame santykį Has_sister tarp sąvokų Male_person ir Female_person:

(24) ∀female ∀male Has_sister(male, female) ⇔ Sister-of(female, male)



Pav. 10.11 Santyks Has_sister tarp sąvokų Male_person ir Female_person

Apibrėžimas (neformalus):

Reifikavimas – tai žiūrėjimas, kaip į objektą.

Apibrėžimas. Reifikavimas – tai predikato arba funkcijos pavertimas algoritminės kalbos objektu. [Russell, Norvig 2003, p 230]

Tokiu būdu charakterizavimas predikatu(-ais) yra paverčiamas sąvoka. O sąvokos įvedimas yra būdingas žmogiškajam intelektui. Tokiu būdu predikatas gali būti pavaizduotas ryšiu (kaip sąvoka).

Pateikime duomenų bazei užklausą:

Sister-of(?, John)

Tai užklausa į Prolog panašia kalba. Sistemos prašoma rasti visas individualias sąvokas, kurios yra santykyje Sister-of su John.

Šios užklausos sudėtingumas yra tiesinis pagal lentelės Sister-of ilgį N. Perrenkame visus lentelės įrašus iš eilės ir ieškome tokių įrašų, kurių stulpelio Male-person reikšmė yra John. Randame tris tokius įrašus.

Ar galimas geresnis sudėtingumas?

Kitas susijęs klausimas, koks yra užklausos Has_sister(John, ?) sudėtingumas? Atsakymas: tiesinis pagal lentelės Sister-of ilgį N.

Įrodymas. Pagal (24) turime:

Has_sister (John, ?) ⇔ Sister-of (?, John)

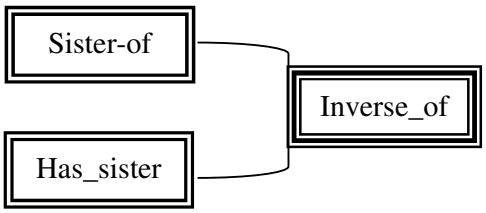
Įrodymo pabaiga.

Tuo tarpu, pavyzdžiui, užklausos Sister-of (Ada, ?) sudėtingumas yra logaritminis, t.y O(ln(N)). Tai yra todėl, kad predikatas gali būti yra surūšiuotas pagal pirmąjį parametrą.

Kaip gauti Has_sister (John, ?) logaritminį sudėtingumą?

Tam būtina sistemą iš anksto pateikti euristinę informaciją apie dalykinę sritį, kad santykis Has-sister yra atvirkštinis santykiui Sister-of:

(25) Inverse_of(Has_sister, Sister-of).



Pav. 10.12 Pasakymas Inverse-of(Has_sister, Sister-of), kad santykis Has_sister yra atvirkštinis santykiui Sister-of įgalina užklausos Has_sister(John, ?) logaritminį, o ne tiesinį sudėtingumą.

Tokia euristinė informacija pagal savo esmę yra *ryšys* (o ne santykis) tarp santykių Has_sister ir Sister-of kaip sąvokų.

Šis mūsų įterptas predikatas Inverse_of pagal savo esmę papildo išvedimo programą Pav. 10.10 atvirkštine lentele, kuri toliau pavaizduota kaip Pav. 10.13.

Male_person	Female_person
John	Mary
John	Ada
John	Lena
Peter	Mary
Peter	Ada
Peter	Lena
Boris	Nataly
Steve	Nataly

Pav. 10.13 Santykis Has_sister lentelė

Kaip matome, šioje lentelėje yra atlikta daugiau, negu ji tik apversta. Nauju raktu čia tapo Male_person. Ši apvertimo procedūra (25) yra brangi. Tačiau ją atlikę užklausos Has_sister (John, ?) sudėtingumą gauname $O(\log_2(N))$. Čia galima rašyti ir O(ln(N)), nes jie skiriasi tik konstanta daugiklyje: $\log_2(N) = C*\ln(N)$.

Teiginys apie atvirkštinį santykį yra efektyviai išvedamas jeigu:

1. Ryšys yra reifikuotas, t.y. paverstas objektu, kaip (25).
2. Ryšiui priskirta procedūra, t.y. atliktas procedūrinis priskyrimas.

Jeigu parašysime procedūrą, kuri realizuoją šį apvertimą, tai turėsime galimybę deklaratyviai nusakytį šio tipo santykius (inverse_of pagalba).

5.11 apibrėžimas. *Procedūrinis priskyrimas*, tai procedūra priskirta ryšiui ir skirta atsakymui į užklausą arba teiginio išvedimui, kuris yra efektyvesnis negu bendru loginio išvedimo algoritmo atveju.

Procedūrinis priskyrimas ir yra bruožas atskiriantis semantinį tinklą nuo logikos (čia turime galvoje teiginių logiką arba pirmos eilės predikatų logiką).

Reifikavimas –yra deklaravus vaizdavimo būdas, o procedūrinis prisegima tai iš procedūrinio vaizdavimo srities (procedūrinis vaizdavimas šiuo atveju net nevadinamas vaizdavimu).

Nagrinėtas santykis ∀female ∀male Has_sister(male, female) ⇔ Sister-of(female, male) yra labai paprastas, aprašomas viena eilute. Nėra kažkokių ypatingų sąlygų, kaip, pavyzdžiui, buvo su motinos santykiu ir ryšys („tada ir tik tada“ ⇔) yra labai stiprus. Jis reiškia, jog aibės sutampa t.y. has sister sutampa su sister of kaip aibės.

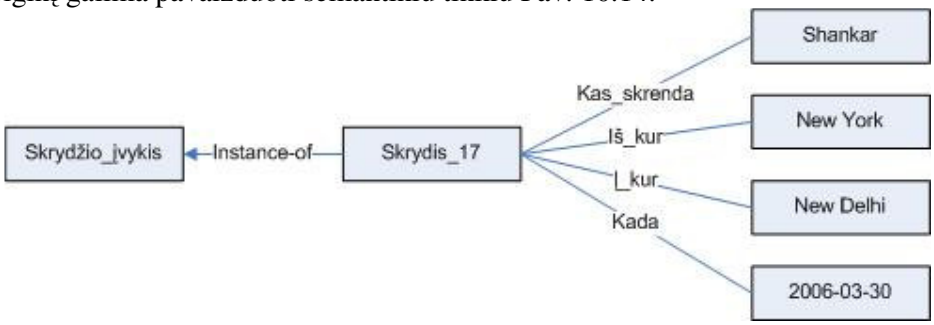
Pateiksime dar vieną pavyzdį, kad predikatą galime traktuoti kaip santykį. Taip pat primename, kad predikatai būna ne tik binariniai, jie būna dvinariai, trinariai ir t.t.

Prisiminkime 9.2 skyrelio Pav. 9.3 pavyzdį. Įsivaizduokime, kad turime predikatą

Skrydis(Kas_skrenda, Iš_kur, Į_kur, Kada) ir žinių bazėje daug teiginių jį attitinkančių, tokie teiginiai tai ir bus lyg santykis. Pavyzdžiui teiginys

Skrydis(Shankar, New York, New Delhi, 2006-03-30)

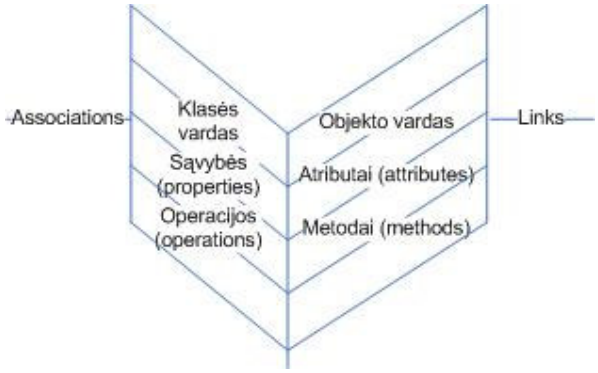
Šį teiginį galima pavaizduoti semantiniu tinklu Pav. 10.14.



Pav. 10.14 Predikato Skrydis(Shankar, New York, New Delhi, 2006-03-30) pavaizdavimas semantiniu tinklu.

Primename, kad mūsų santykį realizuojančioje lentelėje turėsime tiek stulpelių, kiek atributų santykyje.

Objektinio programavimo (OP) terminais dualistinis požiūris į klases ir objektus yra pavaizduotas Pav. 10.15. Toks požiūris į klasę-objektą yra vadinamas Clabject ir yra suprantamas kaip metamodelis.

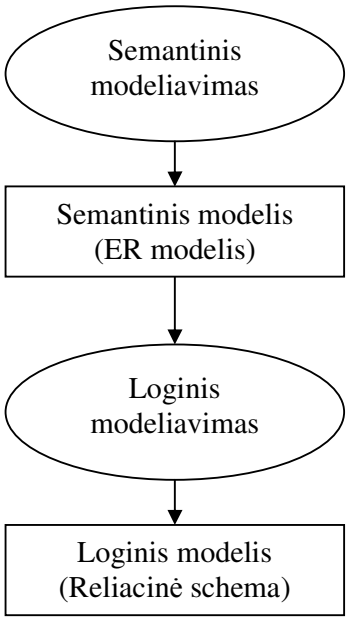


Pav. 10.15 Clabject (angl. class-object) modelis. Tai objektinio programavimo požiūris į klases ir objektus kaip metamodelis.

11. Koncepciniai modeliai duomenų bazėse

Šis skyrius dėstomas remiantis Romo Barono 2002 m. išleistos mokymo priemonės ir atnaujinto vadovėlio [Baronas 2005] 4 skyriumi „Semantinis modeliavimas“. Pastarajame yra aptariamas duomenų bazių (DB) esybių-ryšių modelis (toliau – ER modelis, angl. *entity-relationship model*), kurį 1976 m. pasiūlė Piteris Pin-Šan Čenas (Peter Pin-Shan Chen).

Po to kai turime ER modelį, vyksta loginis modeliavimas ir pagaminama reliacinė schema.



Pav. 11.1Koncepcinis modelis

Semantiniai tinklai yra siejami su analizės etapu, kai mes nieko nežinome, turime naują dalykinę sritį.

DB koncepcinė schema yra deklaratyvus žinių pavaizdavimas, t.y. vienas iš žinių pavaizdavimo būdų yra lentelių reliacinės schemos – lentelės ir visų jų stulpelių pavadinimai, su pažymėtu (išskirtu, pabrauktu) pirminiu raktu.

Žemiau imamas duomenų bazės pavyzdys iš [Baronas 2005] 2 skyriaus.

Dumenų bazę *Darbai* trys lentelės: *Vykdytojai*, *Projektai*, *Vykdymas*. Tarkime, kad šios lentelės užpildytos tokiais duomenimis:

Vykdytojai

Nr	Pavardė	Kvalifikacija	Kategorija	Išsilavinimas
1	Jonaitis	Informatikas	2	VU
2	Petraitis	Statistika	3	VU
3	Gražulytė	Inžinierius	1	NULL
4	Onaitytė	Vadybininkas	5	VDU
5	Antanaitis	Informatikas	3	VU

Pav. 11.2 Lentelė Vykdytojai.

Projektai

Nr	Pavadinimas	Svarba	Pradžia	Trukmė

1	Studentų apskaita	Maža	2001.01.01	12
2	Buhalterinė apskaita	Vidutinė	2001.03.01	10
3	WWW svetainė	Didelė	2001.06.01	2

Pav. 11.3 Lentelė Projektai

Vykdymas

Projektas	Vykdytojas	Statusas	Trukmė
1	1	Programuotojas	30
1	2	Dokumentuotojas	100
1	3	Testuotojas	100
1	4	Vadovas	100
2	1	Programuotojas	300
2	2	Analitikas	250
2	4	Vadovas	100
3	1	Programuotojas	250
3	2	Vadovas	400
3	3	Dizaineris	150

Pav. 11.4 Lentelė Vykdymas

Lentelės eilutes galime traktuoti kaip kortežus. O lentelę galime traktuoti kaip Dekarto sandaugos poaibį:

$Vykdytojai \subset integer \times string \times string \times integer \times string$

Čia × žymi aibių Dekarto sandaugą. Lentelėje *Vykdytojai* stulpelis *Nr* yra integer tipo, *Pavardė* – string tipo, *Kvalifikacija* – string tipo, *Kategorija* – integer tipo, *Išsilavinimas* – string tipo. Stulpelių pavadinimai, pvz., *Pavardė*, *Kvalifikacija*, *Kategorija*, *Išsilavinimas* yra vadinami **atributais**. Dekarto sandaugos elementas gali būti suprantamas ir vaizduojamas kaip kortežas $(e_1, e_2, e_3, e_4, e_5)$, kur e_i yra kokia nors *i*-tojo atributo reikšmė, pavyzdžiui:

$(1, Jonaitis, Informatikas, 2, VU) \in Vykdytojai$

Analogiškai:

$Vykdytojai \subset Nr \times Pavardė \times Kvalifikacija \times Kategorija \times Išsilavinimas$

Čia $Nr \subset integer$, $Kvalifikacija \subset string$, $Pavardė \subset string$, $Kategorija \subset integer$, $Išsilavinimas \subset string$.

Parašysime užklausą SQL kalba ir išrinksime iš lentelės *Vykdytojai* tas atributų *Pavardė* ir *Kategorija* reikšmes, kurios priklauso tiems vektoriams, kurių atributo *Kvalifikacija* reikšmė yra „Informatikas“:

SELECT *Pavardė, Kategorija* FROM *Vykdytojai* WHERE *Kvalifikacija* = *„Informatikas“*

Duomenų bazės *Darbai* **reliacinė schema**:

Vykdytojai (Nr, Pavardė, Kvalifikacija, Kategorija, Išsilavinimas)

Projektai (Nr, Pavadinimas, Svarba, Pradžia, Trukmė)

Vykdymas (Projektas, Vykdytojas, Statusas, Valandos)

Išoriniai raktai:

Projektas → Projektai

Vykdytojas → Vykdytojai

ER modelis yra vienas iš populiariausių semantinių modelių. Duomenų bazės projektavimas yra sudėtingas ir ilgas procesas. DB loginė struktūra (schema) yra visos

informacinės sistemos pagrindas. Todėl projektuojant dideles DB svarbu sudaryti tikslų dalykinės srities aprašą, t.y. kuo adekvačiau pavaizduoti žinias apie dalykinę sritį. Formalizuotas dalykinės srities aprašas, kuris gali būti supantamas tiek DB specialistams, tiek ir dalyko specialistams, vadinamas semantiniu (arba sampratos, konceptualiu) modeliu. Reliacinis modelis yra lakoniškas ir jame nelabai atsispindi dalykinės srities semantika.

ER modelis vaizduojamas **ER schema**.

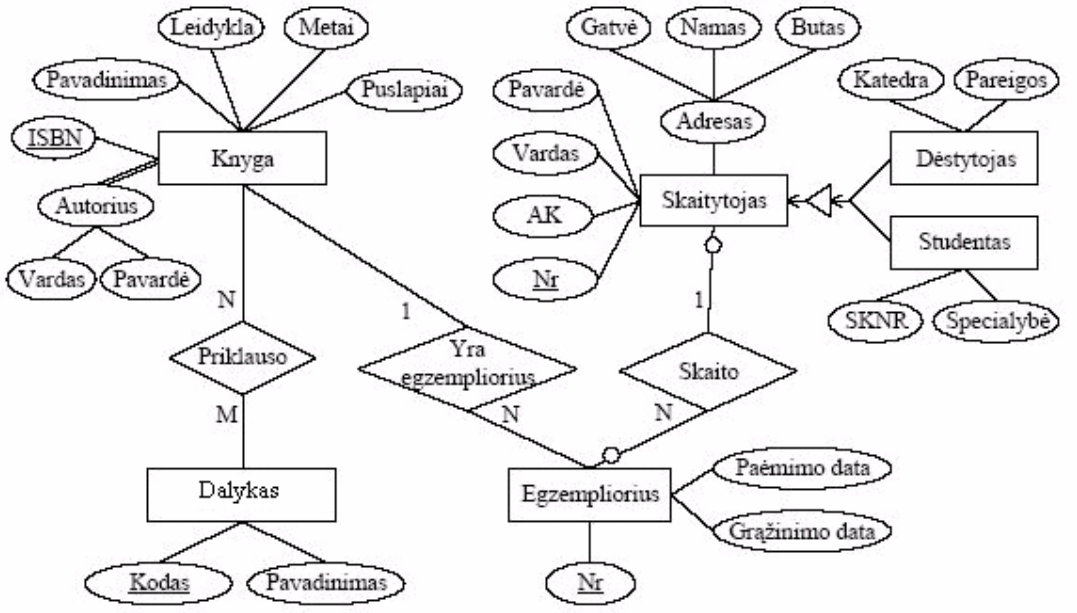
Esybės. Kiekviena esybė vaizduojama atskiru stačiakampiu. Jo viduje rašomas esybės vardas.

Atributai. Kiekvienas atributas ER schemoje vaizduojamas atskiru ovalu. Su atitinkamos esybės stačiakampiu atributas yra sujungiamas ištisine linija. Ovalo viduje rašomas atributo vardas. Sudėtinio atributo sudedamosios dalys vaizduojamos ovalais, sujungtais ištisine linija su sudėtinio atributo ovalu. Raktinių atributų vardai pabraukiami. Daugiareikšmiai atributai jungiami su esybe dviguba linija.

Ryšiai. Kiekvienas ryšis vaizduojamas atskiru rombu su sąryšio vardu viduje. Rombas apvedamas dviguba linija, jeigu sąryšis yra tarp silpnosios esybės ir pagrindinės esybės. Sąryšį vaizduojamtis rombas sujungiamas ištisinėmis linijomis su visais sąryšio dalyviais. Kiekviena tokia linija pažymima „1“, „N“ ar „M“ sąryšio tipui pažymėti. Linija, jungianti sąryšio rombą su silpnąja esybe, yra dviguba.

Potipiai ir virštipiai. Jeigu esybės X1, X2, ..., Xn yra esybės Y potipiai, tai iš kiekvienos esybės Xi (i=1,…,n) stačiakampio brėžiama linija į grafinį elementą, vadinamą *deskriptoriumi* (jį vaizduosime trikampiu), o iš jo į esybės Y stačiakampį su rodykle linijos gale.

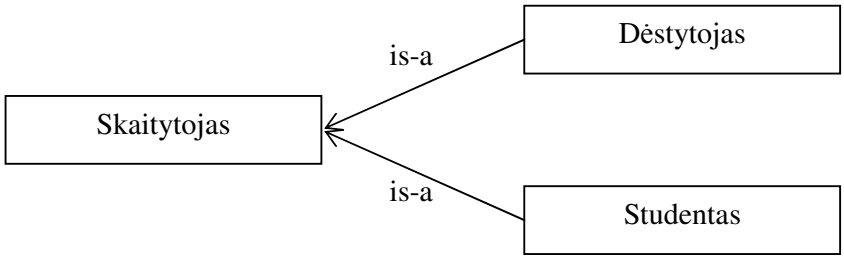
Žemiau pateikiama bibliotekos ER modelio *Biblioteka* ER schemą iš [Baronas 2002, p. 53] ir [Baronas 2005, p. 74]:



Pav. 11.5 Semantinio modelio Biblioteka ER schema iš [Baronas 2005, p. 74]

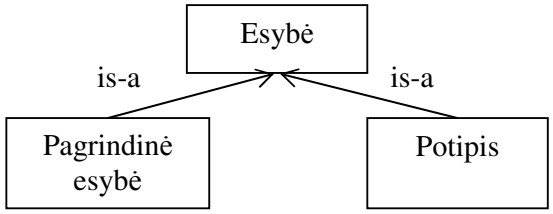
Skirtinguose lygmenyse yra skirtingos sąvokos. Čia mes žiūrėjome duomenų bazės lygmenyje. O į tą patį galima pažvelgti iš semantinio tinklo lygmens. Semantinis tinklas arčiau analizės lygmens, o duomenų bazių lygmuo – projektavimo.

Taigi mūsų požiūriu, mes išskyrėme 4 pagrindines esybes (semantinių tinklų kalba - kategorijos): knyga, skaitytojas, dalykas ir egzempliorius. Vėliau skaitytoją patikslinome dar dvejomis: dėstytojas ir studentas. Semantinio tinklo požiūriu mes juos prijungėme is–a ryšiu.



Pav. 11.6 is-a ryšys susiejęs tris esybes:skaitytoją, studentą ir dėstytoją

Priminsime, jog duomenų bazės lygmenyje skaitytojas yra pagrindinė esybė, o dėstytojas ir studentas – potipiai. Tačiau visa tai yra esybės.



Pav. 11.7 pagrindinės esybės ir potipio sąvokos.

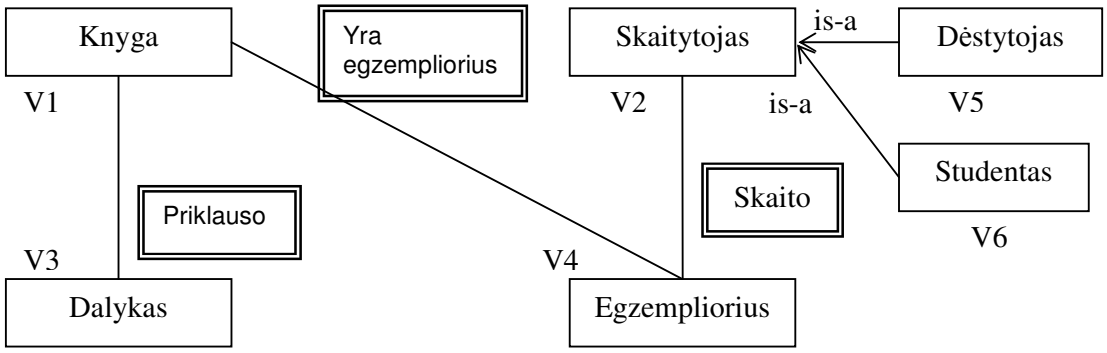
Taigi į skaitytoją žiūrima kaip į aibę, o į studentą ir dėstytoją, kaip į poaibius. Tačiau kaip kalbame apie semantinį tinklą, iš pradžių net nesakoma, ar tai aibė, ar dar kas nors, o tiesiog kalbama apie **sąvokos vardą**. O vėliau bus tikslinama, kur yra aibė arba predikatas.

Jeigu labai griežtai žiūrėti, tai šias kategorijas, kaip grafo viršūnes, galėtume pažymėti V1, V2, V3, V4, V5, V6, o mūsų užrašytos sąvokos, tai jau būtų mūsų viršūnių vardai t.y.:

V1.vardas = knyga;
V2.vardas = skaitytojas;
V3.vardas =dalykas;
V4.vardas = egzempliorius;
V5.vardas = dėstytojas;
V6.vardas = studentas.

Tačiau taip griežtai mes nežiūrėsime.

Toliau pridedame dar tris briaunas (skaito, yra egzempliorius, priklauso) ir, jau turėdami omenyje, kad tai yra santykiai, pažymime juos dvigubu rėmeliu stačiakampiais. Priminsime, kad santykį suprantame kaip lentelę arba predikatą. Taigi mes briaunoms suteikiame daugiau negu tik vardą. Mes pridedame semantiką, pasakome, jog tai santykis, kas jau yra turtingesnis objektas.



Pav. 11.8 Schema, pridėjus santykius

Pastebėsime, jog briaunų pavadinimus galėjome įdėti ir kaip viršūnes. Tačiau tai būtų kitokio tipo viršūnės. Taigi mes jau turėtume dvidalį grafą.

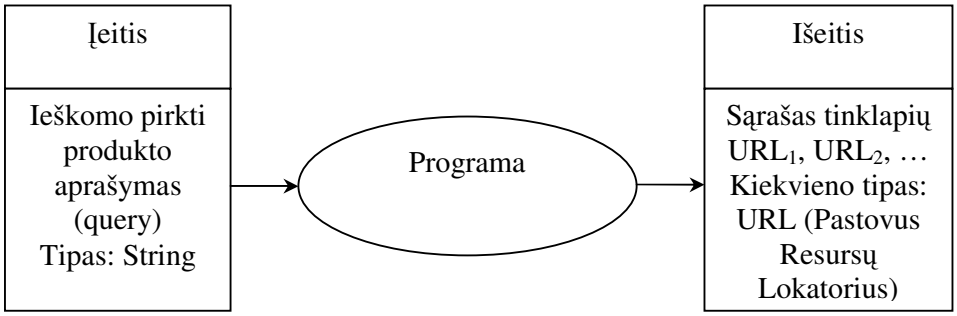
Taip pat pastebėsime, jog santykis, tai Dekarto sandauga arba jos poaibis, o ryšys – tai mistinė sąvoka, mūsų atveju – briauna (arba jeigu naudosime prieš tai minėtą dvidalį grafą – tai pografis, pavyzdžiui santykio viršūnė su savo briaunomis). Taigi nekonkretizuodami ryši kaip santykį, mes nesuteikiame jam semantikos. Nors semantinis tinklas – tai grafas, kurio viršūnėms ir briaunoms yra suteikia semantika, tačiau semantikos nesuteikimas irgi yra tam tikra semantika, laisvė. Čia mes samprotaujame dar visiškai nemąstydami apie realizaciją. Taigi, iš pradžių mes suteikiame viršūnėms vardus, nesuteikdami jokios semantikos, o tik vėliau konkretizuojame. Mes pradedame atsargiai, neprisiimdami iš karto semantikos, įsipareigojimų. Tačiau palaipsniui praturtindami semantinį tinklą, įvedant vis kitokių semantikos pavaizdavimo būdų ir gimsta Pav. 11.5 pavaizduota ER schema.

Toliau yra pereinama prie reliacinės schemos. Pereinant prie reliacinės schemos jau galvojama apie ryšių realizaciją, efektyvumą ir santykiai gali būti verčiami papildoma skiltimi lentelėje arba atskira lentele.

12. Internetinė parduotuvė

Toliau dėstome vadovaudamiesi [Russell, Norvig 2003, p. 344-348] skyriaus „Žinių vaizdavimas“ poskyriui, kuriame nagrinėjama internetinė parduotuvė. Reikalavimų analizė atliekama žinių vaizdavimo tikslu.

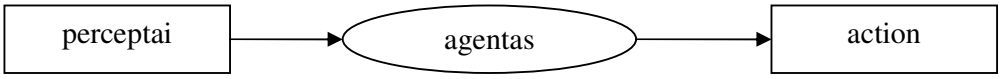
Nustatykime sąvokas, reikalingas internetinės parduotuvės programai. Pradėkime nuo programos įeties ir išeities.



Pav. 12.1 Internetinės parduotuvės programos įeitis ir išeitis.

Mes norime parašyti programą (agentą), kuri naršo po internetą ieškodama, ko prašo vartotojas.

Russell, Norvig tokią programą vadina agentu:



Pav. 12.2 Agento programa.

Žmoniškasis agentas skiriasi nuo kompiuterinio agento:

žmogus operuoja vaizdu ir savo mintyse suveda „laptops“ su „ computers“ ir pasidaro išvada, ar čia yra jo norimas daiktas, ar nėra.

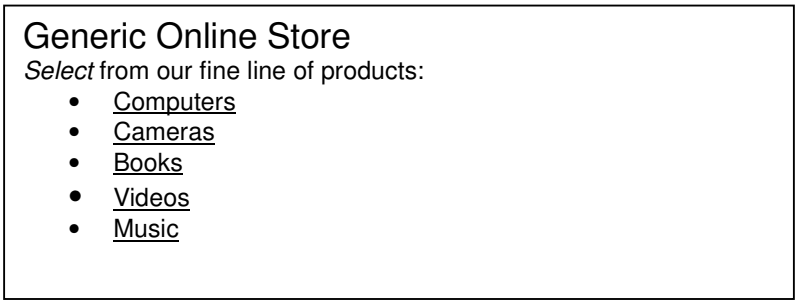
kompiuteris vaizdu neoperuoja, o operuoja HTML failais.

Skirtingas žmogaus ir kompiuterio gebėjimas apdoroti vaizdą plačiai naudojamas internete,atskirti žmogui nuo kompiuterinės programos. Kompiuteris paveikslėliu negali analizuoti, o žmogus gali.

Pavyzdžiui jpeg paveiksliuke iškraipytai pavaizduota simbolių eilutė ir prašoma suvesti ją simboliais. Protingas žmogus gali tai padaryti, o kompiuterinis agentas vargu ar sugebės išanalizuoti šį paveiksliuką.

Dar daugiau, jeigu mūsų parduotuvė vaizduotų kategorijas ne žodžiais, o kompiuterių, knygų ir kitų kategorijų atvaizdais (piktogramomis), tai žmogus vis tiek sugebėtų atskirti, kurioje kategorijoje gali būti norimos prekės, o kompiuteriui tokia užduotis pernelyg sudėtinga.

Pavyzdys pateikiamas „tikslinimo nuo viršaus iki apačios“ metodu (angl. *top-down refinement*).



Pav. 12.3 Vaizdas kompiuterio naršyklės lange

Jeigu duodame užklausą „laptop“, tai programa turėtų susieti su kategorija „computer“ (netgi atskirdama, jog šiame puslapyje tai parašyta daugiskaita). Taigi mums reikia susieti terminus. Ieškoma simbolių eilutė gali būti kaip ir poklasių („laptop“) taip ir virškласių (computing devices).

Pateiksime šio puslapio kodą HTML (HyperText Markup Language) kalba:

```
<h1>Generic Online Store</h1>
<i>Select</i> from our fine line of products
<ul>
   <li><a href="http://gen-store.com/compu">  Computers</a>
   <li><a href="http://gen-store.com/camera"> Kameras
   </a>
   <li><a href="http://gen-store.com/books">  Books  </a>
   <li><a href="http://gen-store.com/videos"> Videos </a>
   <li><a href="http://gen-store.com/music">  Music  </a>
</ul>
```

```
RelevantOffer(page,url,query) ⇔
                    Relevant(page,url,query) & Offer(page)
```

Pastarajame predikate kintamieji yra šių tipų: page – HTML tipo, url – URL tipo, query – String tipo.

Mes duodame užklausą ir norime gauti atsakymą – daug tinlapių. Mūsų rastas tinklapis turi būti relevantinis užklausai ir tas tinklapis turi siųlyti pirkti. Čia slypi žinios apie sąvoką „ apsipirkti“ Jeigu mus domintų „pirkimas“, tai mes naudotume „purchase order“, kas būtų arčiau tikros elektroninės komercijos. Tačiau čia mus domina „apsipirkimas“, kas yra arčiau žioplinėjimoTam kad tinklapis būtų relevantinis, jame turi būti kažkokie raktiniai žodžiai išreiškiantys tai, ko mes norime. Pastebėsime, kad žmogus vertindamas interfeisą žiūri į tuos pačius kriterijus. Mūsų ieškomas tinklapis nebūtinai bus šakninis, iki jo gali vesti tinklapių grandinė. Mūsų programa operuos tik su tekstu, tačiau žinoma galima daug sudėtingesnė programa, kuri bandys pagal tinklapiuose pateikiamą vaizdinę informaciją (paveiksliukus), kažką išsiaiškinti.

Detalizuokime predikatą Offer.

```
Offer(Page) ⇔ (InTag('a',str,page) ∨ InTag('form',str,page))
& (In('buy',str) ∨ In('price',str))
InTag(tag,str,page) ⇔ In('<' + tag + str + '</' + tag,page)
In(sub,str) ⇔ ∃i str[i:i+Length(sub)-1] = sub
```

Predikatas `In(sub,str)` išreiškia, kad simbolių eilutė `sub` įeina į simbolių eilutę `str`. Pavyzdžiui:

In('KAD', 'ABRAKADABRA') = true

In('laptopiukai', 'ABRAKADABRA') = false

Mūsų pavyzdyje pasakyta, kad tinklapis Page yra pasiūlomas jei yra po žymių <a> arba <form> žodis buy, arba žodis price. Ir mūsų žinios šiame pavyzdyje yra ('Buy', 'Price'). Tai yra tai kaip [Russell, Norvig 2003] supranta „offer“ sąvoką. Tačiau norint parašyti programą, mums reikia savo predikatus išskleisti smulkiau. Taigi mes išreiškėme, jog eilutė 'Buy' arba 'Price' turi būti žymėje ir, antra, kad ta žymė turi būti puslapyje. Mūsų aprašyti trys predikatai yra detalizavimas (griežta specifikacija) predikato „**Offer**“ t.y. ką reiškia pasiūlymas. Jeigu pradėtume programuoti, tai pamatytume, jog ši specifikacija nėra pilna. Pavyzdžiui, neaišku, ar eilutės 'Buy' ir 'Price' turi būti iš didžiosios raidės, ar ir kiti variantai tinka. Jeigu norėtume suprogramuoti tai pakankamai rimtai, tai reikėtų sudėti visas žinias apie sąvoka „**Offer**“. Tikriausiai ten būtų daugiau negu du žodžiai ('Buy', 'Price'). Daugiau informacijos apie tinkamą jų vietą ir pan.

Grįžkime prie pagrindinio predikato.

Etapas 1.

Relevant Offer(page,url,query) ⇔
Relevant(page,url,query) & **Offer**(page)

Predikato išskleidimas yra mūsų atliekamos analizės dalis.

Etapas 2.

Amazon ∈ OnlineStores & **HomePage**(Amazon, http://www.amazon.com)

Ebay ∈ OnlineStores & **HomePage**(Ebay, http://www.ebay.com)

GenStore ∈ OnlineStores & **HomePage**(GenStore, http://www.gen-store.com)

Vardai Amazon, Ebay, GenStore žymi individualias sąvokas. Jų tipas yra identifikatorius. Elektroninių adresų tipas URL. Primename, kad URL yra tipo String poaibis.

Pateikdami šiuos tris pavyzdžius mes norėjome pasakyti faktą, kad pasaulyje yra tinklapių, kurie yra elektroninės prekyvietės. (∃x, kur Elektronine_prekyvietė(x) = true, kur Elektroninė_prekyvietė yra predikatas.)

Prisiminę mūsų programos idėją Pav. 12.1. pradėkime nuo kažkur ir eikime per internetą ieškodami pasiūlymų bei relevantinių puslapių. Tokiu būdu pradėkime nuo prekyviečių aibės, kuri nėra tuščia.

Etapas 3. Išskleisime predikatą `Relevant`.

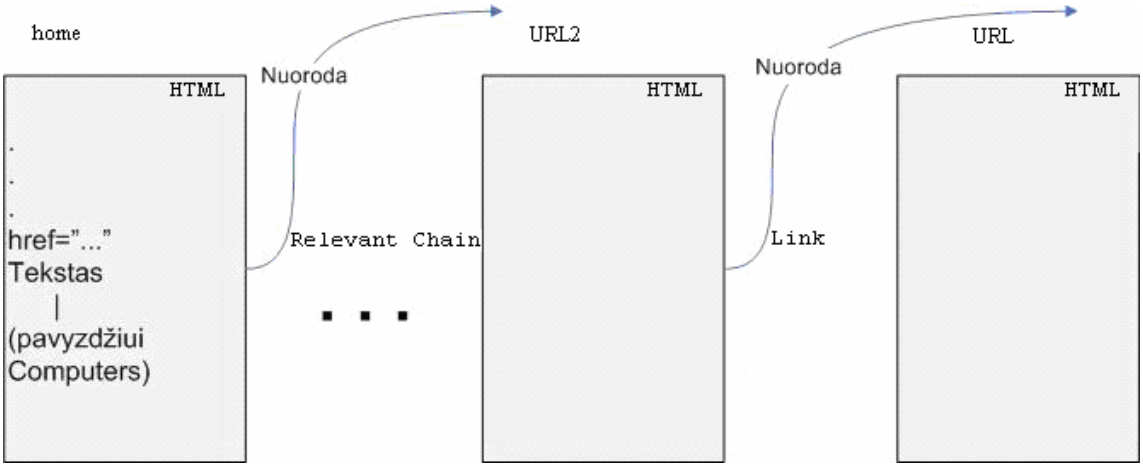
Relevant(page,url,query) ⇔ ∃store, ∃home ((store ∈ OnlineStores) & **HomePage**(store, home) & (∃url_2 **Relevant Chain**(home, url_2, query) & **Link**(url_2, url)) & **Page**=**GetPage**(url))

Pirmiausia priminsime tipus: store tipas Identifikatorius, home tipas URL.

Kaip pastebėjote – skleisdami predikatą `Relevant` įvedėme predikatą **Link**. **Link**(from_url,to_url), kur tiek `from_url`, tiek `to_url` yra tipų URL. Jis reiškia, jog pirmajame tinklapyje yra betarpiška nuoroda nuoroda į antrąjį puslapį.

Semantika yra pavaizduota Pav. 12.4.

home – tipas URL url_2 – tipas URL url – tipas URL



Pav. 12.4 Predikato **Relevant** iliustracija.

Mes turime pradinę būseną ir galutinę būseną. Galutinė būsena yra puslapis, kuriame yra pasiūlymas (oferta). Pradiniame puslapyje mes tiesioginės nuorodos į jį galime ir neturėti.

Taigi, mums iki galutinio puslapio dar reikia nukeliauti, nes savo pradiniame puslapyje, mes galbūt nieko neturime.

Puslapis atitinka procedūros `page:= getPage(url)` kvietimą. Tai naršyklės iškvietimas.

Tinklapis su home turi identifikatorių store, tačiau tai kintamasis.

store ∈ OnlineStores

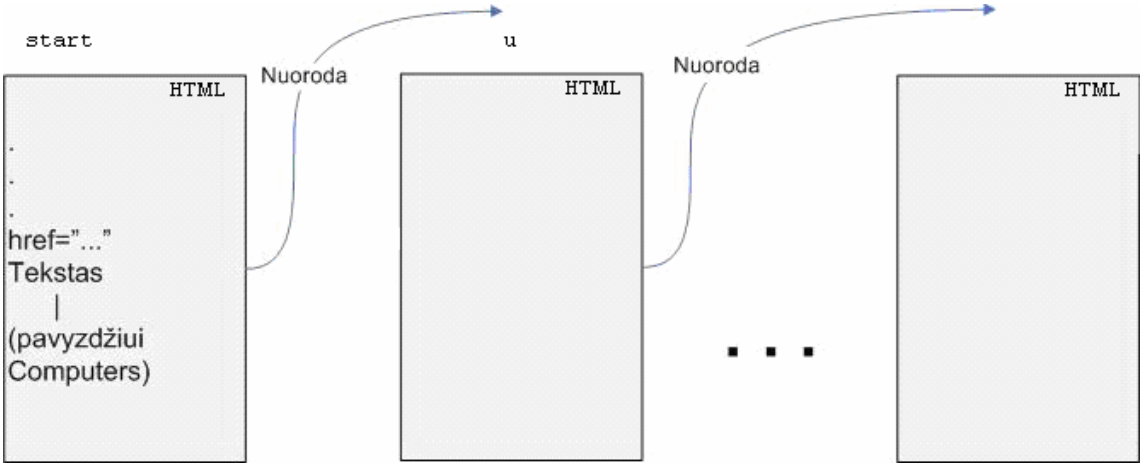
Taigi perrenkame visas turimas namų parduotuves ir nuo konstruojame kelia.

Predikatą **`RelevantChain`**`(home, url2, query)` atitinka Pav. 12.4 pirmieji du stačiakampiai, kur query yra užklausa mūsų programai. Ir šis predikatas pasako, kad egzistuoja grandinė į url_2. O naudodami predikatą **`Link`**`(url2,url)` (žr. Pav. 12.4 antrą ir trečią stačiakampius) baigtiniu tekstu užrašome potencialiai begalinę sekų aibę.

Etapas 4.

`RelevantChain``(start, end, query)` ⇔ `(start = end)` ∨ `(`∃`u` ∃`text (`**`LinkText`**`(start, u, end)` & **`RelevantCategoryName`**`(query, text)` & **`RelevantChain`**`(u,end,query)))`

Čia parametrų tipai yra šie: `start` tipas URL, `end` tipas URL, `query` tipas String.



Pav. 12.5 Predikato **RelevantChain** iliustracija.

Rekursyviai apibrėžtas predikatas RelevantChain Pav. 12.5 pradeda nuo „Start“ puslapių. Jame pasirenka nuorodas su jų inkaru (anchor) tekstu, jeigu tas tekstas yra susijęs su užklausa (kas patikrinama predikatu RelevantCategotyName). Tuomet rastai nuorodai rekursyviai iškviečiamas predikatas RelevantChain, kuriuo ieškomi susiję puslapiai nuo naujosios nuorodos. Kai patenkame į predikatą RelevantChain, jis tuoj pat gražina kaip „end“ puslapį, tą ties kuriuo mer pradedame („start“).

Query mūsų programai užduodamas kaip tekstas, pavyzdžiui, kaip „Laptops“. Taip pat gali būti „Ieškau laptopiukų“. Tai paprasčiausias tekstas. Žmogus mašinai (kad ir Google) užduoda tekstą, kuris yra kategorizuojamas tam tikrais žodžiais. Pavyzdžiui, žodis „lapiukas“ kategorizuojant gali būti siejamas su kailiniu žvėreliu lape arba su popieriaus lapu.

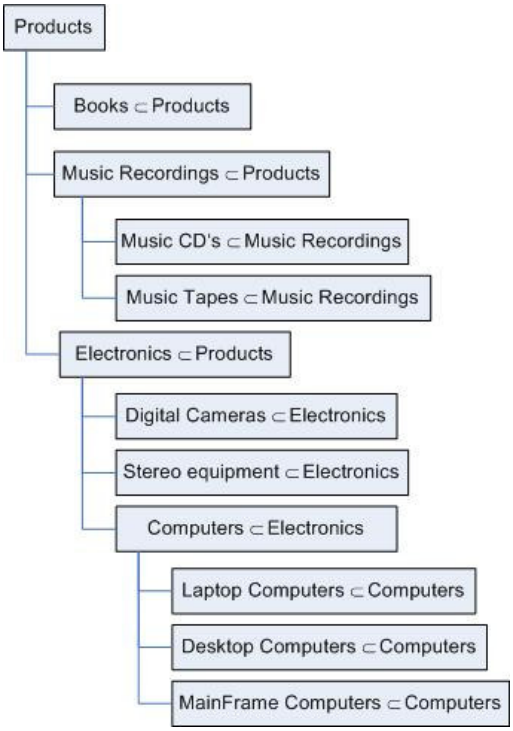
Toliau aiškinsime Pav. 12.4. Pirmajame stačiakampyje yra pateikiamas tekstas, kuris pagal nutylėjimą yra visuose stačiakampiuose ir [Russel, Norvig 2003] yra vadinamas inkaru (angl. *anchor*). Šis tekstas bus matomas kompiuterio ekrane. Analogiškai galėtume pavadinti schemą ant automobilio pavarų perjungimo svirties.

Mūsų predikatai

```
LinkText(start, u, text) & RelevantChain(u, end, query)
```

kur `text`, pavyzdžiui, įgija reikšmę „computers“, o `query` – „ieškau laptopiukų“.

Bandysime paaiškinti kategorizavimo sąvoką. Įsivaizduokime turime produktų hierarchiją pavaizduotą medžiu (is-a):



Pav. 12.6 Produktų hierarchija pavaizduotą medžiu (is-a)

Pavyzdžiui, žodis „book“, atitinka sąvoką (kategoriją) Books, tai yra galime apibrėžti predikatą Name(„book“,Books), analogiškai Name(„music“,Music Recordings), Name(„CD‘s“,Music CD‘s), Name(„tapes“,Music tapes), Name(„electronics“,Electronics), Name(„digital cameras“,Digital Cameras), Name(„stereos“,Stereo Equipment), Name(„computers“,Computers), Name(„laptops“,Laptop Computers), Name(„mainframes“,MainFrame Computers), Name („laptopiukai“,Laptop Computers).

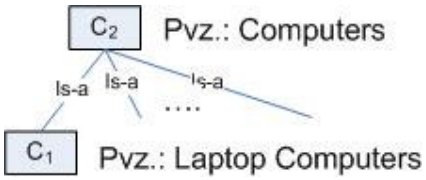
Šio predikato tikslas yra susieti query tekstą, su kategorijomis galutiniame rezultate per tarpinį tekstą internetiniuose puslapiuose (anchor). Mūsų programa turėtų mokėti suprasti lietuvių kalbos linksnius ir nepriklausomai nuo jų kategorizuoti tekstą. Be to, apibrėžiant predikatą Name mes turime susitarti, ar sieti tekstą su labiausiai bendra, ar labiausiai specifine kategorija. Pavyzdžiui, ar bus Name(„laptops“, LaptopComputers), arba Name(„laptops“, Computers). Dar norėtųsi atkreipti dėmesį į sinonimijos problemą. Pavyzdžiui trumpinys CDs gali reikšti Compact Disks arba Certificate of Deposits, ir pan.

Etapas 5.

Kaip susieti užklausą query su tekstu. Išskleiskime predikatą:

$RelevantCategoryName(query,text) \Leftrightarrow \exists c_1 \ \exists c_2 \ ((c_1 \subseteq c_2) \vee (c_2 \subseteq c_1))$

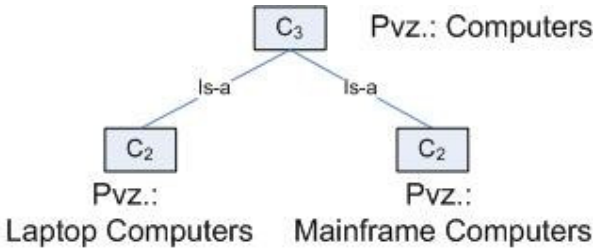
Turime kategoriją C_2 ir yra is-a ryšys su subkategorijomis.



Pav. 12.7 $c_1 \subseteq c_2$.

Pav. 12.8 . $c_2 \subseteq c_1$

Tačiau negalima žiūrėti, kaip pavaizduota Pav. 12.9



Pav. 12.9 Nėra nei $c_1 \subseteq c_2$, nei $c_2 \subseteq c_1$.

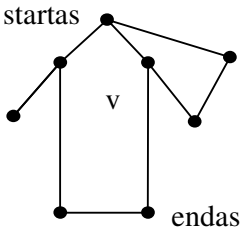
Turint $(c_1 \subseteq c_2) \vee (c_2 \subseteq c_1)$ yra galimi trys atvejai:

1. text ir query priklauso tai pačiai kategorijai. Pvz., text =„laptop computers", query=„laptops", tačiau net ir šiuo atveju turime neužmiršti darbo su galūnėmis.
2. text yra viškategoris (kaip viršaibis) atžvilgiu query. Pvz., text=„computers", query=„laptop computers".
3. text yra subkategorija atžvilgiu query. Pvz., text=„ultra light notebooks", query=„laptops"

Šioje vietoje pasibaigia [Russell, Norvig 2003] „know-how" atskleidimas.

Toliau parašykime specifikaciją predikato `Linkas` programai:

```
Linkas(startas,endas) = Briauna(startas,v) & Linkas(v,endas)
```



Pav. 12.10 Toliau specifikaciją detalizuoti galima ir kelio paieškos į plotį, ir įgylį.

Šią rekursyvią specifikaciją grafiškai galima pavaizduoti, kaip kelią nuo nuo startas iki endas

13. Argumentavimas teisėje

Šį skyrių dar galėtume pavadinti: „Argumentavimo teisėje automatizavimas“. Jis parengtas pagal [Bench-Capon 2002] straipsnį žurnale „Artificial Intelligence and Law“.

Kalbėsime, apie loginį išvedimą, tačiau mūsų siekiai liečia argumentavimą ir semantinius tinklus, o ne matematinės logikos taikymą.

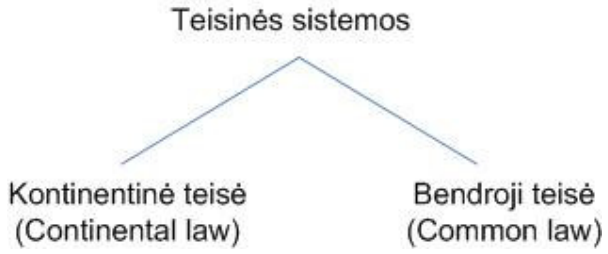
Nagrinėsime tris bylas iš amerikiečių teisės vadovėlio (pavyzdinius supaprastintus precedentinės teisės precedentus).

Įvesdami į dalykinę sritį norėtume pastebėti, kad civilinėse, tiek ir baudžiamosiose bylose visada būna dvi pusės tai: ieškovas ir atsakovas. Tiesa, baudžiamosiose bylose ieškovo vaidmenyje paprastai yra valstybė.

Kaip jau minėjome, mes nagrinėsime precedentinės teisės atvejus, todėl norėtume pastebėti, kad egzistuoja dvi pagrindinės teisinės sistemos – kontinentinė ir bendroji teisė.

Bendrojoje teisėje yra vadovaujamasi precedentais. Ji dar vadinama anglo-saksų teise arba precedentine teise. Bendroji teisė yra įsigalėjusi Jungtinėje Karalystėje, JAV ir eilėje kitų šalių.

Kontinentinėje teisėje pagrindiniu teisės šaltiniu yra įstatymas, o ne precedentas. Kontinentinė teisė dar vadinama statutine teise. Kontinentinės teisinė sistema yra įsigalėjusi Lietuvoje, Vokietijoje ir kitose Europos šalyse.



Pav. 13.1 Kontinentinė teisė ir bendroji teisė kaip teisinių sistemų pavyzdžiai

Kalbant apie šių teisės formų paplitimą pasaulyje pastebėsime, kad kontinentinė teisė (dar kitaip vadinama statutine arba Romėnų teise) yra įsigalėjusi žemyninėje Europos dalyje, pavyzdžiui, Prancūzijoje, Vokietijoje ir kitose šalyse, tame tarpe ir Lietuvoje. Bendroji teisė (kitaip dar vadinama precedentine teise arba anglo-saksų teise) yra įsitvirtinusi Jungtinėje Karalystėje, Jungtinėse Amerikos Valstijose ir eilėje kitų šalių.

Taigi ginčo teisena bendrojoje teisėje vyksta taip – ieškovas iškelia ieškinį ir pateikia precedentų, kurie palaiko jo pusę, savo ruoštu atsakovas rengdamas atsikirtimą irgi remiasi precedentais. Paminėtina, kad precedento amžius, tai yra data, kada buvo priimtas vienas ar kitas teismo sprendimas nei sumenkina, nei padidina jo vertės.

1. Ieškovas remiasi precedentu → Atsakovas atsikerta → Ieškovas atsikerta į atsikirtimą.
2. Atsakovas remiasi precedentu → Ieškovas atsikerta → Atsakovas atsikerta į atsikirtimą.

Grįžkime, prie mūsų nagrinėjamo pavyzdžio, kaip jau minėjome turime tris bylas, toliau jas ir pateiksime.

1. Pierson vs. Post (1805 m.) „lapė atviroje vietovėje“.

   Pierson‘as medžiojo lapę niekieno žemėje (atviroje visiems), lapė nuo jo bėga ir ją pagauna Post‘as. Pierson‘as paduoda Post‘ą į teismą. Teismo

sprendimu, teisus Post‘a. Nors intuityviai atrodytų, jog Pierson teisus, tačiau teisėje galioja toks principas, kad kas pagauna laukinį gyvūne, to ir nuosavybė.

2. Keeble vs. Hickeringill (1707 m.) „antys nuosavoje kūdroje“.

   Keeble‘as savo sklype, kūdroje masalu (jauku) viliodavo antis, jas nušaudavo ir pardavinėdavo. Šitos antys buvo jo pragyvenimo šaltinis. Jo kaimynas Hickeringill‘as iš pavydo šaudydamas į orą gąsdindavo antis, kad šios nesileistų į Keeble‘o kūdrą. Keeble‘as padavė savo kaimyną į teismą. Teismo sprendimu teisus Keeble‘as.

3. Young vs. Hitchens (1844 m.) „žuvys jūroje“.

   Verslininkai Young‘as ir Hitchens‘as gaudė žuvis (atviroje) jūroje. Young‘as pastatė pusės apskritimo tinklą (apie 100 m. pločio) ir palaipsniui jį siaurino, kai atstumas tarp tinklo kraštų tapo keliasdešimt metrų atvyko Hitchens‘as, priešais pastatė nedidelį tinklą ir pagavo visas žuvis. Young‘as padavė savo konkurentą į teismą. Teismo sprendimu bylą laimėjo Hitchens‘as.

Šiame skyriuje siekiame formalizuoti teismo sprendimo trečioje byloje priėmimą ir pagrįsti teismo sprendimą. Bandykime formalizuoti faktorius, kurie įtakoja teismo sprendimą. Sudarykime lentelę:

Metai	Ieškovas vs. Atsakovas	LIVELIHOOD Pragyvenimas A	OWNLAND Nuosava žemė B	NOTCAUGHT Nepagavo C	COMPETE Konkurentai D	OPEN Atvira žemė (jūra) E	Matematiškai pagal aplinkybes
1805	Pierson vs. Post „lapė atviroje vietovėje“	—	—	NOTCAUGHT	—	OPEN	C&E
1707	Keeble vs. Hickeringill „antys nuosavoje kūdroje“	LIVELIHOOD	OWNLAND	NOTCAUGHT	—	—	A&B&C
1844	Young vs. Hitchens „žuvys jūroje“	LIVELIHOOD	—	NOTCAUGHT	COMPETE	OPEN	A&C&D&E

Pav. 13.2 Faktoriai, įtakojantys teismo sprendimą.

Faktoriai lentelės antraštėje yra sužymėti raidėmis A, B, C, D, E, ir gali būti suskirstyti į ieškovą remiančius ir atsakovą remiančius. Tad faktoriai A ir B remia ieškovą, faktoriai C, D ir E remia atsakovą.

Faktiškai formalizavimui užtektų ir keturių faktorių, nes jai yra OWNLAND (Nuosava žemė) faktorius, jau nebegali būti OPEN (atvira žemė) faktoriaus. Šiems faktoriams suteikę vienodą svorį ir žinodami, kurie palaiko ieškovą, kurie palaiko atsakovą, galėtume suformuoti tokius santykius:

Byla	Matematiniai santykiai

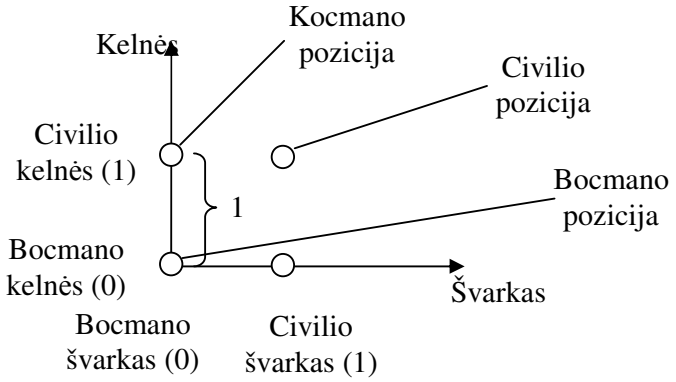
Pierson vs. Post	0:2
Keeble vs. Hickeringill	2:1
Young vs. Hitchens	1:3

Pav. 13.3 Faktorių santykiai, palaikantis ieškovą ir atsakovą.

Tokie matematiniai santykiai realybėje yra neįmanomi, nes teismas turi iš esmės įsigilinti į ieškinį ir tik tada priimti vieną ar kitą sprendimą, o bandant formalizuoti sprendimo priėmimą susidursime su dviem pagrindinėm problemom. Pirmiausia visų galimų faktorių išrašymas yra praktikai neįmanomas dalykas, antra svoriai kurie tenka tiems faktoriams yra sunkiai apibrėžiami.

Šiai problemai vaizdžiau paaiškinti pabandykime panagrinėti anekdotą apie aprangas. Įsivaizduokime, civilio apranga susideda iš civilio kelnių bei civilio švarko, taip pat bocmano apranga susideda iš bocmano kelnių ir bocmano švarko. Situacija ant laivo denio išeina žmogus su bocmano švarku ir civilio kelnėmis. Jo ten dirbęs jūrininkas klausia: – Sakykite Jūs locmanas. Šis atsako: – Ne. – Tada Jūs tikriausiai bocmanas. Šis vėl atsako: – Ne. – Tai kas tada Jūs esat? Nepažįstamasis atsako: – Aš esu Kocmanas.

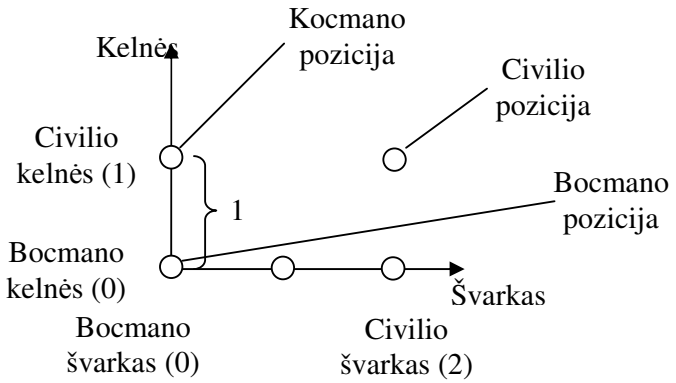
Pabandykime formalizuoti galimus aprangų tipus:



Pav. 13.4 Aprangos formalizavimas, kai švarkas ir kelnės turi lygius svorius: 1.

Bocmaną Pav. 13.4 atitinka taškas (0,0), civilį – taškas (1,1). Tačiau turėdami žmogų pavyzdžiui (0,1) jau turime su kažkuo sutapatinti. Teisėje tai galėtų būti vadinama aplinkybių pasvėrimu.

Matematikoje žinome, kad galime palyginti vektorius (0,0) < (1,1). Tačiau negalime palyginti vektorių (0,1) ir (1,0). Tokius vektorius lygtimi galėtume užrašyti taip c_1*y+c_2*x (nors yra ir daugiau vektorių ilgių skaičiavimo būdų). Čia x ir y tai koordinatės, o c_1 ir c_2 tai pozicijų svoriai. Laikant kad pozicijos turi vienodą svorį Kocmaną galime užrašyti taip: 1*1 + 1*0. O bandant Kocmaną pritraukti prie kurios nors žinomos pozicijos (Bocmano arba Civilio) mes susidursime su problema, kad atstumai iki jų yra vienodi ir yra lygus c, kur $c_1=c_2=c$. Jei pavyzdžiui, civilio švarko svorį padidinsime iki 2, Kocmano atstumas iki civilio irgi taps 2, o jo atstumas iki Bocmano išliks 1 (žr. Pav. 13.5.).



Pav. 13.5 Aprangos formalizavimas, kai švarkas turi svorį 2, o kelnės svorį 1

Analogiška svorių problema iškyla iš mūsų nagrinėjamame pavyzdyje su teismo sprendimą įtakojančiais faktoriais. Tokių svorių parinkimas (jei tartume, kad svoriai įtakoja teismo sprendimą) būtų svarbus nulemiant teismo baigtį.

Taip pat nagrinėjamus faktorius galėtume dar papildomai skirstyti į dimensijas. Pavyzdžiui, jei turėtume faktorių pagavo galėtumėme išskirti dimensijas kaip: sužeidė, pagavo, prisilietė ir pan. Juk skirtingai vertintume faktą, jeigu Post‘as būtų pagavęs Pierson‘o sužeistą lapę.

Norėtume grįžti prie teisinio argumentavimo temos. Mes jau minėjome, kad turime precedentinę teisinę sistemą ir nagrinėdami Young vs. Hitchens bylą galėtume taip nusakyti argumentavimą. Tačiau trumpam nukrypsime i kontinentinę teisę, civilinį kodeksą:

4.59 straipsnis. Laukiniai gyvūnai

Laisvėje esantys laukiniai gyvūnai, kurie laikantis įstatymų buvo pagauti arba nušauti, tampa juos pagavusiojo arba nušovusiojo nuosavybe, jeigu įstatymų nenustatyta kitaip.

Pagavo_ar_nušovė(Asmuo, Gyvūnas) $\Rightarrow$ Nuosavybė(Asmuo, Gyvūnas);

Bet čia nėra absoliutu, nes yra pasakytą: „laikantis įstatymų“ bei „jeigu įstatymų nenustatyta kitaip“. Kitaip tariant, jeigu nėra išimčių. Taigi čia turime taisyklę aiškiems atvejams. O ginčas todėl ir kyla, kad yra neaišku. Grįšime prie precedentinės teisės.

Young‘as kaip argumentą, kodėl teismas turėtų priimti jam palankų sprendimą pateikia Keeble vs. Hickerigill bylos precedentą. Jis tvirtina, jog jo atvejis panašus, sutampa faktoriai A ir C (žr. Pav. 13.2). Hitchens‘as atsikerta, kad toje byloje nebuvo faktorių D ir E, o šioje byloje nėra faktoriaus B. Pagal šiuos kriterijus Young‘as neturi kuo atsikirsti į atsikirtimą. Tuomet Hitchens‘as remiasi Pierson vs. Post precedentu. Young‘as atsikerta, kad turi faktorių A. Tuomet Hitchens‘as atsikerta į atsikirtimą, sakydamas, kad užtad turi faktorių D. Kaip matome, į abu precedentus buvo iš esmės atsikirsta, kad nagalima jais vienareikšmiškai remtis. Tačiau teismas vis tiek privalo būtinai nuspręsti kieno nors naudai.

Jeigu lyginti atsikirtimus faktorių skaičiumi, tai pirmuoju atvėju Hitchens‘as atsikerta D ir E buvimu ir B nebuvimu, o Young;as neturi ką pasakyti, taigi 3:0 prieš rėmimąsi Keeble vs. Hickerigill bylos precedentu. Antruoju atveju Young‘as atsikerta faktoriaus A buvimu, o Hitchens‘as atsikerta į atsikirtimą faktoriaus D buvimu, taigi 1:1. Tačiau svoris neaiškus.

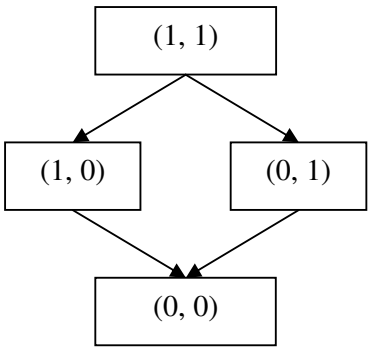
Kaip jau minėjome anksčiau čia teismas nusprendė, kad teisus buvo Hitchens‘as. Mūsų uždavinys bus pabandyti tai formalizuoti.

Atkreipsime dėmesį, jog kaip buvo išspręstos minėtos nagrinėjamos bylos, tai buvo ne tik atsakyta, kas laimėjo, o turėjo būti ir argumentuojama. Šis argumentavimas mums suteikia papildomos informacijos.

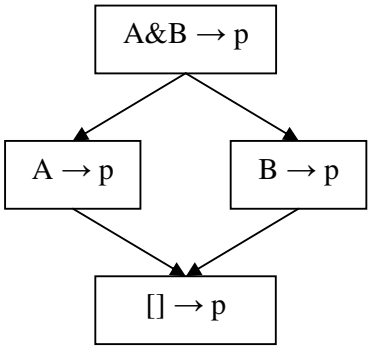
Norėtume paminėti, kodėl visgi yra svarbu formalizuoti žinias, šiuo atveju įvesti įvairius faktorius pasitaikiusius byloje. Kadangi kaip jau buvo minėti, Bendroji teisė remiasi precedentais, tai šie precedentai yra kaupiami. Ir norint automatizuoti šių precedentų paiešką (siekiant, kad precedentų ieškojimas, būtų įmanomas ne tik vartant storas knygas, kuriose jie surašyti) turime įvesti kriterijus pagal kuriuos galėtume atlikti paieška.

Paminėsime, jog rėmimasi precedentais esmė, kad panašios aplinkybės sąlygoja panašius sprendimus.

Grįždami prie bylų nagrinėjimo, sudarysime pro-ieškovo (angl. *plaintiff*) gardelę (Pav. 13.7) panašiai kaip ir vektorių sutvarkymo gardelė (Pav. 13.6). Jeigu ieškovas turi A ir B faktorius, tai tarsime, jog jis laimi, taip pat kaip ir vieną iš jų, arba netgi nei vieno.

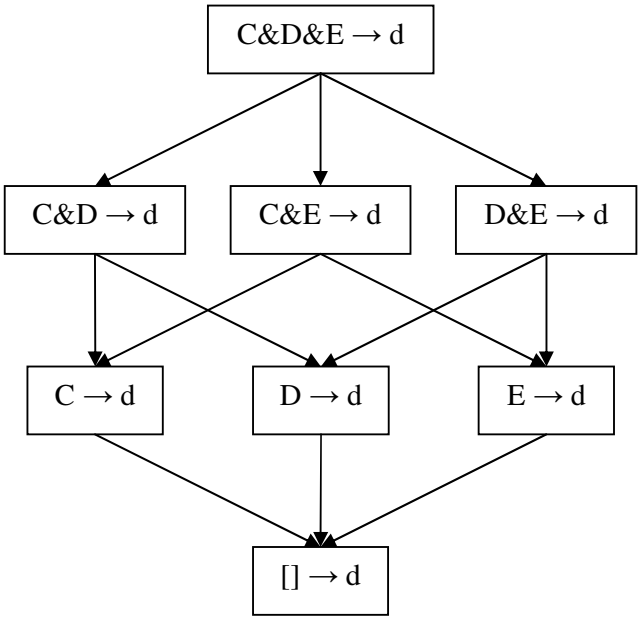


Pav. 13.6 Aibės, turinčios du elementus, poaibių sutvakymas



Pav. 13.7 Pro-ieškovo faktorių gardelė

Tokiu pat principu sudarome ir pro-atsakovo (angl. *defendant*) gardelę (Pav. 13.8), tačiau čia jau 3 faktoriai, todėl geometriškai bus panašu į kubą.



Pav. 13.8 Pro-atsakovo faktorių gardelė

Bandykime atlikti mūsų turimų trijų atvejų formalizavimą. Žiūrėkite Pav. 13.9, kur R_N-nusakys taisyklę. O plaintiff reikš – teisus ieškovas, defendant – teisus atsakovas.

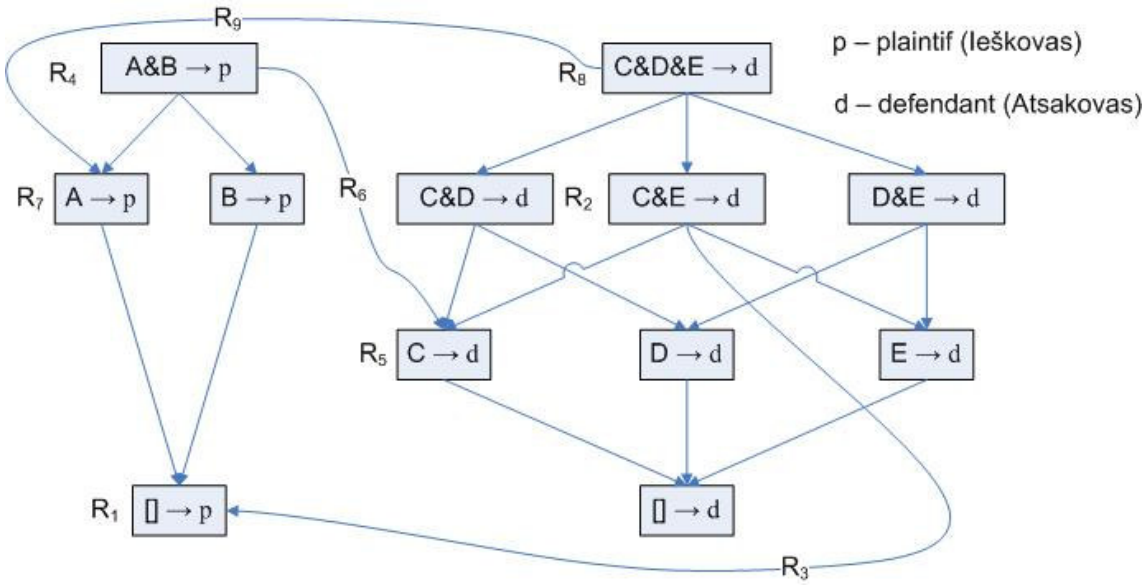
Pierson vs. Post	Keeble vs. Hickeringill	Young vs. Hitchens
R_1 **if** [] **then** plaintiff	R_4 **if** A&B **then** plaintiff	R_7 **if** A **then** plaintiff
R_2 **if** C&E **then** defendant	R_5 **if** C **then** defendant	R_8 **if** C&D&E **then** defendant
R_3 $R_2 > R_1$	R_6 $R_4 > R_5$	R_9 ?

Pav. 13.9 Tyrimų formalizavimas, nusakantis atsakovo ar išškovo teisumą.

Atkreipsime dėmesį, jog R_3, R_6 ir R_9 yra kitokio tipo, nei likusios taisyklės.

Atkreipsime dėmesi, jog sprendžiant Pierson vs. Post bylą, pagal šiuos faktorius (atsakovo naudai yra faktoriai C ir E, o ieškovo naudai nieko nėra) atrodytų akivaizdu, jog vienoje pusėje kažko turėjimas yra geriau, nei kitoje pusėje nieko neturėjimas. Nors iš pradžių skaitant bylą, intuityviai atrodė, jog Pierson‘as teisus. Taip atsitiko todėl, kad ne visi faktoriai buvo formalizuoti. Pavyzdžiui mes neformalizavome, kad buvo išsiruošta medžioklei, kad Pierson‘as persekiojo gyvūną, o Post‘as išlindo ir nušovė.

Bandykime atsakyti į klausimą: ar $R_7 > R_8$, ar $R_8 > R_7$



Pav. 13.10 Pirmoji gardelė, vaizduojanti matematinę teoriją, kuri pagrindžia argumentavimą trijuose precedentuose.

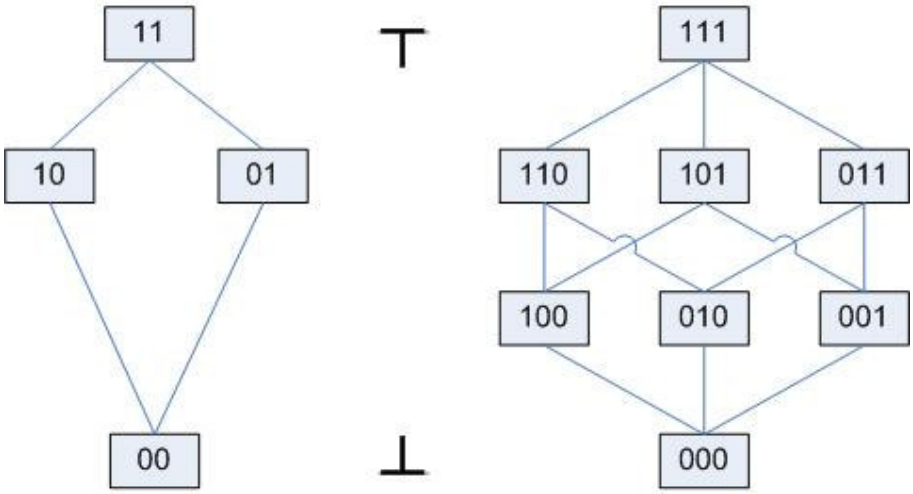
Kaip matėme Pav. 13.9 bei Pav. 13.10, taisyklės R_3, R_6, R_9 tai teismo sprendimai Pav. 13.10 pavaizduota gardelė, pagrindžianti precedentus. Tačiau R_9 mes negalėjome išskaičiuoti iš praitų precedentų, o paėmėme jį iš teismo sprendimo.

Priminsime, jog faktorių kiekis nenuliame sprendimo. Todėl be teismo sprendimo šios gardelės mums visais atvejais nepadės.

Tai, kad faktorių kiekis nenuliame sprendimo, o reikalingi ir svoriai, iliustruoja pavyzdys su banknotais: Visų pirma, jeigu veinas žmogus turi vieną banknotą, o kitas žmogas turi 20 banknotų, mes dar negalime atsakyti, kuriam yra geriau. Mums reikalingi banknotų svoriai. Pavyzdžiui, sužinoję, jog tas vienas banknotas yra 100lt., o natrojo žmogaus banknotų yra po 1lt, mes jau žinome, kad pirmuoju atveju yra geriau, nors banknotų skaičiumi atrodė visiškai atvirkščiai.

Taip pat neaiškus klausimas palyginti kitokį baknotų rinkinį, su turimais atvejais, pavyzdžiui, 3banknotai po 10lt. Pagal banknotų skaičių yra arčiau pirmojo atvejo (1 banknoto), negu antrojo (20 banknotų). O pagal pinigų sumą arčiau antrojo atvejo(30lt arčiau 20lt., o ne 100lt.).

Dar kartą priminsime gardelės sąvoką. Gardelė tai grafas, kuriame nėra ciklų ir yra dalinis sutvarkymas. Šis grafas yra sutvarkytas taip, kad kiekvienas dvi gretimas jo viršūnes galima vertinti taip, kad arba viena kuri nors viršūnė yra didesnė už kitą arba jos nepalyginamos. Įsivaizduokime vektorių CDE {C, D ir E tai faktoriai, jei visi šie faktoriai yra turėsime vektorių (111), analogiškai, jei savo nagrinėjamame atvejyje neturėsime nei vieno iš šių faktorių mūsų vektorius bus (000)}. Binarine išraiška mūsų gardelę galima pavaizduoti Pav. 13.11.



Pav. 13.11 Dvimatė gardelė – kvadratas ir trimatė gardelė – kubas.

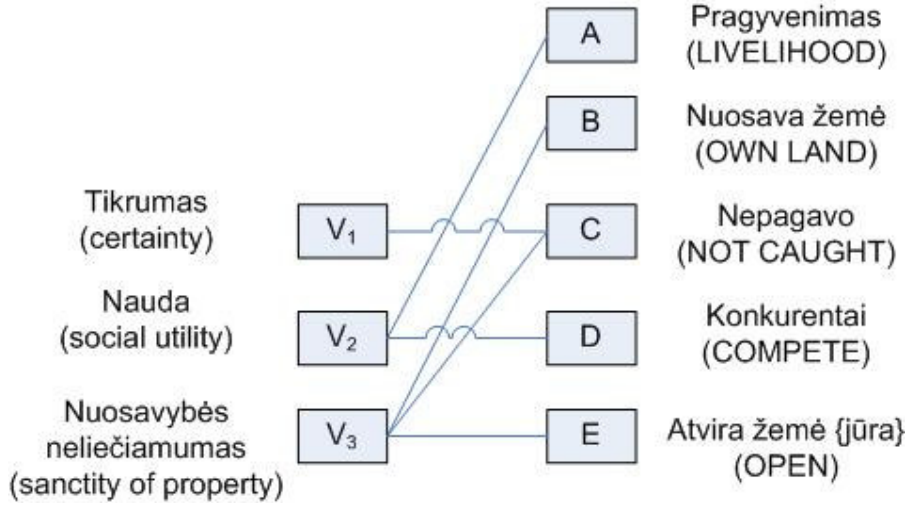
Grafas (Pav. 13.10) vaizduoja nagrinėjamų trijų precedentų sprendimus. Tokiu būdu šis grafas vaizduoja teoriją matematinės logikos prasme.

Toliau siūlomas kitas modelis šiems trims precedentams modeliuoti. Šiame modelyje įvesime vertybės sąvoką. Vertybės (teisės prasme) bus siejamos su faktoriais ir jiems suteiks tam tikrą svorį. Taigi išskirkime tris vertybes:

V_1 – teisinis tikrumas (angl. *certainty*). Trupiname *tikrumas*.

V_2 – nauda, kuri yra suprantama kaip bendrojo vidaus produkto (BVP) didėjimas (angl. *promotion of gross national product* (GNP) arba *social utility*). Trumpiname *nauda*.

V_3 – nuosavybės neliečiamumas, šventumas (angl. *sanctity of property*). Trumpiname *nuosavybė*.



Pav. 13.12 Vertybių susiejimas su faktoriais

Tiesinio stiprumo susiejimo galima interpretuoti taip: „jeigu nepagavome, tai ir viskas“.

Nauda suprantama kaip socialinė nauda, BVP (bendro vidaus produkto) didinimas. Pavyzdžiui, Keeble vs. Hickeringill byloje, vienas didino BVP, o kitas - ne. Todėl ir buvo nuspręsta Keeble naudai.

Norėtume sutvarkyti vertybes tokia tvarka: $V_3 > V_2 > V_1$. Kaip žinoma iš kombinatorikos trijų vertybių sutvarkymų iš viso yra 3!=6. Iš pradžių pagrįsime, kodėl pasirinkome $V_2 > V_1$. Tuo tarpu nemotyvuosime $V_3 > V_2$, nes tai nebus aktualu mūsų išsikeltame uždavinyje.

Pabandykime įrodyti $V_2 > V_1$.

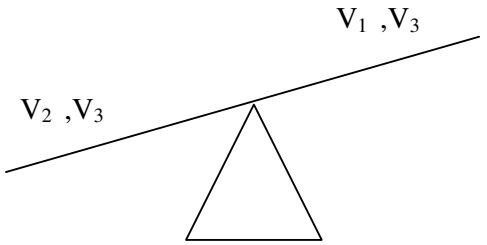
Nagrinėkime Keeble'o bylą (žr. Pav. 13.13). Teismas priimdamas savo sprendimą rėmėsi faktoriumi A, tačiau net nepaminėjo, faktorių B ir C. Primename, kad mūsų apsibrėžti faktoriai A, B remia ieškovą, o faktoriai C, D, E remia atsakovą. Keeble'o bylą galime apibrėžti lentele (žr. Pav. 13.13). Čia vertybė V_3 yra tiek ieškovo tiek atsakovo pusėje, todėl galima manyti, kad jos viena kitą nusveria. Lieka tik vertybės V_2 ir V_1, kaip jau minėjome šios vertybės teismas nepaminėjo, todėl galima tvirtinti, kad ji pasirodė nesvarbi.

Veikiantys faktoriai (juos apimančios vertybės)				
A (V_2)	B (V_3)	C (V_1, V_3)	D (V_2)	E (V_3)
LIVELIHOOD	OWNLAND	NOTCAUGHT	–	–

Pav. 13.13 Byla Keeble vs. Hickeringill „antys nuosavoje kūdroje"

Iš šio samprotavimo ir išplaukia, kad $V_2 > V_1$.

Į tai galime pažvelgti ir kitaip: surašę kuri pusė, kurias vertybes turi. Šiuo atveju ieškovas turėjo A ir B, taigi V_2 ir V_3 O atsakovas turėjo Cm taigi V_1 ir V_3. Kadangi abi pusės turėjoV_3. O teismas nusprendė ieškovo naudai, tai matome, kad $V_2 > V_1$.



Pav. 13.14 Ieškovas turėjo V2 ir V3, o atsakovas turėjo V1 ir V3

Norėtume iškelti klausimą, ar turėdami tiek duomenų galime išspręsti Young vs. Hitchens bylą? Atsakymas Ne.

Pabandykime pagrįsti savo atsakymą. Šis precedentas pavaizduotas Pav. 13.15.

Veikiantys faktoriai (juos apimančios vertybės)				
A (V_2)	B (V_3)	C (V_1, V_3)	D (V_2)	E (V_3)
LIVELIHOOD	–	NOTCAUGHT	COMPETE	OPEN

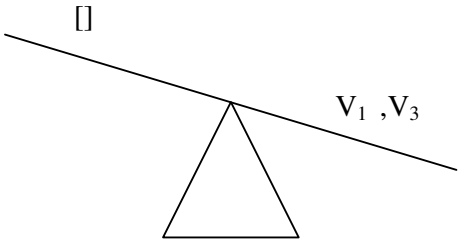
Pav. 13.15 Byla Young vs. Hitchens „Žuvys jūroje"

Šioje byloje, jau nebegalime remtis V_2 vertybe, nes tiek ieškovas, tiek atsakovas ją turi. Be to, šioje byloje ieškovas neturi V_3 vertybės (neturi B faktoriaus), o atsakovas turi V_1 (turi C faktorių), V_2 (turi D faktorių) bei V_3 (turi C ir E faktorių). Todėl neišeina taikyti Keeble'o precedento, o tenka taikyti Pierson'o precedentą. Nes jei įsivaizduotume, kad tiems faktoriams, kurie ieškovo pusėje ir atsakovo pusėje vienas kitą nusveria, įrašytume minusus (A ir D) – tai mūsų bylos faktoriai visiškai atitiktų Pierson'o bylą (žr. Pav. 13.17).

Pierson'o precedentas (Pav. 13.16) buvo išspręstas atsakovo naudai. Pagal vertybes [] < V_1, V_3 (žr. Pav. 13.17). Atkreipsime dėmesį, kad V_3 turime ir iš faktoriaus C ir iš faktoriaus E. Tačiau mums svarbu, kad tiesiog jį turime, t.y. $V_3 + V_3 = V_3$.

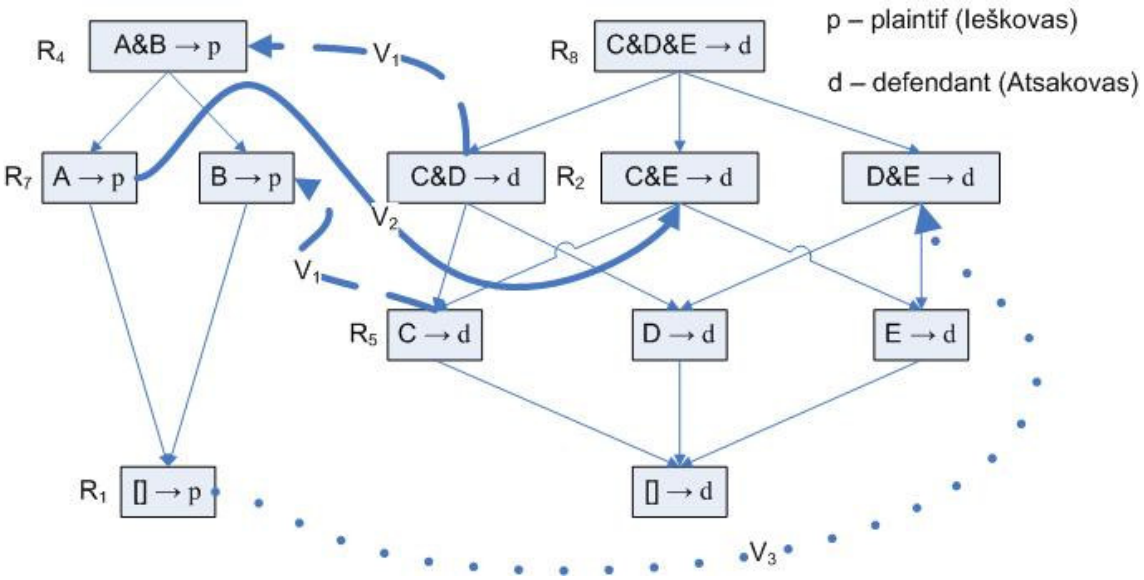
Veikiantys faktoriai (juos apimančios vertybės)				
A (V_2)	B (V_3)	C (V_1, V_3)	D (V_2)	E (V_3)
–	–	NOTCAUGHT	–	OPEN

Pav. 13.16 Byla Pierson vs. Post „lapė atviroje vietovėje“.



Pav. 13.17 Ieškovas nieko neturėjo, o atsakovas turėjo V1 ir V3

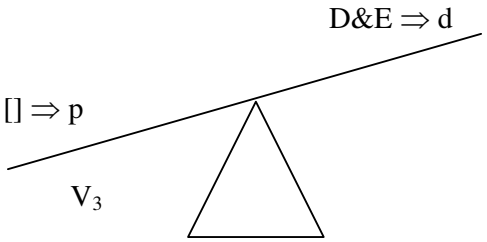
Pabandykime modifikuoti teorijos grafą (gardelę) įvesdami vertybių briaunas kaip pavaizduota Pav. 13.18.



Pav. 13.18 Kitas grafas, vaizduojantis kitą tris precedentus pagrindžiančią teoriją.

Dabar trumpai aptarsime briaunas, kurias įvardinome vertybėmis (V_1, V_2, V_3):

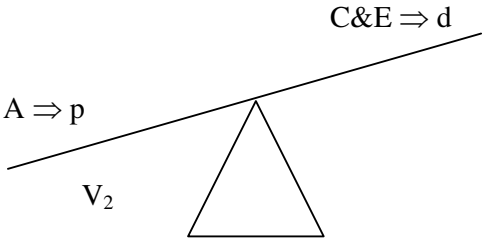
1. (Briauna V_3). []→p > D(V_2) & E(V_3)→d. Kai nėra faktoriaus C, tai nėra jokių klausimų dėl tikrumo V_1, nes ieškovas tikrai pagavo. Faktorius D reiškia, kad jie buvo konkurentai, todėl BVP nepriklauso nuo to, kas iš verslininkų pagavo objektą (čia gali būti antis, lapė ar pan.). Taigi pagal naudos principą V_2, jie yra lygūs. Taigi manome, kad ieškovas pranašesnis, nes jis pagavo, tai jo ir nuosavybė V_3, ir nėra jokių prieštaravimų kitų vertybių atžvilgiu. Taigi, pranašumą turi ieškovas.



Pav. 13.19 Briauna V_3

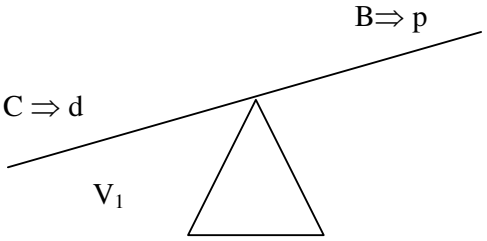
Atkreipsime dėmesį, jog iš vertybių prizmės žiūrint, faktoriai D ir E jau nebeatrodo pro-atsakovo.

2. (Briauna V_2). $A(V_2) \rightarrow p > C(V_1,V_3)$ & $E(V_3) \rightarrow d$. Čia yra faktorius C, taigi ieškovas neturi tikrumo V_3. Lyginsime pagal naudos principą V_2, nes jau žinome, kad $V_2 > V_1$. Ieškovas turi faktorių A, t.y. jis tai daro dėl pragyvenimo. O atsakovas neturi faktoriaus D, kas reiškia, jog jis tai daro dėl pramogos. Taigi V_2 (BVP) atžvilgiu pranašumą turi ieškovas.



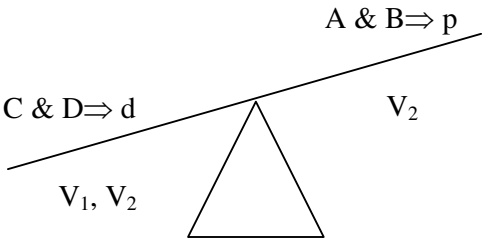
Pav. 13.20 Briauna V_2

3. (Briauna V_1). C $(V_1,V_3) \rightarrow d > B(V_3) \rightarrow p$. Čia vertybė V_2 yra neliečiama (nėra A). Laimi V_1 – teisinis tikrumas. Tačiau norisi pažymėti, kad tokio rezultato nebūtų, jei būtų koks nors pažeidimo faktorius (angl. *trespass*). Reziumuojant C laimi, kai nėra A.



Pav. 13.21 Briauna V_1

4. (Briauna V_1). C (V_1, V_3) & D (V_2) → d > A (V_2) & B (V_3) → p. Vėl tas atvejis, kai pagal vertybę V_2 – visi lygūs, taigi pagal V_1 iš faktoriaus C, laimi atsakovas.



Pav. 13.22 Briauna V_1

Štai viena iš interpretacijų, kuri sumenkina B ir E ryšį su vertybe V_3. T.y. nuosava žemė dar nesuteikia teisės į laukinį gyvūną, užklydusį į šią žemę.

Bandykime paaiškinti, kodėl mūsų grafas yra teorija matematinės logikos prasme. Vėl grįžkime prie mūsų precedentų.

1. Pierson'o precedentas, matematiškai turime (C&E): Pagal mūsų teoriją jį apsprendžia (C→d) > ([]→p), į E galima net neatsižvelgti. Žiūrime į gardelę Pav. 13.23. Pagal V_1 briauną žinome, kad (3) C→d > B→p, tuo tarpu pagal tą pačią gardelę matome, kad (C→d) < (C&E→d), o (B→p > []→p). Taigi gauname (C&E→d) > (C→d) > (B→p) > ([]→p) ⇒ (C&E→d) > ([]→p). O tai ir yra mūsų bylos sprendimas.
2. Keeble'o precedentas, matematiškai turime (A&B&C). Pagal mūsų teoriją, jį apsprendžia V_2 briauna (A→p) > (C&E→d). Kaip jau minėjome turime (A&B→p) bei (C→d). Žiūrime į gardelę Pav. 13.23. Sakėme, kad pagal briauną V_2 turime (A→p) > (C&E →d), tuo tarpu, pagal tą pačią gardelę (A→p) < (A&B→p), bei (C→d) < (C&E→d), todėl (A&B→p) > (A→p) > (C&E→d) > (C→d) ⇒ (A&B→p) > (C→d). Toks ir yra teismo sprendimas.
3. Young'o precedentas, matematiškai turime (A&C&D&E). Pagal mūsų teoriją, jį apsprendžia V1 briauna (C&D→d) > (A&B→p). Kaip jau minėjome turime (A→p) bei (C&D&E→d). Žiūrime į gardelę Pav. 13.23. Pagal briauną V_1 turime (4) (C&D→d) > (A&B→p). Taigi turime (C&D&E→d) > (C&D→d) > (A&B→p) > (A→p) ⇒ (C&D&E→d) > (A→p). Toks ir yra teismo sprendimas.

Validavome, kad mūsų teorija veikia.

Dar toliau įsivaizduokime tokį naują precedentą. Mes vaikštote po *savo* valdą, ančių kūdrą, į ją ateina profesionalus medžiotojas ir nušauna antį.

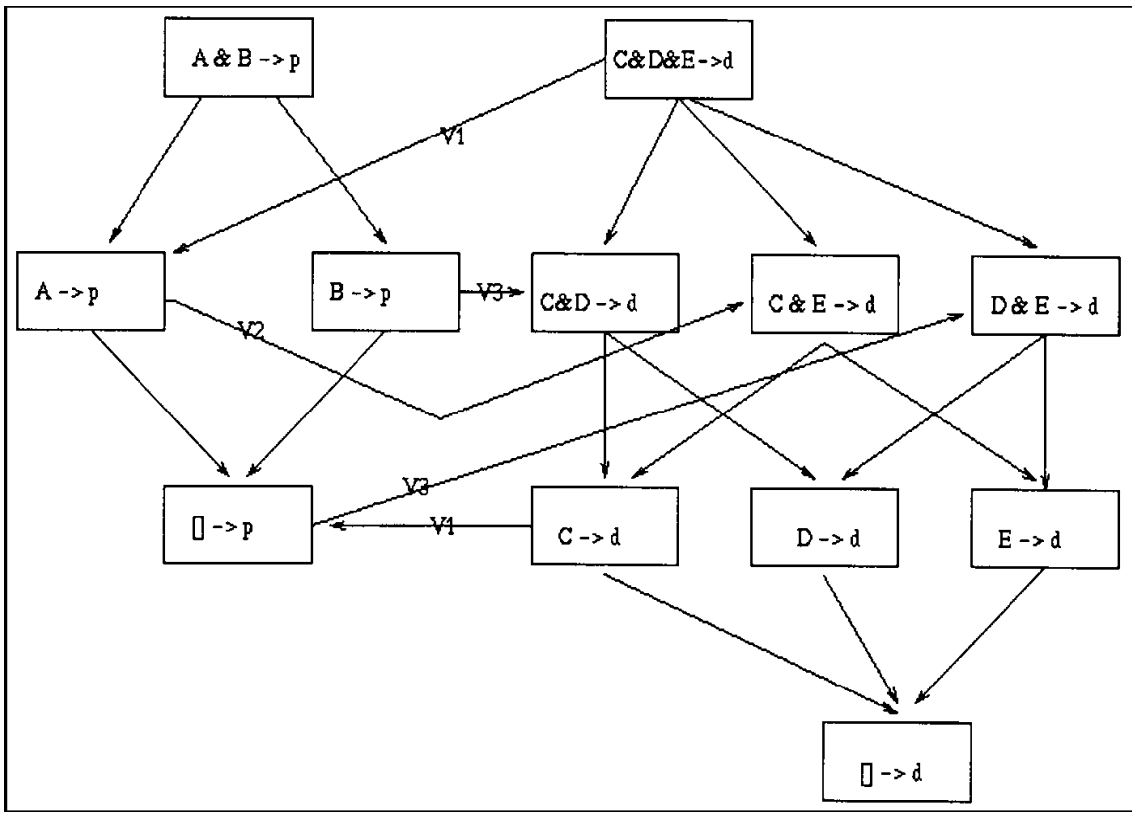
Veikiantys faktoriai (juos apimančios vertybės)				
A (V_2)	B (V_3)	C (V_1, V_3)	D (V_2)	E (V_3)
–	OWNLAND	NOTCAUGHT	COMPETE	–

Pav. 13.23 Ketvirtasis hipotetinis precedentas „antis savo žemėje"

Pagal mūsų teoriją gausime, kad turimi duomenys matematiškai (B&C&D). Todėl mums reikia išsiaiškinti ar (B→p) > (C&D→d) ar atvirkščiai. Tačiau čia matome pagal V_1 briauną (3), kad (C→d) > (B→p).

Tačiau iš tikrųjų teismas nuspręstų ieškovo naudai. Taip yra todėl, kad mes sumenkinome B ir E ryšį su vertybe V_3. Tačiau jeigu paimti B ryšį kaip pro-ieškovo, kuris skatintu vertybę V_3, tai kadangi $V_3 > V_2$, tai $(B \rightarrow p) > (C\&D \rightarrow d)$. Mes tarėme, kad $V_3 > V_2$, todėl kad gyvūno pagavimas, taigi turėjimas kaip nuosavybės (V_3), yra svarbiau už BVP (V_2).

Remdamiesi šiais samprotavimais, galėtume sukonstruoti šiek tiek kitokią teoriją (Pav. 13.24). Abi šios teorijos yra pilnos, tačiau kuri iš jų teisinga, priklauso nuo to, ar B iš tikrųjų skatina V_3. Reikėtų atitinkamo (aukščiau pateiktos bylos) precedento, kad nuspręsti galutinai ir turėti išbaigtą teoriją.



Pav. 13.24 Teorija, nusprendus, jog B skatina V_3.

Taigi apžvelgiame precedentais grindžiamo samprotavimo pavyzdį, tas turi glaudų ryšį su žinių vaizdavimu. Čia buvo pabrėžta, kad reikia remtis vertybėmis. Pamtėme kaip galima konceptuolizuoti vertybes.Teisės esmė – vertybių sprendimas.

14. Žodynas

Lietuvių kalba	Anglų kalba	Vokiečių kalba	Prancūzų kalba
dirbtinis intelektas	artificial intelligence	künstliche Intelligenz	intelligence artificielle
užduotis	task		
žinių vaizdavimas	knowledge representation		
žinios	knowledge		

15. Egzamino klausimai

Pirmasis egzamino klausimas pateikiamas iš paskaitų medžiagos. Antrasis klausimas pateikiamas pagal [Brachman, Levesque 2004] knygą iš šių temų:

(1 skyriaus 1 poskyris) Esminės sąvokos: žinios, vaizdavimas ir samprotavimas.
(1sk. 2 posk.) Žinių vaizdavimo ir samprotavimo motyvacija.
(1 sk. 3 posk. ir 2 sk.) Logikos vaidmuo žinių vaizdavime. Pirmos eilės logika: sintaksė ir semantika.
(3 sk.) Žinių išreiškimas. Individualios sąvokos, atributai, savybės, ryšiai, sudėtiniai faktai, terminologiniai faktai, reifikavimas (*reification*) – predikato pavertimas objektu.
(4 sk.) Rezoliucija.
(7 sk.) Produkcijų taisyklės produkcijų sistemose.
(8 sk.) Objektiškai orientuotas vaizdavimas.
(9 sk.) Struktūrizuoti aprašymai. Deskriptyvinė logika (kokia logine kalba aprašyti), klasifikavimas(*is-a*, *instance-of*), paveldėjimas.
(10 sk.) Paveldėjimas. Perklojamas paveldėjimas (*defeasible inheritance*). Paveldėjimo hierarchija (*inheritance hierarchy*).

16. Literatūra

Pagrindinė literatūra

1. [Brachman, Levesque 2004] Ronald J. BRACHMAN, Hector J. LEVESQUE. Knowledge representation and reasoning. The Morgan Kaufmann Series in Artificial Intelligence, 2004. 381 p.
2. [Stefik 1995] Mark STEFIK. Introduction to knowledge systems. Morgan Kaufmann Publishers, 1995. 871 p.
3. [Sowa 2000] John F. SOWA. Knowledge representation: logical philosophical, and computational foundations. Brooks/Cole, a division of Thomson Learning. 2000. 573 p.
4. [Russell, Norvig 2003] Stuart RUSSELL, Peter NORVIG. Artificial intelligence: a modern approach. Second edition, Prentice Hall, 2003. 1132 p. http://aima.cs.berkeley.edu.

Papildoma literatūra

5. [Baronas 2002] Romas BARONAS. Duomenų bazių valdymo sistemos. Vadovėlis aukštosioms mokykloms. Vilnius: TEV, 2005. 184 p. Prieiga per internetą http://www.mif.vu.lt/~baronas/dbvs/book/.
6. [Baniulis, Tamulynas 2003] Kazys BANIULIS, Bronius TAMULYNAS. Duomenų struktūros. Kaunas: Technologija, 2003. 283 p.
7. [Luger 2005]. George LUGER. Artificial Intelligence: Structures and Strategies for Complex Problem Solving (Fifth Edition), Addison-Wesley, 2005, 928 p. Prieiga per internetą http://www.cs.unm.edu/~luger/.
8. [Nilsson 1982] Nils J. NILSSON. Principles of artificial intelligence. Tioga Publishing, Palo Alto, CA, 1980; Springer-Verlag, Berlin, 1982, 476 p. Vertimas į rusų kalbą: Н. Нильсон. Принципы искусственного интеллекта: Перевод с англ. – Москва: Радио и связь, 1985. – 376 с.
9. [Nilsson 1998] Nils J. NILSSON. Artificial Intelligence: a new synthesis. Morgan Kaufmann Publishers, 1998 513 p.
10. [Čyras 2006] Vytautas ČYRAS. Dirbtinis intelektas. Elektroninė knyga, 86 p. Prieiga internete http://www.mif.vu.lt/katedros/se/veikla/konspektai-DI-Cyras-2006-01-10.pdf

Kita naudota arba cituota literatūra

11. [Ackoff 1989] R. L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, v. 16, p. 3-9.
12. [Bellinger, Castro, Mills 2004] Gene Bellinger, Durval Castro, Anthony Mills. Data, information, knowledge and wisdom. [Interaktyvus, žiūrėta 2006-11-05] http://www.systems-thinking.org/dikw/dikw.htm.
13. [Bench-Capon 2002] Bench-Capon T. J. M. (2002). The missing link revisited: the role of teleology in representing legal argument. *Artificial Intelligence and Law*, v. 10, p. 76-94.
14. [Buchanan et al. 1990] Bruce G. Buchanan, Daniel Bobrow, Randall Davis, John McDermott, Edward H. Shortliffe. Knowledge-based systems. *Annu. Rev. Comput. Sci.* 1990, v. 4, p. 395-416 http://arjournals.annualreviews.org/action/showJournals

15. [Chandrasekaran et al. 1992] B. Chandrasekaran, T. Johnson, J. Smith. A task-structure analysis for knowledge modeling. *Communications of the ACM*, v. 35, n. 9, p. 124-137.
16. [Chen 1976] Peter Pin-Shan CHEN (1976). The entity-relationship model – towards a unified view of data. ACM Transactions on Database Systems (TODS), v. 1, n. 1, p. 9-36. http://portal.acm.org/citation.cfm?id=320434.320440&dl=portal&dl=ACM&idx=J777&part=periodical&WantType=periodical&title=ACM%20Transactions%20on%20Database%20Systems%20%28TODS%29&CFID=3314899&CFTOKEN=29141759
17. [Ernst, Newell 1969] G. W. Ernst, A. Newell. GPS: A case study in generality and problem solving. New York: Academic Press.
18. [Filosofija WWW] Filosofijos žodynas (interaktyvus). [Žiūrėta 2006 m. spalio 29 d.]. Prieiga per internetą http://filo.web1000.com/texts/zodynas/raides/i.htm.
19. [Geldenhuys 1999] Geldenhuys A.E., van Rooyen H. O., Stetter F. (1999). *Knowledge representation and relation nets*. Kluwer Academic Publishers, Boston, 279 p.
20. [Gruber 1993] T. J. GRUBER. A translation approach to portable ontologies. *Knowledge Acquisition*, 592): 1999-220, 1993. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
21. [Hornby 1980] A. S. HORNBY. *Oxford Advanced Learner's Dictionary of Current English*. Twelfth impression. Oxford University Press, 1980, 1037 p.
22. [Marcijonas, Sudavičius 2003] Antanas MARCIJONAS, Bronius SUDAVIČIUS. Mokesčių teisė. – Vilnius: Teisinės informacijos centras, 2003. – 288 p.
23. [MAĮ] Lietuvos Respublikos mokesčių administravimo įstatymas, 2004 m. balandžio 13 d. Nr. IX-2112, Žin., 2004, Nr. 63-2243.
24. [Valente 1995] A. Valente. *Legal knowledge engineering: A modelling approach*. IOS Press, Amsterdam, 217 p.