Uz nodaļas satura rādītāju

Aprakstošās loģikas un secinātāji: ievads

Kārlis Podnieks, LU

Literatūra:

Šī kursa grāmata:

The Description Logic Handbook, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, 2007

LU bibliotēkā ir viens šīs grāmatas eksemplārs.

1.1. Ko vēl lasīt...

Kursa grāmatas pirmās nodaļas

D. Nardi, R. J. Brachman. An Introduction to Description Logics (ap 40 lpp.)

elektronisko kopiju var atrast **Enrico Franconi** (Free University of Bozen-Bolzano asociētā profesora) tīmekļa vietnē. Viņš lasa kursu Description Logic. Ja kopijas adreses beigās skaitlisko daļu maina no nulle-viens līdz viens-seši, tad tā var ielūkoties grāmatas visās sešpadsmit nodaļās.

Pavisam īss ievads un pārskats: Description Logic Wikipedia.

Alternatīvas (ļoti labas!) ievadlekcijas: **Enrico Franconi**, Free University of Bozen-Bolzano, Description Logics, sk, otro moduli (lekcija 1, lekcija 2).

Nopietnāku ievadu – no "augstākām pozīcijām" piedāvā **Hector Levesque,** University of Toronto, Knowledge Representation and Reasoning (tie ir publiski pieejami lekciju slaidi, balstīti uz viņa un **R. J. Brachman** tāda paša nosaukuma grāmatas).

Matemātiskā orientētiem cilvēkiem, droši vien, būtu patīkamak studēt **Stephan Tobies** disertācijas tekstu adresē http://citeseer.ist.psu.edu/tobies01complexity.html, sadaļas 1.1-1.4.

Ļoti labas ir arī attiecīgās nodaļas **Carlos Areces** disertācijā, 2000.

1.2. Zināšanu formalizācija, izmantojot predikātu loģiku

Vispirms mēģināsim nonākt pie aprakstošo loģiku jēdziena, ejot "no augšas".

"Parastās" datubāzes

Piemērs. Relāciju datubāze CILVEKI, tikai viena tabula:

CILVEKS(*vards, dzimums, teva-vards, mates-vards*).

Vārdus uzskatīsim par atslēgām (t.i. identifikatoriem).

Šajā datubāzē tiešā veidā ir reģistrēti tikai cilvēku tiešie senči, t.i. vecāki. Bet no šiem datiem, ja tie ir pietiekami pilnīgi, var **izsecināt** arī citu informāciju: kas ir dotā cilvēka vecvecāki (un vēl tālāki senči), cik cilvēkam ir bērnu (un mazbērnu), vai diviem dotajiem cilvēkiem ir kopīgs sencis, citi kopīgi radinieki, iespējamie interešu konflikti utt.

Bet kā šo "izsecināmo" informāciju no mūsu datubāzes var iegūt?

Uzdevums. Uzrakstiet SQL vaicājumus, kas cilvēkam vārdā *John* atrod: 1) abus vectēvus (visas māsas, visus brālēnus utt.); 2) bērnu skaitu (mazbērnu skaitu utt.).

Relāciju datubāzēs tātad atvasinātās informācijas iegūšanas (**“secināšanas”!**) līdzeklis ir SQL vaicājumi.

Šeit visus izmantotos jēdzienus (vectēvs, māsa, brālēns utt.) nākas "definēt" tieši pašās meklēšanas programmās. Nav pārliecības, ka visas šīs "definīcijas" uzreiz (bez ilgāka skaņošanas procesa) būs korektas, un ka tās visos gadījumos (dažādu autoru programmās) būs vienādas.

Deduktīvās datubāzes / Zināšanu bāzes

Kas tad ir "atvasināta informācija"?

Visvispārīgākajā veidā tā ir informācija, kas **seko** no tiem datiem, kas glabājas datubāzē. Tātad atvasinātas informācijas iegūšana (visvispārīgākajā

gadījumā) ir **secināšana** (*reasoning*).

Bet ja tā, tad būtu dabiski datubāzē visu informāciju attēlot ar loģikas formulu palīdzību un secināšanai izmantot loģikas aksiomas un izveduma likumus.

Tieši tā tiek būvētas t.s. **deduktīvās datubāzes, vai zināšanu bāzes**. Šajās bāzēs līdz ar **faktiem** (parastu datubāzu informāciju, *ground axioms*) var reģistrēt arī **likumus** (*deductive axioms*), kurus izmantojot, no vieniem faktiem izvest citus faktus, t.sk. tādus, kuri tiešā veidā bāzē nav reģistrēti.

Piemērs. Tos pašus datus, kas glabājas datubāzē CILVEKI, var attēlot formālā valodā (predikātu valodā), kurā:

1) cilvēku vārdi ir **objektu konstantes**, kas šos cilvēkus apzīmē (piemēram, *John, Paris*), tāda konstante ir ievesta katram cilvēkam, kura informācija glabājas datubāzē;

2) ir **mainīgie** x, y, z, ..., kuru vērtību apgabals ir "visi cilvēki, kuru informācija glabājas datubāzē";

3) cilvēka x dzimumu uzdod divas **predikātu konstantes**: S(x) − "x ir sieviete", V(x) − "x ir vīrietis"; (**vīriešu klase un sieviešu klase)**

4) tēva un mātes informāciju uzdod **predikātu konstantes**: M(x, y) − "x ir y-a māte", T(x, y) − "x ir y-a tēvs". (**2 asociācijas**)

5) vienādības predikāts x=y.

Mūsu bāzē tad (kā **aksiomas**) glabātos **fakti**:

V(John), S(Paris), M(Paris, Peter), T(Peter, John), ...

Šī informācija, protams, ir līdzvērtīga datubāzes CILVEKI informācijai.

Bet līdz ar faktiem mēs savā bāzē varam reģistrēt **atvasinātu jēdzienu (klašu) definīcijas**, piemēram:

$$VECTEVS(x,y) \equiv \exists z\,[T(x,z) \wedge (M(z,y) \vee T(z,y))]$$

Arī datubāzes **integritātes nosacījumus** mēs varam glabāt bāzē kā formulas (**likumus, aksiomas**), piemēram:

$\forall x \forall y [M(x, y) \rightarrow S(x)]$;

$\forall x \forall y [T(x, y) \rightarrow V(x)]$;

$\forall x [S(x) \vee V(x)]$;

$\forall x [\neg (S(x) \wedge V(x))]$;

$\forall x_1, x_2, z [T(x_1, z) \wedge T(x_2, z) \rightarrow x_1 = x_2 \wedge \neg (x_1 = z)]$;

$\forall x_1, x_2, z [M(x_1, z) \wedge M(x_2, z) \rightarrow x_1 = x_2 \wedge \neg (x_1 = z)]$;

utt.

Šādi likumi ir atklāti jānoformulē un jāievada zināšanu bāzē (kā **aksiomas**), jo no datora programmatūras viedokļa T, M, S, V ir patvaļīgas attiecīgi 2-vietīgas un patvaļīgas 1-vietīgas predikātu konstantes., bet jebkādām citām "noklusētām" īpašībām.

Reiz nodefinējuši, mēs jaunos atvasinātos jēdzienus (klases) varam izmantot vēl sarežģītāku jēdzienu (klašu) definēšanai un aksiomu formulēšanai.

Vaicājumi, piemēram, valodas *Prolog* stilā:

?VECTEVS(x, John) – sameklēt visus *John*-a vectēvus (t.i. visas objektu konstantes c, kam formula VECTEVS(c, John) seko no zināšanu bāzē esošajām formulām;

$?\forall x \exists y VECĀKS(y, x)$ - vai šī formula seko no zināšanu bāzē esošajām formulām?

Mūsu jaunajai (zināšanu!) bāzei, ar šādiem vaicājumiem būtu jāprot tikt galā.

Uzdevums. a) Kā definēt bērna, mazbērna, brāļa, māsas, brālēna utt. jēdzienus?
b) Kā definēt jēdzienu SENCIS(x, y) − "x ir y-a sencis"? Vaicājums: ?SENCIS(x, John) – sameklēt visus *John*-a senčus.

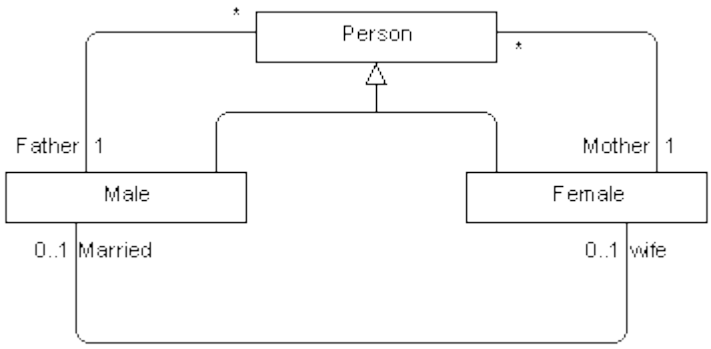
Uzdevums. Pārdomājiet, kā šīs lietas varētu realizēt SQL un citās programmēšanas valodās? Vai tas sanāktu tikpat dabiski?

Ontoloģijas, UML un predikātu valodas

Loģikas formulu izmantošana datubāzēs pirmajā brīdī liekas neparasta. Daudz pierastāka ir ER-diagrammu vai UML klašu

diagrammu izmantošana datubāzes struktūras uzdošanai.

1.piemērs: cilvēki, tēvi, mātes, vīrieši un sievietes.



2.piemērs: kursi, studenti un pasniedzēji.

Bet katras šādas diagrammas precīzu jēgu var nodefinēt, izmantojot loģikas formulas. **Diagramma būtībā ir ekvivalenta noteiktai formulu kopai atbilstošā predikatu valodā.**

Daļa no formulām tiek attēlotas ar noteiktiem grafiskiem līdzekļiem (kastītes, līnijas, kardinalitātes, apakšklases, complete, disjoint utt.). Bet ne katru vajadzīgu formulu varēsim uzskatāmi attēlot grafiski. **Formulas ir universālāks līdzeklis, salīdzinot ar diagrammām!**

Tāpēc diagrammas nevajadzētu mistificēt. Tās cilvēkiem ir ļoti ērtas un ļauj ātri aptvert lietas būtību, jo cilvēka smadzenēs visjaudīgākais procesors ir t.s. **redzes centrs**. Vizuālu informāciju cilvēks apstrādā visefektīvāk. Toties datoram diagrammas neko nedod – to saturu nākas tulkot, piemēram, ar loģikas formulu palīdzību (vai kā savādāk, bet arī šeit formulas izrādās visuniversālākais līdzeklis).

Uzdevums. Uzzīmējiet nelielu UML klašu diagrammas piemēru un tad ievediet speciālu predikātu valodu un uzrakstiet šajā valodā formulu kopu, kas būtu ekvivalenta šai diagrammai. Sevišķi uzmanīgi reģistrējiet visus integritātes nosacījumus.

Atkārtojums

par predikātu (jeb pirmās pakāpes) valodām, interpretācijām,

modeļiem, loģiski vispārderīgām un izpildāmām formulām. Spēja ātri lasīt un saprast predikātu valodas formulas, ātri formulēt savas zināšanas predikātu valodā būs ļoti vēlama šī kursa sekmīgai apguvei.

Senči un rekursija

Jēdzienu SENCIS mūsu deduktīvajā datubāzē var ievest ar likumu palīdzību:

$$SENCIS(x, y) \leftarrow M(x, y) \vee T(x, y) \ ;$$
$$SENCIS(x, y) \leftarrow SENCIS(x, z) \wedge SENCIS(z, y) \ ;$$

(SENCIS ir "vecāku predikāta" $M(x, y) \vee T(x, y)$ **tranzitīvais slēgums**).

Valodas SQL standartā, sākot ar 1999.gada versiju, jau tiek atbalstīti rekursīvi vaicajumi (*recursive queries*). Ar to palīdzību var realizēt vaicājumus, kas saistīti ar senča jēdzienu. [Vai Oracle, MS SQL Server, MySQL tas ir realizēts?]

Pirmās pakāpes valodas ir pārāk vājas, lai **definētu senča jēdzienu ar vienas formulas palīdzību!**

Precīzu definīciju, ko tas nozīmē, un elegantu šīs teorēmas pierādījumu sk. **Carlos Areces** disertācijā, 2000.

[Ko tad tas precīzi nozīmē? T.i. ko nozīmētu izteikt senča jēdzienu ar vienu formulu? Tas nozīmē, ka, izmantojot kā atomus tikai predikātu konstantes T(x, y), M(x, y), konjunkcijas, disjunkcijas, negācijas, implikācijas un kvantorus, mums vajadzētu uzrakstīt (varbūt, sarežģītu, bet **vienu**) formulu F(x, y), kurai būtu šāda īpašība: jebkurā valodas interpretācijā jebkuriem objektiem c, d, formula F(c, d) ir patiesa tad un tikai tad, ja šajā interpretācijā c ir d sencis (šī vārda parastajā nozīmē). Tā to formulē *Carlos Areces*, jo viņš interesējas par abstraktāku problēmu – par patvaļīgas relācijas R *tranzitīvā slēguma* definējamību. Mūsu gadījumā interepretācijām būtu jāuzliek papildus prasība: tajās nedrīkst būt ciklu – "pats neesmu sev sencis". Vai *Areces* aprakstītais elegantais pierādījums ir pielāgojams arī šai situācijai? Droši vien, ir...]

Zināšanu bāze vispārīgajā gadījumā

Noteikta predikātu valoda (mainīgie, objektu konstantes, vienvietīgi un divvietīgi predikāti, loģikas saikļi un kvantori).

Atvasinātu jēdzienu (klašu) definīcijas.

Aksiomas-fakti (A-kaste).

Aksiomas-likumi (T-kaste).

Vaicājumi un secināšanas uzdevums.

Ja mēs savu **zināšanu bāzi** B veidosim, izmantojot predikātu valodu (pirmās pakāpes valodu), tad vispārīgā gadījumā situācija būs šāda:

a) Bāzē ir reģistrētas noteiktas objektu konstantes: c_1, c_2, ..., c_k, un predikātu konstantes: p_1, p_2, ..., p_m (pēdējam katrai ir uzdots savs argumentu skaits).

b) Bāzē glabājas fakti, likumi un jēdzienu definīcijas, t.i. noteiktas formulas F_1, F_2, ..., F_n. Fakti ir atomāras formulas, ar vai bez negācijas, kas nesatur mainīgos: $p_i(c_{j1}, c_{j2}, ..., c_{js})$, vai $\sim p_i(c_{j1}, c_{j2}, ..., c_{js})$.

Būtībā fakti veido datubāzes tabulas. Bet likumi ir integritātes nosacījumi.

c) **Vaicājums** ir kāda cita formula G. Atbildēt uz šādu vaicājumu nozīmē noskaidrot, vai formula G (vai, varbūt, ¬G?) **seko** formulām F_1, F_2, ..., F_n, t.i. no zināšanu bāzes faktiem un likumiem.

Ja formulai G ir brīvs mainīgais x, tad vaicājumu G(x) mēs saprotam šādi: atrast visas tās konstantes c_i, kam formula $G(c_i)$ seko no F_1, F_2, ..., F_n.

Vaicājumā var būt arī vairāki brīvi mainīgie.

Ko šeit nozīmē "seko"? Ja domā "semantiski", tad tad "**formula G seko no formulām F_1, F_2, ..., F_n**" nozīmē, ka "jebkurā valodas interpretācijā, kurā patiesas ir formulas F_1, F_2, ..., F_n, ir patiesa arī formula G". Bet tā kā mēs zinām Gēdela teorēmu par pilnību, tad zinām arī, ka šis formulējums ir ekvivalents ar šādu: "formula G ir

izvedama no formulām F_1, F_2, ..., F_n, izmantojot predikātu loģikas aksiomas un izveduma likumus".

Tātad, veidojot savas zināšanu bāzes programmatūru, mums būs jāizmanto **algoritms**, kas, ja dotas formulas F_1, F_2, ..., F_n, G, var (vēlams, ātri) pateikt, vai G seko no F_1, F_2, ..., F_n, vai neseko. Sauksim to par **formulu secināšanas uzdevumu**.

Diemžēl, jau no 1936.gada ir zināma:

Čērča-Kalmāra teorēma. Ja kādā predikātu valodā ir vismaz viena vismaz divvietīga predikātu konstante, tad formulu secināšanas uzdevums šai valodai nav algoritmiski atrisināms.

Alonzo Church, **Laszlo Kalmar**

Elegantu šīs teorēmas (gan mazliet vājākā formā) pierādījumu sk. Carlos Areces disertācijā, 2000.

Secinājums. Ja zināšanu bāzes būvēšanai mēs pilnā apjomā izmantosim kādu predikātu valodu (ar vismaz vienu vismaz divvietīgu predikātu konstanti), tad mēs šai bāzei **nevarēsim izveidot universālu vaicājumu procesoru.**

Tātad mūsu zināšanu bāzē pieļaujamie predikātu valodu līdzekļi ir jāierobežo tā, lai secināšanas uzdevums būtu algoritmiski atrisināms. Tikai tad varam cerēt izveidot universālu vaicājumu procesoru.

Ko šādā situācijā darīt?

Zināšanu bāzēs predikātu loģiku pilnā apjomā izmantot nevar, jo tad formulu secināšanas uzdevums kļūst algoritmiski neatrisināms.

Aprakstošās loģikas (*description logics*) ir šobrīd populārākais veids kā, saglabājot tomēr pietiekamu formālās valodas izteiksmes iespēju minimumu, ierobežot valodas līdzekļus tā, lai formulu secināšanas uzdevums būtu algoritmiski atrisināms.

Bet tas nav vienīgais iespējamais veids. Par tādām lietām interesējies jau viens no matemātiskās loģikas klasiķiem:

W. Ackermann. Solvable cases of the decision problem. North-Holland, Amsterdam, 1954.

Ļoti vienkāršu un samērā vispārīgu ideju – t.s. **guarded fragment** (vai **guarded logic**) piedāvā

H. Andreka, J. van Benthem, I. Nemeti. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3): 217-274, 1998.

Sk. arī http://en.wikipedia.org/wiki/Guarded_logic.

Guarded logic atzīst tikai īpašā veidā ierobežotus kvantorus, piemēram (p – jebkura divvietīga predikātu konstante, F – formula):

$$\exists y\left(p\left(x,y\right)\wedge F\left(x,y\right)\right)\text{ , jeb}$$

$$G\left(x\right)\equiv\left(\exists y:p\left(x,y\right)\right)F\left(x,y\right)\text{ ;}$$

$$\forall y\left(p\left(x,y\right)\rightarrow F\left(x,y\right)\right)\text{ , jeb}$$

$$G\left(x\right)\equiv\left(\forall y:p\left(x,y\right)\right)F\left(x,y\right)\text{ .}$$

Šeit tiek prasīts, lai formulā F(x, y) nebūtu citi brīvi mainīgie kā formulā p(x,y), t.i. tikai x, y. Kvantora " visiem y" darbības apgabals te ir nevis "visa pasaule", bet tikai tie objekti y, kas ir fiksētā veidā (p ir predikātu konstante!) saistīti ar tiem objektiem x, par kuriem ir runa formulā F(x,y).

[Te nav jābaidās, ka p(x, y) varētu būt identiski patiess (kas varētu izjaukt guard-ēšanu). Divvietīgam predikātam guarded valodā nemaz nevar pateikt, ka $\forall x\forall y\,p\left(x,y\right)$.]

Formulu secināšanas uzdevums *guarded logic* ir algoritmiski atrisināms – tas ir EXPTIME-complete (fiksētai valodai), t.i. ja iesaistīto formulu kopgarums ir N, tad secināšanas uzdevums prasa laiku 2^{CN}.

Protege

Visus eksperimentus ar zināšanu bāzēm, ontoloģijām un secinātājiem mēs veiksim, izmantojot rīku **Protege**, kura autori to pozicionē kā "*free, open source ontology editor and knowledge-base framework*".

No adreses http://protege.stanford.edu var lejupielādēt vairākas

Protege versijas.

Vizualizators no LU MII: http://owlgred.lumii.lv/

1.3. Aprakstošā loģika ALC

ALC ir saīsinājums no "Attribute Logic with Complement".

"Cilvēkiem no loģikas" parasti skaidro tā: ja aplūkojam *guarded logic* vēl vienkāršāku gadījumu – kad formulā F ir tikai viens brīvs mainīgais y:

$$G_1(x) \equiv \exists y\,(p(x,y) \wedge F(y)),$$

"Klase G sastāv no tiem x, kam eksistē p-saistīts y klasē F"

$$G_2(x) \equiv \forall y\,(p(x,y) \rightarrow F(y)),$$

"Klase G sastāv no tiem x, kam visi p-saistītie y ir klasē F"

tad mēs nonākam jau pie klasiskās aprakstošās loģikas, t.s. loģikas **ALC**, kurai (kā vēlāk redzēsim) formulu secināšanas uzdevums ir PSPACE-complete.

Piemērs: p(x, y) = "x is parent of y"; F(y)="y ir aristokrāts";
G_1(x)="x-am vismaz viens no vecākiem ir aristokrāts";
G_2(x)="x-am abi vecāki ir aristokrāti".

Piezīme. **Kāpēc tik liela interese par PSPACE**, t.i. par iespēju risināt uzdevumu, izmantojot atmiņu, kuras apjoms nepārsniedz polinomu no ieejas datu garuma? Tāpēc, ka nākošais sarežģītības līmenis ir eksponenciāls atmiņas apjoms. Pie tam atmiņas apjomam (atšķirībā no rēķināšanas laika) ir nejauka īpašība – tas bieži ir vienāds gan "labajos", gan "sliktajos" gadījumos. [Rēķināšanaas laiks var būt nepieņemams "sliktajos" gadījumos, bet pieņemams - "prakstiskajos" gadījumos.] Tāpēc, ja mēs kādu uzdevumu neprotam risināt polinomiālā atmiņā, tad ir gandrīz droši, ka šis uzdevums būs datoriem "nepaceļams". Tāpēc arī tāda interese par uzdevumiem, kuri pieder sarežģītības klasei PSPACE.

Loģikas ALC elegantākā definīcija

Bet tikko minētais "skaidrojums" liekas pārāk tehnisks, lai noticētu, ka loģika ALC ir kaut kas vairāk nekā šauriem mērķiem domāts izgudrojums.

Manuprāt, esmu atradis daudz elegantāku veidu ALC ievešanai.

"Vīzijas" pamatā, tāpat kā predikātu valodām, ir "apgabals", kas sastāv no "indivīdiem", par kuriem interesējamies.

Piemēri: a) "visi cilvēki"; b) "kursi, pasniedzēji un studenti".

Indivīdiem ir piešķirti vārdi:

Piemēri: a) Jobn, Paris, Peter; b) Algebra, ASmith, BSmith.

Šajā apgabalā mūs interesē dažādas objektu kopas. Sauksim šīs

kopas par **klasēm** (**jēdzieniem** jeb **konceptiem**). Dažas no šīm klasēm mums ir definētas jau pašā sākumā kā konstantes *p, q,* utt. (pamatklases). Un mums ir zināma indivīdu piederība vai nepiederība šīm klasēm.

Piemēri: a) klases: *Sieviete* – sievietes, *Virietis* – vīrieši, *Cilveks* – visi cilvēki;
Paris: *Sieviete*, John: *Virietis*; Peter: *Virietis*.
b) *Kur*ss – kursi, *Pasniedzejs* – pasniedzēji, *Students* – studenti.
Algebra: *Kurss*, ASmith: *Pasniedzejs*; BSmith: *Students*.

Kā jau datubāzēs parasts, klašu nosaukumi ir vienskaitlī.

Pārējās klases vajadzēs atvasināt šīm pamatklasēm. Ar kādiem līdzekļiem?

Protams, vispirms prātā nāk tradicionālās **Būla operācijas** – šķēlums, apvienojums un papildinājums: ja mums jau ir izveidotas klases C, D, tad varam izveidot arī klases $C \cap D\,; C \cup D\,; \neg C$.

Bet starp mūsu objektiem varēs pastāvēt arī divvietīgas asociācijas. Tām jabūt definētām jau pašā sākumā kā konstantēm *r, s*, utt. (līdzekļus atvasinātu asociāciju veidošanai pagaidām neatļausim). Sauksim tās par **lomām** – ja objektus x un y saista asociācija *r*, tad rakstīsīm $x\ \boldsymbol{r}\ y$ un teiksim, ka y-am pie x-a ir loma r (*x*-a *r* ir *y*).

Piemēri: a) loma *berns*, rakstām: x *berns* y (lasām: x-a bērns y).
b) loma *maca*, rakstām*: x maca y* (lasām: [pasn.]x māca [kursu]y);
loma *stude*, rakstām: *x stude y* (lasā: [stud.]x studē [kursu]y.

Lomu vārdus vajag censties izvēlēties tā, lai lasot *x r y*, sanāktu kas labi un dabiski saprotams.

Tātad mūsu apgabals būtībā ir orientēts grafs, kura virsotnes ir objekti x, y, ..., un ja divus objektus x, y saista loma r, tad no x uz y tiek novilkta (orientēta!)

šķautne, kurai pierakstīts burts r.

Šādā grafā ir ieraugāms vēl vismaz viens dabisks veids jaunu klašu definēšanai. Ja mums jau ir definēta klase C, tad lomai r mēs varam aplūkot visu to grafa virsotņu kopu, **no** kurām **uz** klasi C ved šķautnes ar burtu r.

[Uzzīmējiet bildi, kas šo operāciju demonstrē vizuāli: vecāki → bērni.]

Kopu teorijā šo operāciju pieņemts saukt par **inverso attēlu** un apzīmēt ar $r^{-1}(C)$. Loģikas valodā:

$$r^{-1}(C)=\{x|\exists y(r(x,y)\wedge y\in C)\} .$$

Aprakstošo loģiku pasaulē šo operāciju apzīmē ar $\exists r.C$ – "visi tie objekti, kam eksistē r klasē C".

Piemēri: a) $\exists berns.Sieviete$ – klase "visi cilvēki, kam ir meitas".

b) $\exists maca.Kurss$ – klase "visi [pasniedzēji], kas māca kādu kursu".

Tad lūk, ja Būla operācijām pievienojam inversā attēla operāciju un aplūkojam visas klases, ko var nodefinēt, kombinējot šīs četras operācijas

$$C\cap D, C\cup D, \neg C, r^{-1}(C)$$

(sākumā mums ir tikai galīgs skaits pamatklašu p, q, utt. un galīgs skaits lomu r, s utt.), tad tā arī būs slavenās **aprakstošā loģikas ALC valoda**!

Ja pievienosim vēl vienu operāciju – kopas attēlu:

$$r''(C)=\{y|\exists x(x\in C\wedge r(x,y))\} ,$$

tad komplekts

$$C \cap D, C \cup D, \neg C, r^{-1}(C), r''(C)$$

dos mums **aprakstošo loģiku ALCI** (t.i. līdz ar lomu r mēs te drīkstam izmantot arī tās inverso lomu).

[Uzzīmējiet bildi, kas šo operāciju demonstrē vizuāli: vecāki → bērni.]

Tik eleganta formulējuma iespējamība liecina, ka aprakstošās loģikas nav tikai šauri tehnisks izgudrojums!

Piemērs. a) Apgabals – visi cilvēki. Pamatklases: *Sieviete, Virietis*. Viena *loma*: *berns*. Kādas klases vēl varam nodefinēt:

$\neg Sieviete$ ("vīrieši", jeb koncepts Virietis);

$Sieviete \cup \neg Sieviete$ ("cilvēki", jeb koncepts Cilveks),

$\exists berns.Sieviete$ ("cilvēki, kam ir meitas", jeb koncepts "cilvēks, kam ir meitas");

$\exists berns.Cilveks$ (kas tā ir par klasi?);

$Sieviete \cap \exists berns.Sieviete$ (kas tā ir par klasi?);

$\exists berns.(\exists berns.\neg Sieviete)$ (kas tā ir par klasi?);

Piezīme. Klase un koncepts (jēdziens) – kāda ir atšķirība?

Kādu klasi definē izteiksme $\neg \exists r.\neg C$? Objekts x pieder šai klasei tad un tikai tad, ja

$$\neg \exists y(r(x,y) \wedge \neg(y \in C)) .$$

Pārveidojam pēc klasiskās loģikas likumiem:

$$\forall y \neg(r(x,y) \wedge \neg(y \in C)) ;$$

$\forall y(\neg r(x,y) \vee y \in C)$;
$\forall y(r(x,y) \rightarrow y \in C)$.

Var arī bez loģikas:

$\exists r.\neg C$ nozīmē: "tie objekti, kam eksistē r ārpus klases C";

$\neg\exists r.\neg C$ nozīmē: "tie objekti, kam neeksistē r ārpus klases C", jeb:

$\neg\exists r.\neg C$ nozīmē: "tie objekti, kam **visi** r ir klasē C".

Tāpēc aprakstošo loģiku pasaulē konstrukciju $\neg\exists r.\neg C$ pieņemts apzīmēt ar $\forall r.C$ – "tie objekti, kam **visi** *r* ir klasē *C*".

Piemēri: $\forall berns.Sieviete$ – "cilvēki, kam visi bērni ir meitas". BET – ievērosim klasiskās loģikas dīvainību – šai klasei pieder arī visi tie cilvēki, kam bērnu nemaz nav!

Rodas dabisks jautājums:

Kāpēc mēs atļaujam veidot formulas, kas definē atvasinātas **klases**, bet neļaujam veidot formulas, kas definē **atvasinātas lomas**?

Tiešām, loģikā ALC pie kvantoriem atļauts izmantot tikai "atomāras" lomas r(x ,y), s(x, y) utt. Bet formulas r(x, y) v s(x, y) utt. (jebkuras formulas ar diviem brīviem mainīgiem) taču arī definē lomas? Lomas var kombinēt arī ar t.s.lomu *savienošanas* (jeb *konkatenācijas*) palīdzību:

$r \circ s = \{(x,y) \vee \exists z(r(x,z) \wedge s(z,y))\}$. Piemēram, izteiksme *berns o berns* būtībā nozīmē lomu *mazberns*. Vēl viens populārs jaunu lomu veidošanas līdzeklis ir *tranzitīvais slēgums* (piemēram, Trans(t v m) – tā no tēva un mātes lomām iegūst senča lomu). Arī tādus loģikas ALC paplašinājumus mēs vēlāk pētīsim.

Loģikas ALC definīcija "loģikas cilvēkiem"

a) Pamatā ir predikātu valoda, kurā ir tikai **objektu konstantes** (indivīdu vārdi), **vienvietīgas predikātu konstantes** (jēdzieni, koncepti, jeb klases): p(x), q(x), ... un **divvietīgas predikātu konstantes** (lomas): r(x,y), s(x,y), ...

b) Aplūko tikai formulas ar vienu brīvu mainīgo x. Katra tāda formula definē jēdzienu (jeb klasi). Atomārās ALC formulas ir tikai vienvietīgās predikātu konstantes ar mainīgo x: p(x), q(x), ...

c) Ja F(x), G(x) ir ALC formulas ar vienu brīvo mainīgo x, tad formulas $F \wedge G$, $F \vee G$, $F \rightarrow G$, $\neg F$ arī ir ALC formulas.

d) Ja F(x) ir ALC formula ar vienu brīvo mainīgo x, un r ir divvietīga predikātu konstante, tad formulas $\exists y(r(x,y) \wedge F(y))$ un $\forall y(r(x,y) \rightarrow F(y))$ arī ir ALC formulas ar vienu brīvo mainīgo x. [Pie tam, formulu F(y) varam veidot no F(x), vienkārši apmainot vietām mainīgos x un y.]

e) Ja F(x) ir ALC formula ar vienu brīvo mainīgo x, un r ir divvietīga predikātu konstante, un c, d ir objektu konstantes, tad formulas F(c) un r(c, d) arī ir ALC formulas.

No šīs definīcijas redzams, ka ALC ir apakškopa predikātu loģikai L_2 (t.i. loģikai, kuras valodā **atļauti tikai divi mainīgie** – piemēram, x un y).

Uzdevums. Pārliecināsimies, ka tā ir.

Ir zināms, ka formulu secināšanas uzdevums loģikai L_2 ir NEXPTIME-complete (vēlāk redzēsim ka loģikai ALC tas ir PSPACE-complete). Arī ALC paplašinājums ar lomu inversiju un lomu Būla operācijām (sk. tālāk) ir L_2 apakškopa. T.i. arī šim paplašinājumam formulu secināšanas uzdevums nav sarežģītāks par NEXPTIME-complete.

[Scott, 1962] – pirmoreiz pierādīja, ka loģika L2 bez vienādības ir atrisināma, [Mortimer, 1975] – ka L_2 ar vienādību arī ir atrisināma.

L3 ir neatrisināma jau formulām bez vienādības.

Par šīm lietām sk. **Carlos Areces** disertācijā, 2000.

Kā pie aprakstošajām loģikām nonāk "no apakšas"?

Semantiskie tīkli un freimi kā pirmie mēģinājumi zināšanu reprezentācijā:

Semantic network, Wikipedia, 1950-tie, 1960-tie, **M. Ross Quillian**, 1966

Marvin L. Minsky, Frames, June 1974

Aprakstošās loģikas radās, cenšoties pārvarēt semantisko tīklu un freimu semantikas neprecīzitāti. Par izšķirošo soli pieņemts uzskatīt 1985.gadā publicēto rakstu par sistēmu KL-ONE.

R. J. Brachman, **J. Schmolze**. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Sci* 9(2), 172-216, 1985. (Sk. arī KL-ONE, Wikipedia.)

Loģiku ALC pirmoreiz noformulēja 1991.gadā:

M. Schmidt-Schauss, **G. Smolka**. Attributive concept descriptions with complements. *Artificial Intelligence*, 48, 1-26, 1991.

Par šo vēsturi vislabāk izlasīt **Enrico Franconi** lekcijās Description Logics, sk. otro moduli (lekcija 2).

Loģikas ALC tradicionālā ("īstā") definīcija

Tāpat kā predikātu loģikām, arī šeit katras konkrētas sistēmas pamatā ir fiksēta valoda. ALC gadījumā valoda ir mazliet neparasta. **Mainīgos tajā neizmantojam**. Valodas primitīvi ir:

Indivīdu vārdi (piemēram, *John, Paris*).

Atomārie jēdzieni (jēdzieni, koncepti, klases): p, q, ... (piemēram, *Sieviete, Virietis, Bagats*).

Lomas: r, s, ... (piemēram, *berns, dzivesbiedrs,* lomai ir divi gali: x-a *berns* y, x-a *dzivesbiedrs* y, bet oficiāli mainīgos te neizmanto).

Atvasināto **jēdzienu (klašu) konstruktori** ļauj veidot klašu izteiksmes:

a) Būla operācijas: $C \cap D$, $C \cup D$, $\neg C$.

Piemēram,

$Virietis \cap Bagats$ (bagāts vīrietis);

$(Virietis \cup Sieviete) \cap \neg Bagats$ (nabags cilvēks).

Implikāciju $C \rightarrow D$ vienmēr reducējam uz $\neg C \cup D$.

[Grāmatās par aprakstošajām loģikām, šķēlumu pieņemts attēlot kā kvadrātu, kam izņemta apakšējā mala, bet apvienojumu – kā kvadrātu, kam izņemta augšējā mala.]

b) Kvantori: $\exists r.C$, $\forall r.C$.

Piemēram:

$\exists berns.Sieviete$ – cilvēks, kam ir meita;

$\forall berns.Sieviete$ – cilvēks, kam visi bērni ir meitas (t.sk. bērnu var nebūt nemaz);

$\exists berns.Cilveks \cap \forall berns.Sieviete$ – cilvēks, kam ir bērni un visi ir meitas.

Lomu konstruktori ALC nav paredzēti (bet jau ALCI viens tāds: inversās lomas konstruktors r^{-1}).

Aksiomas

T-Box

Paredzēts tikai viens aksiomu veids: $C_1 \subseteq C_2$, kur abās pusēs ir klašu izteiksmes.

Bet ar to var pateikt diezgan daudz, piemēram:

$C_1 \subseteq \neg C_2; \neg C_1 \subseteq C_2 \left(jeb\, T = C_1 \cup C_2 \right)$; utt.

A-Box

Tie ir apgalvojumi par indivīdiem un klasēm: $vards : C$, kur C ir klašu izteiksme (ne tikai atomāra klase!)

Piemēram, $Paris : Sieviete$;

Un par indivīdiem un lomām: $\left(vards\,1, vards\,2 \right) : r$, kur r ir atomāra loma (vai atvasināta, ja mūsu izvēlētajā loģikā tādas ir).

Manuprāt, tas ir neveiksmīgs pieraksts (bet ar to būs jādzīvo).

Piemēram, $\left(John\,,\, Paris \right) : berns$ (Džona bērns ir Parisa).

Vēl piemēri:

$Virietis \cap \exists berns. \left(Sieviete \cap \forall berns.Bagats \right)$ –

vīrietis, kam ir meita, kurai visi bērni ir bagāti.

$Cilveks \equiv Sieviete \cup Virietis$ – definē atvasinātu jēdzienu.

$Virs = Virietis \cap \exists\, dzivesbiedrs.Sieviete$ – vīrs te tiek definēts kā vīrietis, kas precējies ar sievieti.

$John : Virietis \cap Bagats$ – Džons ir bagāts vīrietis.

$(John, Paris) : dzivesbiedrs$ – Džona dzīvesbiedre ir Parisa.

Uzdevums. Šai brīdī būtu ieteicams patrenēties. Veikla ALC formulu lasīšana un rakstīšana atvieglos kursa apgūšanu un padarīs to patīkamu. Izmantojot minētos valodas līdzekļus, uzrakstiet piecus jaunus jēdzienus. Ja vēlaties, varat ievest papildus atomāros jēdzienus un lomas.

Vaicājumi

T-Box consistency: vai no T-kastes aksiomām neseko pretruna? (jeb – kas ir tas pats: vai tai eksistē modelis – par to vēlāk);

Concept satisfiability: dota klašu izteiksme C, vai pievienojot T-kastei apgalvojumu $C \neq \emptyset$, nerodas pretruna?

Concept subsumption: dotas klašu izteiksmes C_1, C_2 , vai no T-kastes seko, ka $C_1 \subseteq C_2$?

Šie trīs uzdevumi viegli reducējas viens uz otru.

A-Box consistency: vai no A-kastes aksiomām (kopā ar T-kastes aksiomām) neseko pretruna?

Instance checking: dota klašu izteiksme C un indivīds *c*, vai no A-kastes aksiomām (kopā ar T-kastes aksiomām) seko, ka $c \in C$? Vai arī: izdot visu indivīdu sarakstu, kam piederība pie C seko no aksiomām.

Šie divi uzdevumi viegli reducējas viens uz otru.

Formāli (bet bieži vien – arī faktiski), *A-kastes uzdevumi ir sarežģītāki par T-kastes uzdevumiem.*

Dažas redukcijas

1) Vai klase C nav tukša? Citiem vārdiem, vai $C \subseteq C \cap \neg C$? [Ja vēlamies, varam ievest speciālu apzīmējumu tukšai klasei, piemēram, O, un tad vaicāt: $C \subseteq O$?]

2) Vai klases C un D šķeļas? Citiem vārdiem, vai $C \cap D$ nav tukša klase?

3) Vai klases C un D nepārklāj visu objektu apgabalu? Citiem vārdiem, vai $C \cup \neg C \subseteq C \cup D$? [Ja vēlamies, varam ievest speciālu apzīmējumu "pilnajai" klasei, piemēram, T (“thing” vai “total”), un tad vaicāt: $T \subseteq C \cup D$?]

4) Tikpat labi vaicājumus var reducēt uz vaicājumu "Vai C ir tukša klase?" (kur C ir jebkura izteiksme). Tiešām, apgalvojums $C_1 \subseteq C_2$ ir ekvivalents apgalvojumam " $C_1 \cap \neg C_2$ ir tukšā klase".

Šī ALC definīcija var likties patīkamāka par definīciju "no augšas", bet šādi definētai "loģikas sistēmai" nav tik viegli uzreiz precīzi nodefinēt semantiku. Definīcijai "no augšas" šīs problēmas nav – aiz tās stāv simtgadīga matemātiskās loģikas vēsture – precīzais formālais jēdziens par valodas interpretāciju. Tāpēc dabiskais ceļš uz precīzu "otrās" ALC semantikas definīciju būtu

ALC "īstās" definīcijas translācija uz loģisko definīciju

Šī translācija definē loģikas ALC izteiksmju precīzo semantiku – katrs jēdziens C (atomārs vai atvasināts) tiks translēts par atbilstošās predikātu valodas formulu C(x) ar vienu brīvu mainīgo x. Un predikātu loģikas formulu semantikai eksistē standarta definīcija – jēdziens par valodas interpretāciju.

[Ko vispār nozīmē – definēt formālas valodas semantiku?]

Piemērs. Minētās valodas primitīvi *Sieviete, Virietis, Bagats* jātranslē par vienvietīgām predikātu konstantēm *Sieviete(x), Virietis(x), Bagats(x)*, kur x vērtību apgabals ir "visi cilvēki". Predikāts *Bagats(x)* būs patiess tad un tikai tad, ja x ir bagāts cilvēks. Lomas *berns, dzivesbiedrs* jātranslē par divvietīgām predikātu konstantēm

berns(x, y), dzivesbiedrs(x, y), kas nozīmē: "x-a bērns ir y", "x-a dzīvesbiedrs ir y" (vai otrādi: "y ir x-a bērns", "y ir x-a dzīvesbiedrs"). Un konstrukcija

$$\exists berns.Sieviete$$

jātranslē par formulu

$$F(x) \equiv \exists y\,(berns(x, y) \wedge Sieviete(y)),$$

t.i. "x-am ir meita". Bet konstrukcija

$$\forall berns.Sieviete$$

jātranslē par formulu

$$\forall y\,(berns(x, y) \rightarrow Sieviete(y)),$$

t.i. "x-am visi bērni ir meitas" (pat tad, ja x-am bērnu vispār nav).

Vispārīgais gadījums:

Valodas primitīvi:

a) Indivīdu vārdus translējam par tādām pat objektu konstantēm.

b) Atomāros jēdzienus p, q, ... translējam par vienvietīgām predikātu konstantēm ar mainīgo x: p(x), q(x), ... (tātad jēdziens būtībā nozīmēs objektu klasi).

c) Lomas s, t, ... translējam par divvietīgām predikātu konstantēm ar mainīgajiem x, y: r(x, y), s(x, y), ... (tātad lomas būtībā ir bināras asociācijas).

Konstruktori:

a) Būla operācijas. Ja jēdzieni C, D jau ir translēti par formulām C(x), D(x), tad $C \cap D$ translējam par $C(x) \wedge D(x)$, $C \cup D$ – par $C(x) \vee D(x)$, $\neg C$ – par $\neg C(x)$. Šīs operācijas tātad atbilst klašu šķēlumam, apvienojumam un papildinājumam.

b) Kvantori. Ja jēdziens C jau ir translēts par formulu C(x), un r ir loma r(x,y), tad $\exists r.C$ translējam par $\exists y\,(r(x, y) \wedge C(y))$, bet $\forall r.C$ – par formulu $\forall y\,(r(x, y) \rightarrow C(y))$. [Pie tam, formulu C(y) varam veidot no C(x), vienkārši apmainot vietām mainīgos x un y.]

Apgalvojumi par indivīdiem:

a) Ja jēdziens C jau ir translēts par formulu C(x), un c ir indivīda vārds, tad $c:C$ translējam par formulu $C(c)$.

b) Ja c, d ir indivīdu vārdi, un r – lomas, tad $(c,d):r$ translējam par formulu $r(c,d)$.

Tā kā predikātu loģikas formulu semantikai eksistē standarta definīcija – jēdziens par valodas interpretāciju, tad līdz ar to varam teikt, ka šī mūsu translācija precīzi definē arī loģikas ALC "īstās" definīcijas semantiku.

Piezīme. Kā zināms, klasiskajā predikāti loģikā principā **varētu iztikt ar vienu kvantoru,** piemēram, eksistences kvantoru $\exists xB(x)$. Universālkvantoru $\forall B(x)$ var izteikt kā $\neg\exists x\neg B(x)$. Arī loģikā ALC situācija ir līdzīga: mēs jau zinām, ka izteiksme $\forall r.C$ nozīmē klasi $\{x|\forall y(r(x,y)\rightarrow C(y))\}$, jeb $\neg\{x|\exists y(r(x,y)\wedge\neg C(y))\}$, t.i. $\neg\exists r.(\neg C)$.

1.4. Vienkāršākie ALC paplašinājumi

Vai Jums jau ir izveidojies priekšstats, ko ar ALC valodas līdzekļiem var pierakstīt, un ko nevar? Daudzi nav bijuši apmierināti ar ALC iespējām, tāpēc ir mēģinājuši šo loģiku papildināt ar citiem izteiksmes līdzekļiem. Arī šos ALC papildinājumus pieņemts saukt par aprakstošajām loģikām. Lūk, pirmie piemēri.

Lomu inversija

Tas būtu vienkāršākais **lomu konstruktora** gadījums (līdz šim mums bija tikai **klašu konstruktori**).

Katrai lomai r(x, y) dabiski atbilst **inversā** (jeb apgrieztā) **loma** $r^{-1}(y,x)$. Piemēram, *berns(x, y)* inverso lomu $berns^{-1}(y,x)$ varētu apzīmēt ar *vecaks(y, x*). Citiem vārdiem, apgalvojums $r^{-1}(y,x)$ ir patiess tad un tikai tad, ja *r(x, y).*

Te ir divas iespējas:

a) Katrai lomai *r* inversās lomas r^{-1} apzīmējums ir paredzēts jau pašā valodā. Tad mums ir vajadzīgas attiecīgās aksiomas, piemēram, $r^{-1}(y, x) \equiv r(x, y)$. Citādi sakaru starp *r* un r^{-1} dators nevarēs izmantot. Piemēram, ja mums valodā ir gan loma *berns*, gan loma *vecaks*, tad ir vajadzīga aksioma $vecaks(y, x) \equiv berns(x, y)$, citādi abi jēdzieni nebūs saistīti.

b) (Populārākā versija) Inversija ir operācija, kas jebkurai lomai *r* piekārto inverso lomu r^{-1}. Tad sakarība starp *r* un r^{-1} pieder pie inversijas operācijas semantikas (t.i. $r^{-1}(y, x) \equiv r(x, y)$ skaitīsies jau "loģikas aksioma"). Bet tad *vecaks* vietā visu laiku būs jāraksta $berns^{-1}$.

Variantu b) – šādi paplašinātu loģiku ALC pieņemt apzīmēt ar ALCI (I – inversion).

Skaitliskie ierobežojumi (*number restrictions*)

Būtībā tie ir eksistences kvantori "ar skaitītājiem":

$(\geq 2\, berns).Sieviete$ – kāds, kam ir vismaz 2 meitas,

$(\leq 4\, dzivesbiedrs).Sieviete$ – kāds, kam ir ne vairāk par 4 sievām.

Parastais eksistences kvantors tad, protams, ir rakstāms kā ≥1. Piemēram, *E berns.Sieviete* vietā varam rakstīt

$(\geq 1\, berns).Sieviete$. Skaitlisko ierobežojumu pirmsākumi ir ER-modeļos un UML klašu diagrammās – kā asociāciju galu "kardinalitātes" vai "multiplicitātes". Piemēram, lai pateiktu, ka *"katrs darbinieks strādā ne vairāk kā vienā departamentā"*, mums būtu vajadzīga šāda aksioma:

$$Darbinieks \subseteq (\leq 1\, strada).Departaments \ .$$

Šeit <= apzīmē klašu iekļaušanu (apakšklasi).

Šādi paplašinātu loģiku ALC pieņemt apzīmēt ar **ALCQ** (Q – *qualified number restrictions*).

Ja pieņem abus jau minētos paplašinājumus (skaitliskos ierobežojumus un lomu inversiju), tad iegūstam aprakstošo loģiku **ALCIQ**.

Lomu savienošana (*concatenation*)

Tas ir nākamais lomu konstruktora piemērs.

Kā ALC var nodefinēt minēto predikātu VECTEVS(x, y)?

Ar ALC var definēt tikai jēdzienus, bet ne to attiecības.

$$Vectevs = Virietis \cap \left(E\ berns.\left(E\ berns.Cilveks \right) \right)$$

Šeit nodefinēts vectēva jēdziens tā, kā to dažreiz lieto sarunvalodā cilvēka vecuma apzīmēšanai – "kāds, kam jau ir mazbērns(i)". Bet attiecību "x ir y-a vectēvs" ALC nodefinēt nevarēs, jo no ALC viedokļa tā ir loma, nevis jēdziens.

Lai šādu lomu nodefinētu, vajadzētu ievest **lomu savienošanas operāciju** "*berns* **o** *berns*" – tas būtu viens no iespējamajiem ALC paplašinājumiem. Precīzāk, lomu r un s savienojums *r* **o** *s* nozīmē pāru kopu $\{(x, y) | \exists z (r(x, z) \wedge s(z, y))\}$. T.i. *berns* **o** *berns* nozīmē lomu *mazberns*. Tālāk, $mazberns^{-1}$ nozīmē lomu *vecvecaks*. [Kā vēl pietrūkst, lai iegūtu lomu *vectevs*?]

Būla operācijas ar lomām

Bet principā klašu konstruktoros $\exists loma.jedziens$ un $\forall loma.jedziens$ varētu izmantot arī **lomu izteiksmēs**. Valodā pieejamo lomu kolekciju var paplašināt ne tikai, izmantojot inversiju un savienošanu, bet arī, izmantojot, konjunkciju, disjunkciju un pat negāciju:

$r \cap s$ jeb $r(x, y) \wedge s(x, y)$,
$r \cup s$ jeb $r(x, y) \vee s(x, y)$,
$\neg r$ jeb $\neg r(x, y)$.

Piemēram, lomu *berns* un *students* konjunkcija $berns \cap students$ saista doto personu (pasniedzēju) ar tām personām, kas vienlaicīgi ir tās bērni un studenti.

Šādi iegūtos ALC paplašinājumus pieņemts apzīmēt ar **ALC(^, v, -)**, ALC(^, v), ALC(^) utml.

Tāpat kā **klašu konstruktori** dod iespēju veidot izteiksmes, kas definē atvasinātas klases, tā arī lomu savienošanu, inversiju un Būla operācijas var uzskatīt par **lomu konstruktoriem**, kas dod iespēju veidot izteiksmes, kas definē atvasinātas lomas.

Nomināļi

Kā nodefinēt jēdzienu "John-a bērni"? Ar šo jautājumu mēs nonākam pie vēl viena loģikas ALC paplašinājuma – jēdzieniem, kas ir definēti vienkārši kā indivīdu kopas (tos sauc par **nomināļiem**). Piemēram, jēdziens *{John}* vai jēdziens *{John, Paris}*. Ja ir ievests šāds valodas līdzeklis, tad jēdzienu "John-a bērni" var uzrakstīt kā

$$\exists berns^{-1}.John \ .$$

Šo ALC paplašinājumu pieņemts apzīmēt ar **ALCO** (vai **ALCOQ**, vai **ALCOIQ**, utt. ja vienlaicīgi ir atļauti arī citi palīglīdzekļi).

Expressiveness of language pret *tractability of reasoning*

Aprakstošo loģiku paplašināšanas "kopējo noskaņu" var labi sajust, paspēlējoties ar **Evgeny Zolin**, Description Logic Complexity Navigator.

Vēlāk mēs šo ainu izpētīsim ļoti detalizēti. Navigators parāda, kā, mainot atļauto loģikas līdzekļu komplektu, mainās secināšanas sarežģītība (tiek aplūkoti divi secināšanas uzdevumi – *Concept satisfiability* un *ABox consistency*). Var novērot, ka jo vairāk izteiksmes līdzekļu mēs iekļaujam valodā, jo sarežģītāks kļūst attiecīgais secināšanas uzdevums. Piemēram, ALC un ALCI, un pat ALCIQ(^, v) secināšanas sarežģītība ir PSPACE-complete, bet ALC(^, v, -) un ALCI(^, v, -) – jau NEXPTIME-complete, t.i. iespēja izmantot lomu negācijas "maksā dārgi".

Veidojot reāli lietojamas secinātāju programmas, ir jābalansē starp valodas "izteiksmību" (*expressiveness of language*) un secināšanas realizējamību (*tractability of reasoning*). Nebūs reāli lietojama ne pārāk nabadzīga, ne pārāk bagāta valoda, kurai šīs bagātības dēļ būs pārāk grūti risināms secināšanas uzdevums. Šo novērojumu par savu nopelnu uzskata

R. J. Brachman, H. J. Levesque. The tractability of subsumption in frame-based description languages. In *AAAI-84*, pp. 34-37, 1984.

Ar šādu balansēšanu mēs tagad kādu laiciņu nodarbosimies...