Uz nodaļas satura rādītāju

Aprakstošās loģikas: teorija

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.

2.1. T-Box, A-Box un R-Box

Šis garlaicīgais temats ir ļoti labi izklāstīts kursa grāmatas 2.nodaļā.

Predikātu valodām

Ja mūsu zināšanu bāze būtu izveidota, balstoties uz predikātu valodu, tad tajā varētu izdalīt divas daļas – terminoloģijas daļu (T-kasti) un apgalvojumu daļu (A-kasti).

a) **Terminoloģijas daļa** (*terminological box*, T-Box, T-kaste) – tās formulas, kas nesatur objektu konstantes. Daži to iesaka saukt par **zināšanu bāzes shēmu.** Piemēram,

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

Te definēts vectēva jēdziens, un tas nav atkarīgs no konkrētiem cilvēkiem (ko apzīmē objektu konstantes). Protams, jēdzienu definīcijās *nedrīkstētu būt cikli*?

Šai daļā var būt arī likumi jeb aksiomas, piemēram:

$$\forall x \forall y (M(x,y) \rightarrow S(x))$$ – "visas mātes ir sievietes".

No "parasto" datubāzu viedokļa šādas aksiomas būtu integritātes ierobežojumi (*integrity constraints*).

Kādus vaicājumus var izdot šai zināšanu bāzes daļai? Arī vaicājumos T-kastei un to atbildēs nedrīkst figurēt objektu konstantes. T.i. runa var būt tikai par kādu vispārīgu formulu G ("potenciālu likumu"), par

kuru mēs gribam zināt, vai tas seko no zināšanu bāzes likumiem, vai neseko. Piemēram, ja mūsu bāzes aksiomas paredzētu, ka katram cilvēkam var būt ne vairāk kā viena māte un ne vairāk kā viens tēvs ("ne vairāk" – no datubāzes viedokļa), tad no tā sekotu, ka cilvēkam nav vairāk par diviem vectēviem:

$VECTEVS(x_1, y) \wedge VECTEVS(x_2, y) \wedge VECTEVS(x_3, y) \rightarrow x_1 = x_2 \vee x_2 = x_3$.

Uzdodot šo formulu kā vaicājumu mūsu zināšanu bāzei, mums vajadzetu saņemt atbildi "tā ir".

b) **Apgalvojumu daļa** (*assertional box*, A-Box, A-kaste) – formulas, kurās objektu konstantes parādās. Daži to iesaka saukt par **zināšanu bāzes datiem.** Piemēram,

V(John),
S(Paris),
T(Peter, John),
$\forall y \neg T(John, y)$ – John-am nav (nevar būt?) bērnu.

Šajās formulās jau ir runa par konkrētu cilvēku īpašībām un attiecībām.

Piezīme. Te varētu mēģināt iebilst: ja John-am ir bērni, tad tiem ir jābūt mūsu datubāzē, ja nav – tad nav arī datubāzē un ar to būtu jāpietiek, lai izsecinātu, ka John-am bērnu nav? Tā spriežot, mēs savai datubāzei postulējam t.s. **slēgtās pasaules pieņēmumu** (*closed world assumption*): ja pozitīvais fakts nav datubāzē, tad ir spēkā atbilstošais negatīvais fakts. Piemēram, ja datubāzē nav atrodama formula *T(Peter, John)*, tad Peter nav John-a tēvs. Bet var izmantot arī t.s. **atvērtās pasaules pieņēmumu** (*open world assumption*): ja datubāzē nav ne pozitīvā, ne atbilstošā negatīvā fakta, tad "fakts nav zināms". Ja mēs formulu $\forall y \neg T(John, y)$ sapratīsim kā "John-am nevar būt bērnu", tad tas derēs jebkuram no abiem pieņēmumiem – ja mēģināsim datubāzei pievienot, piemēram, faktu *T(John, Paris)*, tad radīsies pretruna.

Kādus vaicājumus var izdot šai zināšanu bāzes daļai? Tādus, kādus uzdod jebkurai datubāzei.

Aprakstošajām loģikām

Bet, ja zināšanu bāzi būvējam, izmantojot **aprakstošo loģiku ALC**

(ar "īsto" definīciju) vai tās paplašinājumus, tad šis iedalījums (T-kaste, A-kaste) iznāk vēl izteiktāks.

a) **T-kastes** mums arī šai gadījumā var būt **jēdzienu definīcijas un aksiomas (likumi).** Definīciju piemēri:

$$Tevs = Virietis \cap \exists berns.Cilveks \ ,$$
$$Vectevs = Virietis \cap \exists berns.\left(E\ berns.Cilveks\right) \ ,$$
$$LaimigsTevs = Virietis \cap \exists berns.Sieviete \cap ¿$$
$$\exists berns.Virietis \cap \forall\ berns.\left(Bagats\ v\ Laimigs\right) \ .$$

Vispārīgajā gadījumā jēdziena definīcija ir: $b = C$, kur b ir (atvasināta) jēdziena vārds, bet C – izteiksme, kas definē b ar citu jēdzienu palīdzību.

Bet kādi mums te varētu būt likumi (aksiomas)? Ko var pateikt par diviem jēdzieniem-izteiksmēm C1 un C2? Ka $C1 \subseteq C2$, t.i. ka jēdziens C1 ir speciālāks par C2 (jeb ka klase C1 ir klases C2 apakšklase). Protams, apgalvojums $C1 = C2$ reducējas uz $C1 \subseteq C2$ un $C2 \subseteq C1$.

Piezīme. Jēdzienu definīcijas $b = C$ **nav** jāreducē uz $b \subseteq C$ un $C \subseteq b$. Jo jēdzienu definīcijas nav aksiomas, tās vienkārši piešķir apzīmējumu atsevišķām izteiksmēm. Šai gadījumā b vienkārši ir ne-atomārs jēdziens.

Ja ir ievests **tukšais jēdziens** O (to var definēt, piemēram, kā $C \subseteq \neg C$, kur C – jebkurš jēdziens), tad var pierakstīt arī likumu $C1 \cap C2 \subseteq O$, t.i. jēdzienu C1 un C2 nesavienojamību Vai arī: likumu $C3 = C1 \cup C2$ kopā ar $C1 \cap C2 \subseteq O$. Iznāk, ka visi mūsu likumi reducējas uz formu $C1 \subseteq C2$, kur abas puses ir izteiksmes. Var ievest arī "visaptverošo" jeb **totālo jēdzienu**, apzīmējot to at T (to var definēt, piemēram, kā $C \cup \neg C$, kur C – jebkurš jēdziens).

T.i. īstenībā mums varētu pietikt ar **jēdzienu pakļautības (subsumption) aksiomām**.

Piemēri:

$$Zilonis \subseteq Dzivnieks \cap Liels \cap Peleks \ ,$$
$$Cilveks \subseteq Virietis \cup Sieviete \ ,$$
$$Virietis \cup Sieviete \subseteq Cilveks \ .$$

Ja jēdziens *Cilveks* būtu definēts kā saīsinājums izteiksmei $Virietis \cup Sieviete$, tad pēdējās divas aksiomas nebūtu vajadzīgas.

Ja mūsu T-kastē nav nevienas aksiomas (bet tajā var būt jēdzienu definīcijas), tad to sauc par **tukšu T-kasti** (empty T-Box).

Vēlams, lai T-kastē ietilpstošās aksiomas neveidotu ciklus. Tādas T-kastes sauc par **acikliskām** (acyclic T-Box). Precīzāk šis termins jādefinē tā: katru T-kastes atomāro jēdzienu mēs attēlosim kā grafa G virsotni. Ja mūsu T-kaste satur aksiomu $C1 \subseteq C2$, kur atomārais jēdziens b ieiet izteiksmē C1, bet atomārais jēdziens c – izteiksmē C2, tad no grafa virsotnes b uz virsotni c novilksim bultu. Mūsu T-kasti sauks par aciklisku, ja grafā G nebūs ciklu.

Uzmanību – precizējums! Literatūrā (un Zoļina navigatorā) tiek lietots daudz šaurāks acikliskās T-kastes jēdziens – tajā nedrīkst būt aksiomas $C1 \subseteq C2$, bet tikai jēdzienu definīcijas p=C, kur p – atomārs jēdziens, bet C – izteiksme. Šo definīciju kopā katram p var būt tikai viena definīcija, un šai kopā nedrīkst būt ciklu. Precīzāk: katru T-kastes atomāro jēdzienu mēs attēlosim kā grafa G virsotni. Ja mūsu T-kaste satur definīciju b=C, kur atomārais jēdziens c ieiet izteiksmē C, tad no grafa virsotnes b uz virsotni c novilksim bultu. Mūsu T-kasti sauks par **aciklisku**, ja grafā G nebūs ciklu. S.Tobies savā disertācijā tādas T-kastes sauc par *simple T-Box*.

Ja T-kastē ir atļautas patvaļīgas aksiomas $C1 \subseteq C2$, tad to pieņemts saukt par **vispārīgo T-kasti** (general T-Box). Vairākos svarīgos gadījumos T-kastes tips ietekmē secināšanas uzdevuma sarežģītību: tukšai T-kastei tas var būt vienkāršāks nekā acikliskai T-kastei, bet acikliskai T-kastei tas bieži ir vienkāršāks nekā visparīgai T-kastei.

Ciklisku aksiomu piemēri? $\exists berns.Zilonis \subseteq Zilonis$? Kā citādi mēs tādu secinājumu varētu iegūt, ja ne ar šādu aksiomu?

b) **A-kastē** mums šai gadījumā var būt, vispirms, **apgalvojumi par indivīdiem**, piemēram,

John: Virietis,
$John : \neg \exists berns.Cilveks$ (John-am nevar būt bērni).

jeb vispārīgajā gadījumā, k: C, kur k ir objektu konstante, bet C – izteiksme. Tas ir ļoti veiksmīgs pieraksta veids – labāks nekā predikātu valodās pieņemtais *F(John)*. Ja *Virietis(John)* vēl var viegli uztvert, tad sarežģītākai formulai tas tā vairs nav, ņemam, piemēram, to pašu apgalvojumu "John ir bagātas meitas tēvs":

$$V(John) \wedge \exists y(berns(John, y) \wedge Sieviete(y) \wedge Bagats(y)) ,$$

salīdzinot ar

$$John : Virietis \cap \exists berns.(Sieviete \cap Bagats) \ .$$

Bet mums te var būt vajadzīgi arī (pozitīvi un negatīvi) **apgalvojumi par lomām**, piemēram,

(John, Paris): berns, (Parisa ir John'a bērns)
$\neg(John, Peter) : berns$, (Peter NAV John'a bērns)

jeb vispārīgajā gadījumā: (k, m): r vai $\neg(k, m)$: r, kur k, m ir objektu konstantes, bet r – loma. Tas vairs nav tik veiksmīgs pieraksts – te *berns(John, Paris)* nebūtu sliktāks, bet *John berns Paris*, manuprāt, būtu vēl labāks (bet nav lemts...).

Tātad A-kastes sastāv no indivīdu apgalvojumiem un lomu apgalvojumiem. Tās ir **A-kastes aksiomas**.

Bet dažreiz vajadzīga vēl viena kaste...

c) **R-Box (lomu aksiomas)** jau ir aprakstošo loģiku specifika – ja mums ir nepieciešams fiksēt kādas specifiskas lomu īpašības, tad pieejamie līdzekļi te ir savādaki nekā klašu izteiksmēs (t.i. T-kastē). Piemēram, loma r var būt tranzitīva:

$$r(x, y) \wedge r(y, z) \rightarrow r(x, z) \ .$$

Aprakstošajās loģikā nav mainīgo, tāpēc šis fakts jāpieraksta, piemēram, kā *Transitive(r)* jeb citos tekstos, *Trans(r)*. Cits piemērs: arī lomām var būt jāfiksē pakļautība, piemēram, $r \subseteq s$ nozīmē, ka $\forall x \forall y(r(x, y) \rightarrow s(x, y))$. Starp citu, tad lomas *r* tranzitivitāti var pierakstīt arī kā $r \circ r \subseteq r$, kur **o** ir lomu savienošanas operācija (paldies Rihardam Opmanim).

2.2. Tipiskie secināšanas uzdevumi (*reasoning tasks*)

Pilnīgāku ārskatu par tipiskajiem vaicājumiem, kurus jāatbalsta aprakstošo loģiku sistēmām (DL systems) sk. kursa grāmatas 8.nodaļas 307. lpp.

Ja mūsu zināšanu bāze būtu izveidota, balstoties uz predikātu valodu, tad tās vaicājumu procesors būtu spiests nodarboties ar formulu secināšanas uzdevumu: vai formula-vaicājums G seko no zināšanu bāzē esošajām aksiomām (faktiem un likumiem) F_1, ..., F_n?

Ko šeit nozīmē "seko"? Ja domā "semantiski", tad tad "**formula G seko no aksiomām**" nozīmē, ka "jebkurā valodas interpretācijā, kurā patiesas ir aksiomas (t.i. aksiomu **modelī**), ir patiesa arī formula G". [Vai Jūs tam piekrītat?] Bet tā kā mēs zinām Gēdela teorēmu par pilnību, tad zinām arī, ka šis formulējums ir ekvivalents ar šādu: "formula G ir izvedama no aksiomām, izmantojot predikātu loģikas aksiomas un izveduma likumus".

Bet ko mēs varētu pavaicāt zināšanu bāzei, kas izveidota, balstoties uz tādu vai citādu **aprakstošo loģiku**? Ja runa ir par indivīdiem (t.i. mums ir gan T-kaste, gan A-kaste), tad varam pavaicāt, vai tiesa, ka b: C? Šeit b ir indivīda vārds, bet C – jēdzienu izteiksme. T.i., varam pavaicāt, vai indivīds b atbilst jēdzienam C? Piemēram,

$John : \exists berns.Cilveks$ – vai John-am ir bērni?

To sauc par **instance checking** (indivīdu piederības noskaidrošana). Atbilde var būt atkarīga gan no T-kastē, gan A-kastē ietilpstošajām aksiomām.

Varam pavaicāt arī, vai tiesa, ka (b, c): r? Šeit b un c ir indivīdu vārdi, bet r – lomas vārds (vai lomu izteiksme, ja mūsu loģikā ir atļauti arī lomu konstruktori). Piemēram,

(John, Paris): berns – vai Parisa ir John-a bērns?

To sauc par **role checking** (lomu noskaidrošana).

Līdzīgs uzdevums būtu "datu izguve" (**data retrieval**): dotajai izteiksmei C izdot visu to indivīdu vārdus, kas pieder jēdzienam C, t.i. visus tos c, kam formula c: C seko no T-kastes un A-kastes formulām. Piemēram, ja vaicājam ar izteiksmi $\exists berns.Cilveks$, tad vēlamies redzet visu to cilvēku vārdus, kam ir bērni.

Bet ko mēs mēs varētu pavaicāt T-kastei vienai pašai, bez A-kastes? Tikai kaut ko par jēdzienu attiecībām, piemēram, vai tiesa, ka no dotās T-kastes aksiomām seko, ka $C1 \subseteq C2$, kur C1 un C2 ir izteiksmes? Tas tad būtu t.s. **subsumption checking** (jēdzienu pakļautības noskaidrošana):

$\exists berns.Sieviete \subseteq \exists berns.Cilveks$ (atbilde būs "jā" – ja T-kastes aksiomas būs pietiekamas), vai

$\exists berns.Sieviete \subseteq \exists berns.Virietis$ (atbilde būs "nē" – ja aksiomas

būs pietiekamas).

Vēl varam pajautāt:

a) Vai dotās T-kastes aksiomas ir saderīgas (bezpretrunīgas, consistent)? Tas būtu **consistency checking** uzdevums. Šis jautājums ir ekvivalents ar jautājumu: vai dotās T-kastes aksiomu kopa ir izpildāma, t.i. vai eksistē interpretācija, kurā visas T-kastes aksiomas ir patiesas?

b) Vai jēdziens, ko uzdod izteiksme C, sader ar dotās T-kastes aksiomām, t.i. vai šis jēdziens nav tukšs? Tas būtu **concept satisfiability checking** uzdevums.

Consistency (satisfiability) checking var attiecināt arī uz T-kasti kopā ar A-kasti, noskaidrojot, vai abu kastu apvienotā aksiomu kopa ir saderīga vai nav. J.Zoļina navigatorā to sauc par **A-Box consistency** (A-kastes saderības) uzdevumu (it kā uzskatot, ka T-kaste un R-kaste ietilpst A-kastē). **Tas ir pats vispārīgākais secināšanas uzdevums** aprakstošajām loģikām, uz kuru var viegli reducēt visus pārējos tikko minētos uzdevumus.

Piemēram, galvenais T-kastes uzdevums, ko apspēlē Zoļina navigators – jau minētais **concept satisfiability** (jēdzienu izpildamības) checking: vai, pieņemot visas T-kastē ietilpstošās aksiomas, dotā izteiksme C attēlo netukšu jēdzienu? Šo jautājumu var reducēt uz *A-Box consistency*, ja aplūko A-kasti, kurā ir tikai viena formula c: C, kur c ir jauns indivīda vārds. Tiešām, ja mūsu jaunā A-kaste kopā at T-kasti ir saderīga (izpildāma, bezpretrunīga), tad no T-kastes aksiomu viedokļa jēdziens C nav tukšs. Un otrādi. [Pārliecināsimies!]

Otrs piemērs, *subsumption checking* uzdevums ir viegli reducējams uz *concept satisfiability*. Tiešām, apgalvojums $C1 \subseteq C2$ seko no T-kastes aksiomām tad un tikai tad, ja izteiksme $C1 \cap \neg C2$ vienlaicīgi ar šīm aksiomām nav izpildāma. [Pārliecināsimies!]

Dažām aprakstošajām loģikām **A-Box consistency (**T+A-kastes saderības) noskaidrošana ir sarežģītāks uzdevums nekā **concept satisfiability (**jēdzienu izpildāmības) noskaidrošana (bet tā, liekas, ir izņēmuma situācija, jo visiem populārākajiem loģikas ALC paplašinājumiem abu uzdevumu sarežģītības klase ir vienāda).

Tāpēc tā nav nejaušība, ka Zoļina navigators apspēlē tikai šos divus uzdevumus – *concept satisfiability* un *A-Box consistency*.

Uzdevumu (algoritmiskās) sarežģītības klases

Dažadus uzdevumus risinošo algoritmu sarežģītība tiek vērtēta ar t.s. **sarežģītības klašu** palīdzību (complexity classes). Īsu pārskatu par šo teoriju sk.:

http://en.wikipedia.org/wiki/Computational_complexity_theory from Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Complexity_class from Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/PSPACE (SEVIŠĶI LABS RAKSTIŅŠ!) from Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/EXPTIME from Wikipedia, the free encyclopedia

Šajā kursā mums vissvarīgākās ir klases PSPACE un EXPTIME.

Uzdevums **pieder klasei PSPACE**, ja to var atrisināt determinēts algoritms, kas tam paredzētos objektus garumā n apstrādā, izmantojot atmiņu, kuras lielums nepārsniedz polinomu no n (piemēram, Cn^2 vai Cn^3, kur C ir konstante, tiesa, arī kāpinātājs te var būt arī ļoti liels – bet fiksēts!).

Uzdevums U ir **PSPACE-hard**, ja jebkuru uzdevumu, ko var atrisināt ar PSPACE-algoritmu, var polinomiālā laikā (no apstrādājamā objekta lieluma) reducēt uz U.

Uzdevums U ir **PSPACE-complete**, ja tas pieder klasei PSPACE un tas ir PSPACE-hard.

Analoģiska klase nedeterminētiem algoritmiem tiek apzīmēta ar NPSPACE. Skaidrs, ka $PSPACE \subseteq NPSPACE$, bet izrādās, ka var pierādīt, ka $PSPACE = NPSPACE$ (t. s. Saviča teorēma, 1970.gads). Tas nozīmē, ka katru nedeterminētu algoritmu, kas izmanto polinomiāla izmēra atmiņu, var "determinizēt", "ne pārāk stipri" palielinot izmantojamo atmiņu.

Klasi EXPTIME definē līdzīgi, tikai te tiek mērīts algoritma darba laiks, un tam jābūt ierobežotam ar $2^{p(n)}$, kur p(n) ir polinoms no apstrādājamā objekta garuma.

Var pierādīt, ka $PSPACE \leq EXPTIME$, bet šo klašu vienādību vai nevienādību pagaidām vēl nav izdevies noskaidrot.

Tāpat ir skaidrs, ka $EXPTIME \subseteq NEXPTIME$, bet arī šo klašu vienādību vai nevienādību pagaidām vēl nav izdevies noskaidrot. Ir zināms tikai ka, ja ir patiesa slavenā hipotēze P=NP, tad arī $EXPTIME = NEXPTIME$.

Pārzinot šīs lietas, var izbaudīt informāciju, ko sniedz **Evgeny Zolin**, Description Logic Complexity Navigator. Te autors centies apkopot visu zināmo informāciju par dažādu loģikas ALC paplašinājumu algoritmisko sarežģītību. Katrai loģikas versijai tiek vērtēta divu uzdevumu sarežģītība:

Concept satisfiability (tas pats, kas consistency checking, bet ir

ekvivalents arī ar subsumption checking) – vai dotā izteiksme C ir saderīga ar dotās T-kastes aksiomām (apstrādājamā objekta garums tad ir C garums plus aksiomu kopējais garums).

Abox consistency – vai dotās T-kastes un A-kastes (un R-kastes) aksiomas ir saderīgas (apstrādājamā objekta garums tad ir abu kastu aksiomu kopējais garums).

Piemēram, pašai loģikai ALC ar tukšu T-kasti abi uzdevumi *PSPACE-complete*. Tas paliek spēkā arī, ja T-kaste nav tukša, bet ir acikliska. Toties, ja T-kaste var būt cikliska, tad abi uzdevumi kļūst *EXPTIME-complete*.

Cits piemērs. Atgriežamies pie tukšas T-kastes, bet ņemam loģiku ALCI, t.i. atļaujam lomu inversijas konstruktoru (t.i. ja mums valodā ir loma *berns*, tad automātiski ir atļauta arī loma $berns^{-1}$, t.i. *vecaks*). Šajā gadījumā concept satisfiability paliek PSPACE-complete, bet par ABox consistency uzdevuma sarežģītību 2006.gada rudenī vēl nekas nebija zināms. Šogad tas ir noskaidrots – loģikai ALCI arī ABox consistency uzdevuma sarežģītība ir PSPACE-complete.

Šī aina nemainās, ja lomu konstruktoriem pievienojam vēl konjunkciju un disjunkciju, bet lomu negācijas pievienošana uzreiz noved pie *NEXPTIME-complete* sarežģītības abiem uzdevumiem.

Šai brīdī vēlams (tagad jau) nopietnāk paspēlēties ar Zoļina navigatoru. Tā varētu sameklēt ne tikai maģistra darba, bet pat doktora disertācijas tēmu...

Zināšanu bāzes vienkāršots piemērs

Aplūkosim vienkāršotu zināšanu bāzi, kas glabā informāciju par sarežģītas programmatūras sistēmas programmām. Sistēma sastāv no programmas koda failiem, kuros definētas funkcijas un globālie mainīgie. Funkcijas izmanto tos vai citus globālos mainīgos. Pieņemsim, ka visi faili ir apstrādāti ar speciālu parsera programmu, kas iegūtos faktus ir ielādējusi mūsu zināšanu bāzē.

Mūsu bāzes valodā būs 3 atomāri jēdzieni:

File (un visu programmu koda failu vārdi kā indivīdu vārdi, piemēram, 230 gab.);
Function (un visu programmās definēto funkciju vārdi kā indivīdu vārdi, piemēram 1400 gab.);
Variable (un visu failos definēto globālo mainīgo vārdi kā indivīdu vārdi, piemēram 40 gab.);

un 2 lomas:

fun1 DefinitionFile *fil1*, jeb pareizāk – *(fun1, fil1):* Definition File*,*
var2 DefinitionFile *fil2,*
var3 UsedInFunction *fun3,*
fun4 UsedInFunction *fun5.*

Parsera programma ir ielādējusi bāzē gan visus minēto indivīdu vārdus ar piederību attiecīgajiem jēdzieniem:

fil1: File; *fun2:* Function; *var3:* Variable;

gan minētos lomu faktus (piemēram, kopā ap 60000 gab.).

T-kaste mums tātad paliek tukša.

Bet ir vajadzīga viena **R-kastes aksioma**, kas pasaka, ka UsedInFunction ir **transitīva** loma:

Trans(UsedInFunction);

Izteiksme $\exists UsedInFunction^{-1}.\{var1\}$ apzīmē jēdzienu "visas funkcijas, kas izmanto mainīgo var1". Tiešām, translējot uz predikātu valodu, iznāktu formula

{x | Ey (y UsedInFunction *x & y=var1)}* – atcerieties par lomas inversiju. Līdzīgi, izteiksme *E* DefinitionFile^{-1}.*K* apzīmē jēdzienu "viss, kas definēts failos no kopas K". Saliekot kopā, izteiksme:

$\exists DefinitionFile^{-1}.\exists UsedInFunction^{-1}.\{var1\}$ apzīmē jēdzienu "visi faili, kuros izmantots mainīgais *var1*". Attiecīgajai DL sistēmai būtu jāprot, atbildot uz šo izteiksmi kā vaicājumu, izdot mums visu to failu sarakstu, kuros izmantots mainīgais *var1* (piemēram, 96 gab.).

2.3. Acikliskās terminoloģijas

Lasiet kursa grāmatas sadaļu 2.2.2 (datne Nr. 2) līdz teorēmai 2.2 (ieskaitot).

Par **terminoloģijām** sauc T-kastes, kuras sastāv tikai no definīcijām, t.i. no aksiomām p = C, kur p ir atomāras klases vārds, un C – klases izteiksme. Šeit izteiksme C "definē" klasi p. Piemēram,

$Cilveks = Sieviete \cup Virietis$;
$Tevs = Virietis \cap \exists berns.Cilveks$.

Kā definē **acikliskas terminoloģijas**? Kas ir terminoloģijas "ekspansija"? Ievērojiet, ka terminoloģijas ekspansijas izmērs var būt eksponenciāls pret pašas terminoloģījas izmēru (Nebel, 1990).

Kā Jūs saprotat **definitoriālas terminoloģijas** jēdzienu?

Teorēma 1. Jebkura acikliska terminoloģija ir definitoriāla.

Pierādījumu Jūs varētu izdomāt paši.

Teorēma 2. Jebkura definitoriāla terminoloģija, kurā izmantoti tikai loģikas ALC līdzekļi, ir ekvivalenta kādai acikliskai terminoloģijai.

Pierādījums seko no Beth's Definability Theorem tai modālai loikai, kas ir ekvivalenta ar ALC. Lai saprastu, par ko te runa, aplūkosim Beth's Definability Theorem klasiskajai predikātu loģikai.

Beth's Definability Theorem. "Ja predikātu var definēt implicīti, tad to var definēt arī explicīti." Precīzāk, pieņemsim, ka mums ir predikātu valoda L, un šajā valodā – teorija T un slēgta formula F, kurā ieiet predikātu konstante p. Izveidosim formulu F', kurā predikātu konstante p ir aizstāta ar citu konstanti p', kas neieiet formulā F un kam ir tāds pat argumentu skaits kā p. Pieņemsim, ka teorijā T var pierādīt, ka (n ir p un p' argumentu skaits):

$$F \wedge F' \rightarrow \forall x_1 \ldots \forall x_n \left(p(x_1, \ldots, x_n) \equiv p'(x_1, \ldots, x_n) \right)$$

(tad mēs varam teikt, ka formula F "**implicīti** definē" predikātu p). Tādā gadījumā var uzbūvēt formulu G, kas nesatur p, un kam teorijā T ar klasisko loģiku var pierādīt, ka:

$$F \rightarrow \mathit{foral}\, x_1 \ldots \forall x_n \left(G \equiv p(x_1, \ldots, x_n) \right)$$

(un mēs varam teikt, ka formula G "**explicīti** definē" predikātu p).

Šo teorēmu pirmo reizi pierādīja Evert Willem Beth (1908-1964, plašāka biogrāfija-nekrologs, portrets) 1953.gadā:

E. W. Beth. On Padoa's method in the theory of definition. *Indagationes Mathematicae*, Vol. 15 (1953), pp. 330-339.

2.4. Fiksētā (nekustīgā) punkta semantika

Lasiet kursa grāmatas sadaļas 2.2.2.3 un 2.2.2.4 (datnē Nr. 2). Te ir runa par modelēšanas "augstāko pilotāžu" un reizē klasiku, ko visiem vajadzētu zināt.

Fiksētie (nekustīgie) punkti matemātiskajā analīzē

Kā aprēķināt skaitļa c kvadrātsakni? Sāksim ar minēšanu – iedomāsimies, ka meklētā kvadrātsakne ir skaitlis x. Pārbaudīsim: ja $x^2<c$, tad $(c/x)^2>c$, tātad meklētā kvadrātsakne atrodas starp x un c/x (diezgan asprātīgs novērojums!). Arī, ja $x^2>c$, tad $(c/x)^2<x$, un meklētā kvadrātsakne atrodas starp x un c/x. Tātad varam cerēt, ka vidējais (x+c/x)/2 būs tuvāk kvadrātsaknei nekā bija x. Šī ideja vedina aplūkot skaitļu virkni $\{x_n(c)\}$, kur

$x_{n+1}(c) = (x_n(c) + c / x_n(c))/2$.

Kāda ir šīs virknes robeža? Ja tāda eksistē, tad uz to tiecas gan $x_n(c)$, gan $x_{n+1}(c)$. Tātad, apzīmējot šo robežu ar a, iznāk, ka $a=(a+c/a)/2$, t.i. $a^2=c$, un tātad robeža ir kvadrātsakne no c! Ja vēlaties, varat paši pārbaudīt, ka virkne konverģē, t.i. robeža a tiešām eksistē.

Bet interesantākais te ir paņēmiens, ar kura palīdzību mēs konstatējām, ka ja robeža a eksistē, tad $a^2=c$. Pāreja no virknes n-tā locekļa $x_n(c)$ uz (n+1)-o locekli $x_{n+1}(c)$ būtībā ir transformācija, kas skaitli x pārveido par (x+c/x)/2. Un virknes robeža ir skaitlis x, kam (x+c/x)/2 = x, t.i. tas ir transformācijas "**nekustīgais punkts**" – skaitlis, kuru transformācija "nepārvieto".

Tas ir diezgan vispārīgs paņēmiens, ko var mēģināt pielietot jebkurai skaitļu virknei $y_{n+1} = f(y_n)$, kas definēta rekursīvi ar kādas funkcijas f palīdzību. Ja virkne konverģē, tad tās robeža a ir funkcijas f nekustīgais punkts: f(a)=a.

Fiksētie (nekustīgie) punkti algoritmu teorijā

Pieņemsim, ka mūs interesē tikai tās datorprogrammas, kas dotajam naturālajam skaitlim x vai nu ieciklojas, vai arī izrēķina kādas funkcija f vērtību – arī naturālu skaitli f(x). Tādas funkcijas sauc par daļēji definētām funkcijām.

Vai ir iespējams uzrakstīt tādu programmu G, kura katru programmu f pārveido par programmu G(f), kura **rēķina citu funkciju**, kas nesakrīt ar f rēķināto funkciju? Pamēģiniet tādu "programmu transformatoru" uztaisīt, un Jūs ievērosiet, ka no katra Jūsu mēģinājuma G kāda funkcija f "izsprūk", t.i. izrādās, ka G(f)=f (gan programma f, gan programma G(f) rēķina vienu un to pašu funkciju – bieži tā ir daļēji definēta vai pat nekur nedefinēta funkcija). Šī programma f tad arī ir Jūsu transformācijas "nekustīgais punkts".

Piezīme. Te ir būtiski, ka atļautas ir arī daļēji definētās funkcijas. Jo neeksistē algoritms, kas varētu pateikt, vai dotā programma rēķina visur definētu funkciju.

Un tiešām, viena no galvenajām algoritmu teorijas teorēmām ir

S. K. Klīni (S. C. Kleene) **nekustīgā punkta teorēma**. Jebkuram

algoritmam G, kurš katru algoritmu f pārveido par citu algoritmu G(f), atradīsies tāds algoritms f_0, kuram algoritms $G(f_0)$ rēķina to pašu funkciju, ko rēķina f_0.

Piezīme. Tas ir mans iecienītais nekustīgā punkta teorēmas variants. Algoritmu teorijas grāmatās gan biežāk raksta par funkciju "Gēdela numerācijām".

Nekustīgie punkti, kas definē semantiku

Ko mēs sagaidām no jēdzienu definīcijām p = C, kurās izteiksme C satur p? Vai ir kāda metode, kas šādām definīcijām var piešķirt precīzu jēgu (to tad arī sauc par semantiku)?≤

Piemēri no grāmatas.

1) $Momo = Virietis \cap \forall\, berns.Momo$

2)
$BinarsKoks = Koks \cap (\leq 2\, sakneszars) \cap \forall\, sakneszars.BinarsKoks$

Pirmās definīcijas mērķis ir jēdziens par vīriešiem, kuriem ir tikai vīriešu dzimuma pēcteči (jebkurā paaudzē). Pirmkārt, visi bezbērnu vīrieši ir Momo (mums jau zināmā loģikas dīvainība!). Otrkārt, ja kādam vīrietim visi bērni ir Momo, tad arī viņš pats ir Momo. Treškārt, ja kāds ir Momo, tad viņš ir vīrietis, un visi viņa bērni (ja tādi ir) ir Momo.

Arī otrās definīcijas ideja, šķiet, ir pilnīgi skaidra. Pirmkārt, bināram kokam saknes zaru ir ne vairāk par divi. Otrkārt, šajos zaros drīkst atrasties tikai binări koki.

Vispārīgajā gadījumā te runa ir par iepriekšējās sadaļas terminoloģijas jēdziena paplašinājumu – tagad par **terminoloģiju** sauksim jebkuru galīgu definīciju kopu $p_i=C_i$, kur katrs simbols p_i kreisajās pusēs parādās ne vairāk kā vienu reizi, bet katrā izteiksmē C_i var parādīties gan dažādi p_j, gan **bāzes simbol**i (t.i. tie simboli, kas parādās tikai izteiksmēs C_i, bet neparādās definīciju kreisajās pusēs). Grāmatas tekstā simbolus p_i sauc par **vārdu simboliem** (*name symbols*).

Kā šādas cikliskas definīcijas izskatās **interpretācijās**? Ja interpretācijā J ir interpretēti visi simboli – gan bāzes, gan vārdu

simboli, tad definīcijas uzskatāmas par aksiomām, kuru patiesumu var pārbaudīt. Grāmatas tekstā minētais piemērs 2.6 parāda, ka te var rasties neviennozīmīgas situācijas.

Aplūkosim šādu Momo piemēra interpretāciju J. Vispirms interpretēsim bāzes simbolus:

D_J = {$Charles_1$, $Charles_2$, ...} v {$James_1$, ..., $James_{Last}$} (t.i. viena bezgalīga un viena galīga indivīdu kopa);

Virietis = D_J (interpretācijā ir tikai vīrieši);

berns = { ($Charles_i$, $Charles_{i+1}$ | i>=1} v { ($James_i$, $James_{i+1}$) | 1<=i<Last}.

Bet kāda te varētu būt Momo interpretācija? Tā kā te visiem vīriešiem ir tikai vīriešu dzimuma pēcteči, tad laikam jau Momo būtu jāinterpretē kā D_J? Tiešām, tad aksioma Momo = Virietis ^ A berns.Momo būs patiesa. BET, ja mēs Momo interpretēsim tikai kā kopu {$James_1$, ..., $James_{Last}$}, arī tad aksioma būs patiesa! Tātad mūsu aksioma pieļauj arī šādu interpretāciju! Un varam pārliecināties, ka minētajā bāzes interpretācijā simbolam Momo var būt tikai šīs divas interpretācijas (jo $James_{Last}$ ir jābūt Momo kā bezbērnu vīrietim, un tad tālāk jābūt Momo arī visiem pārējiem $James_i$).

Ja otro interpretāciju mēs uzskatām par "nepareizu", tad kā mēs to varētu formāli noraidīt? Pēc kāda principa?

Tāda pat situācija ir vispārīgajā gadījumā – vienai un tai pašai bāzes simbolu interpretācijai var atbilst vairākas (vai pat daudzas) vārdu simbolu interpretācijas, kurās visas terminologijas aksiomas-definīcijas ir patiesas. Kuru no šīm interpretācijām uzskatīt par "pareizo"? Jo, ja mums nav principa, pēc kura izvēlēties "pareizo" interpretāciju (ir tikai "sajūta", kura ir "pareizā"), tad mums nav tiesību apgalvot, ka mūsu definīcijas tiešām kaut ko precīzi definē!

Ideālā situācija – mūsu terminoloģija ir tāda, ka katrai bāzes simbolu interpretācijai var būt tikai viena vārdu simbolu interpretācija, kurā visas terminoloģijas aksiomas-definīcijas ir patiesas (iepriekšējā sadaļā tādas terminoloģijas tika nosauktas par **definitoriālām terminoloģijām**, jo tās patiešām viennozīmīgi definē vārdu simbolu

interpretāciju – ja dota bāzes simbolu interpretācija).

Bet, kā redzam Momo piemērā, cikliskās terminoloģijas ne vienmēr ir definitoriālas. Tātad mums ir vajadzīgs kāds princips, kas no visām aksiomu pieļautajām vārdu simbolu interpretācijām ļauj izvēlēties vienu – "pareizo".

Visas dotajai bāzes simbolu interpretācijai atbilstošās vārdu simbolu interpretācijas var mēģināt sakārtot "pēc lieluma". Teiksim, ka $J_1 <= J_2$, ja katram vārdu simbolam p_i interpretācijā J_1 atbilstošā indivīdu klase ir apakškopa interpretācijā J_2 atbilstošajai indivīdu klasei.

"Pusideāla" situācija – mūsu terminoloģija ir tāda, ka katrai bāzes simbolu interpretācijai atbilstošās pieļaujamās vārdu simbolu interpretācijas "pēc lieluma" sakārtojas tā, ka eksistē viena vismazākā un/vai viena vislielākā (tās tad arī sauc par least fixpoint un greatest fixpoint, jo visas pieļaujamās interpretācijas ir terminoloģijas nekustīgie punkti – tam p_i un C_i interpretācijas sakrīt). Mazākais nekustīgais punkts ir tāda interpretācija J_{min}, ka $J_{min} \subseteq J$ jebkurai pieļaujamai interpretācijai J. Lielākais nekustīgais punkts ir tāda interpretācija J_{max}, ka $J_{max} \supseteq J$ jebkurai pieļaujamai interpretācijai J.

Momo piemērā mums bija tikai divas pieļaujamas interpretācijas "tikai Džeimsi" un "visi". Tā kā "tikai Džeimsi" < "visi", tas pirmā ir mazākais nekustīgais punkts, bet otrā – lielākais.

Tad mums ir pamats izvēlēties kā "pareizo" interpretāciju vienu no abām galējam – mazāko vai lielāko. Momo gadījumā mēs, protams, izvēlēsimies lielāko, jo tā labak atbilst mūsu definīcijas iecerei.

Bet kurš no variantiem mums būtu jāizvēlas bināro koku piemērā – mazākais vai lielākais? Šoreiz ir jāizvēlas mazākais nekustīgais punkts, jo mūs interesē vismazākā koku klase, kurā ietilpst:
a) visi koki, kam nav zaru (t.i. ir tikai saknes virsotne),
b) līdz ar katriem diviem saviem kokiem K1, K2 satur arī koku, kam no saknes iziet viens vai divi zari, kam galos ir K1 un K2.

Ciklisku definīciju gadījumā mazākās vai lielākās pieļaujamās interpretācijas ("nekustīgo punktu") izvēle skaitās klasisks

risinājums.

Nekustīgo punktu eksistence

Grāmatas tekstā ir minēti mākslīgi piemēri, kas parāda, ka ne visām terminoloģijām eksistē mazākais vai lielākais nekustīgais punkts. Tālak tekstā tiek meklēti pēc iespējas vispārīgi nosacījumi, kas garantē, ka terminoloģijai šādi punkti eksistē. Nekas liels jau tur nesanāk...

Pirmkārt (teorēma 2.8), mazākais un lielākais nekustīgais punkts eksistē visām **ALCN terminoloģijām, kurās nav izmantota negācija**. Abi mūsu piemēri (*Momo* un *BinaraisKoks*) bija tieši tādi.

Teorēmas 2.8 vispārinājums ir teorēma 2.9: mazākais un lielākais nekustīgais punkts eksistē visām (**ALC?) terminoloģijām, kurām atkarību grafā katrā ciklā ir pāra skaits negatīvu bultu**. (Atkarību grafa defiīciju sk. teksta 62.lpp.)

2.5. Slēgtās un atvērtās pasaules semantika

Lasiet kursa grāmatas sadaļas 2.4.4 (datnē Nr. 2).

Salīdzinām tradicionālās datubāzes un zināšanu bāzes, kas balstās uz aprakstošajām loģikām.

Tradicionālās datubāzes shēmai atbilst zināšanu bāzes T-kaste. Abās runa ir par ierobežojumiem, kas tiek uzlikti datiem (instancēm), kas glabājas bāzē.

Bet ar datubāzes datiem un A-kasti situācija ir savādāka.

Datubāzes dati ir datubāzes shēmas VIENA konkrēta interpretācija.

Zināšanu bāzes A-kaste ir formulu kopa, un tai var būt DAUDZ interpretāciju.

Tāpēc datubāzē kaut kādu datu iztrūkums nozīmē negatīvu informāciju. Piemēram, ja John-am datubāzē nav reģistrēts neviens bērns, tad tas nozīmē, ka John-am bērnu nav (attiecīgais SQL vaicājums Jums atgriezīs nulli). Tā ir t.s. **slēgtās pasaules semantika**, kurā datiem ir tikai viena interpretācija.

Toties ja A-kastē John-am nav reģistrēts neviens bērns, tad tas nozīmē tikai to, ka neviens John-a bērns nav reģistrēts. Jo secinājumu $(<=0)\ berns^{-1}.\{John\}$ ar loģikas palīdzību no A-kastes datiem izsecināt nav

iespējams. Tādu secinājumu var iegūt tikai, ja formulu *(<=0) berns*$^{-1}$. *{John}* pievienojam savai zināšanu bāzei kā aksiomu. Ja tādas aksiomas mums nav, tad interpretācijas, kurā John-am ir viens vai vairāki bērni, A-kastes formulām pretim nerunā. Tā ir t.s. **atvērtās pasaules semantika**, kurā datiem var būt daudz interpretāciju, un tāpēc informācijas trūkums nozīmē tikai informācijas trūkumu.

Piezīme. Protams, arī tradicionālā datubāzē var glabāt "informāciju par informācijas trūkumu". Piemēram, lauku "bērnu skaits" var nosaukt par "DB reģistrēto bērnu skaits", un tad tas figurēs visos attiecīgajos vaicājumos. Bet – reālais John-a bērnu skaits – tādu informāciju no šīs datubāzes saņemt nevarēs.

2.6. Raibā pasaule virs ALC

Kādi jauni sarežģījumi mūs sagaida, ja turpināsim ALC paplašināšanu ar jauniem līdzekļiem? Vispirms, mēs varam zaudēt *finite model property*.

Bezgalīgi modeļi

Aplūkosim to pašu loģiku ALCIQ (vai pat tikai ALCIF – sk. zemāk), kurai A-kastes saderības uzdevums (ar tukšu vai aciklisku T-kasti) ir PSPACE-complete, un spersim atlikušo soli – pāriesim uz vispārīgu T-kasti, t.i. atļausim patvaļīgas aksiomas.

Bet, piemēram, ja mums T-kastē ir šāda aksioma (no kursa grāmatas sadaļas 5.8, Calvanese 1996c):

T <= E padotais.B ^ (<=1 prieksnieks.T); kur T ir universālā klase, bet prieksnieks = padotais^{-1}.

tad jēdzienam -B galīgs modelis (interpretācija) nav iespējams, bet ir iespējams bezgalīgs modelis.

Tiešām, aksioma prasa, lai katram indivīdam: a) eksistētu padotais klasē B; b) eksistētu ne vairāk kā viens prieksnieks. Šai aksiomai vienai pašai vēl ir iespējams galīgs modelis – piemēram, noslēgta cilpa, piemēram: d--padotais->e--padotais->f--padotais->d. Te katram darbiniekam ir padotais, un ne vairāk kā viens priekšnieks.

Bet ja mēs gribam, lai klase -B nav tukša, tad tās indivīdam d0 eksistē padotais d1 klasē B. Protams, d1 nevar sakrist ar d0 (kurš nav klasē

B). Šim d1 arī eksistē padotais d2 klasē B. Pie tam d2 nevar sakrist ar d0 (kurš nav klasē B) un ar d1 (jo tad d1 būtu divi priekšnieki – d0 un pats d1). Arī d2 eksistē padotais d3, kurš tāpat nevar sakrist ar d0, d1 un d2. Tādā veidā modelis, kurā klase -B nav tukša, nevar būt galīgs.

Bezgalīgu modeli (interpretāciju), kurā ir spēkā minētā aksioma, un kurā klase -B nav tukša, varam uzbūvēt pavisam viegli: indivīdi būs, piemēram, naturālie skaitļi, -B = {0}, B={1, 2, 3, 4, ...}.

Bet, diemžēl, tas nozīmē, ka mēs vairs nevarēsim iztikt ar līdzšinējām metodēm, jo tās mums var dot vienīgi galīgu modeļus (t.i. interpretācijas, kas balstās uz galīgu skaitu objektu konstantu kā interpretācijas apgabalu).

Šim novērojumam ir nepatīkamas sekas tablo algoritmu būvētājiem: jau samērā vienkāršos loģikas ALC paplašinājumos saderīgiem jēdzieniem var neeksistēt galīgi modeļi, un tad tablo algoritms var iecikloties - ja vien tas nebūs speciāli izveidots tā, lai šo situāciju pamanītu. Ciklisku modeļu kontrolētai veidošanai ir vajadzīga speciālas metodes – tādas kā bloķēšana (blocking): ja konstatējam, ka veidojas cikls, tad darbošanās attiecīgajā koka zarā ir jāizbeidz. Kā to praktiski dara – var palasīt gan kursa grāmatā, gan S.Tobies disertācijā.

Kā varam novērot **E.Zoļina navigatorā**, ALC paplašinājumi veido visai raibu pasauli, kurā ir daudz interesantu uzdevumu un vēl daudz neatrisinātu problēmu – gan maģistru, an doktoru līmenim.

Piemēram, Būla operācijas ar lomām (loģika ALC(^, v, -)). Var pierādīt, ka šai loģikai A-kastes saderības uzdevums (pat tukšas T-kastes gadījumā) ir NEXPTIME-complete. Vairs netiek garantēta ne Finite model propert, ne Tree model property. Ja atsakāmies no negācijas, tad loģikai ALC(^, v) NEXPTIME vietā atkal varam dabūt PSPACE (jēdzienu saderības uzdevumam). Toties A-kastes saderības uzdevuma sarežģītības klase loģikai ALC(^, v) ir neatrisināta problēma! Manuprāt, arī te vajadzētu sanākt PSPACE, un tas liekas, ir maģistra darba līmeņa uzdevums.

Tiesa, visai šai lielajai dažādībai praktiskas nozīmes vairs nav, jo visa nopietnās publikas interese ir koncentrējusies vienā attīstības zarā – tajā, kas apkalpo semantisko tīmekli. Un šis zars sākas ar loģiku SH.

Tranzitīvas lomas un lomu hierarhijas (loģika SH).

Šis ir ļoti dabisks ALC paplašinājums.

Šai gadījumā, papildus ALC iespējām, ir atļautas, pirmkārt, aksiomas Trans(r), kas pasludina lomu r par tranzitīvu. T.i. šai lomai tad jāpiemīt īpašībai AxAyAz (r(x, y) & r(y, z) -> r(x, z)). Tāda īpašība piemīt, piemēram, lomai sencis(x, y). Faktiski tas nozīmē, ka mūs interesē tikai tie modeļi, kuros attiecīgās lomas ir tranzitīvas.

Otrkārt, ir atļautas lomu pakļautības (jeb hierarhijas) aksiomas r<=s, t.i. AxAy(r(x,y) -> s(x, y)). Tas nozīmē, ka mūs interesē tikai tie modeļi, kuros attiecīgās lomas ir norādītajā veidā pakļautas.

Piemērs (visdabiskākais!). Valodā mums ir 3 lomas: tevs, mate un sencis, un ir vajadzīgas 3 R-kastes aksiomas: tevs <= sencis, mate <= sencis, Trans(sencis). Predikātu valodā: (x ir bērns, bet y – tēvs):
tevs(x,y)->sencis(x,y), mate(x,y)->sencis(x,y), sencis(x,y) & sencis(y,z) -> sencis(x,z).

Piemērs. Ja mums jāiztiek ar lomu berns, tad senča vietā varam ievest pēcteci: berns <= pectecis, Trans(pectecis). Ja mums būtu atļauta arī lomu inversija, tad pie senča varētu tikt pavisam viegli: sencis = pectecis^{-1}.

Izrādās, ka A-kastes saderības noskaidrošanas uzdevums loģikai SH ir EXPTIME-complete. Finite model property ir garantēta, bet Tree model property – nav garantēta (kaut arī tas nav pierādīts).

Loģika SH: jebkuras T-kastes redukcija uz tukšu T-kasti.

Sk. J.Zoļina tekstu http://www.cs.man.ac.uk/~ezolin/dl/reductions.ps.

Izrādās, ka ja loģikā ir pietiekami spēcīgi līdzekļi, tad jebkuras izteiksmes C un dotās T-kastes saderības uzdevums lineārā laikā (no C un T-kastes izteiksmju kopgaruma) reducējas uz vienas izteiksmes (un tukšas T-kastes) saderības uzdevumu. Un loģikā SH tādi līdzekļi ir!

Ja T-kastē T mums ir vairākas aksiomas C1<=C2, kur C1, C2 ir (klašu) izteiksmes (jeb, citiem vārdiem, Ax(C1(x)->C2(x)), tad ar

C_T apzīmēsim visu to objektu kopu, kas šīs aksiomas apmierina. Skaidrs, ka
C_T = visu T-kastes aksiomu (-C1)v-C2 konjunkcija
(jeb šķēlums, t.i. C_T ir visai parasta izteiksme!). Aksiomas prasa, lai $T<=C_T$, t.i. lai visi objekti tās apmierinātu.

Loģikā SH mums ir arī R-kaste, apzīmēsim to ar R. To papildinot, mēs varam ievest papildus lomu U, kas būs visu esošo (T-kastes un R-kastes) lomu tranzitīvais slēgums. T.i. katrai esošai lomai r ievedam R-kastes aksiomu r<=U, un pieņemam vēl arī aksiomu Trans(U). Šādi papildinātu R-kasti apzīmēsim ar R'. (Ja mūsu loģikā – SH paplašinājumā – ir arī lomu inversijas operācija, tad ņemam arī visas aksiomas $r^{-1}<=U$.)

Teorēma 1. Izteiksme C ir saderīga ar T-kasti T un R-kasti R tad un tikai tad, ja izteiksme C ^ C_T ^ AU.C_T ir saderīga ar tukšu T-kasti un papildināto R-kasti R'.

Pierādījums. Izteiksme C ^ C_T ^ AU.C_T ietver objektus, kas pieder C un apmierina T-kastes aksiomas, un no kuriem visi ceļi (arī tukšais ceļš) pa lomām r (un arī r^{-1}, ja vajag) ved uz tikai uz objektiem, kas apmierina T-kastes aksiomas.

1) Ja izteiksme C ir saderīga ar T-kastes T un R-kastes R aksiomām , tad eksistē interpretācija, kurā visas aksiomas ir patiesas (tātad, $T<=C_T$) un kurā ir objekts c, kas pieder C. Tā kā $T<=C_T$, tad šajā interpretācijā: a) c pieder C_T, un b) ja ejam no c pa lomas U ceļiem, tad nonākam C_T piederošos objektos (jo $T<=C_T$), t.i. c pieder C ^ C_T ^ AU.C_T. Tas nozīmē arī, ka līdz ar C, arī C ^ C_T ^ AU.C_T ir netukšs jēdziens.

2) Ja C ^ C_T ^ AU.C_T ir netukšs jēdziens, tad eksistē interpretācija, kurā tam pieder vismaz viens objekts c, un kurā ispildās visas R-kastes R' aksiomas. T.i. šajā interpretācijā, c pieder C_T , un ja ejam no c pa lomas U (t.i. arī pa visu pārējo lomu) visiem ceļiem (arī pa tukšo ceļu), tad nonākam tikai C_T piederošos objektos. Tāpēc mēs varam

mūsu interpretācijas apgabalu sašaurināt, ņemot tikai tos objektus, kas ir sasniedzami no objekta c, ejot pa lomu ceļiem (tukšo ceļu ieskaitot). Ar šo sašaurināšanu mēs iegūstam jaunu interpretāciju (jo lomas "neved" ārā no interpretācijas apgabala), un pie tam: a) R-kastes aksiomas savu patiesumu nemaina, b) visi apgabala objekti pieder C_T, t.i. tie apmierina visas T-kastes T aksiomas. Un piedevām, c pieder C. Tātas esam ieguvuši interpretāciju, kurā izpildās visas T-kastes Tun R-kastes R aksiomas, un kurā izteiksme C nav tukša. Q.E.D.

Uzdevums 1. Pārliecinieties, ka tikko aprakstītā redukcija tiešām ir izpildāma lineārā laikā (t.i. ka redukcijā būvējamo divu objektu (R-kastes R' un izteiksmes C ^ C_T ^ AU.C_T) izmēri nepārsniedz konstanti, reizinātu ar (izteiksmes C garumu plus T-kastes T aksiomu kopgarumu plus R-kastes R aksiomu kopgarumu).

Loģika SHO: A-kastes saderības uzdevuma redukcija uz jēdziena saderības uzdevumu.

Sk. J.Zoļina tekstu http://www.cs.man.ac.uk/~ezolin/dl/reductions.ps.

Izrādās, ka ja loģikā ir nomināļi, tad A-kastes un T-kastes saderības uzdevums polinomiālā laikā (no kastes izteiksmju kopgaruma) reducējas uz vienas izteiksmes (un tās pašas T-kastes) saderības uzdevumu.

Uzdevums 2. Aplūkojiet J.Zoļina tekstā doto redukciju, un pierādiet, ka tā savu mērķi sasniedz. Vai šī redukcija ir izpildāma lineārā laikā? [Precīzi sekojiet saderības jēdziena definīcijai: a) virzienā -> Jums jāparāda, ka ja eksistē interpretācija, kurā visas dotās A-kastes formulas ir patiesas (kādi ir šo formulu veidi?), tad lomu U var "nointerpretēt" tā, ka objekts c piederēs klasei, kuru definē garā izteiksme; b) virzienā <- Jums jāparāda, ka ja eksistē interpretācija, kurā ir objekts c, kam patiesa ir garā izteiksme, tad šajā pat interpretācijā ir patiesas visas dotās A-kastes formulas.]

Tātad loģikas SHO paplašinājumiem visi secināšanas uzdevumi lineārā laikā reducējas uz vienas izteiksmes saderības uzdevumu ar R-kastes aksiomām.

2.7. Aprakstošās loģikas un semantiskais tīmeklis (OWL-Lite, OWL-DL un OWL 1.1)

OWL valodas līdzekļu atbilstība un aprakstošo loģiku līdzekļiem

[Apskatīsim G.Bārzdiņa lekcijas 10. un 11. slaidu.]

OWL klašu konstruktori

[Lite] intersectionOf - tas atbilst ALC klašu šķēlumam C1 ^ ... ^ Cn

[DL] unionOf - tas atbilst ALC klašu apvienojumam C1 v ... v Cn

[DL] complementOf - tas atbilst ALC klases papildinājumam -C

[DL] oneOf - tas atbilst ALCO, t.i. ar nomināļiem – {x1} v ... v {xn}

[Lite] allValuesFrom - tas atbilst ALC kvantoram Ar.C

[Lite] someValuesFrom - tas atbilst ALC kvantoram Er.C

[DL] maxCardinality - tas atbilst ALCN kvantoram <=n r.T (T ir totālā klase, nevis patvaļīga izteiksme) [Lite] - tikai ar 0 vai 1

minCardinality - tas atbilst ALCN kvantoram >=n r.T (T ir totālā klase, nevis patvaļīga izteiksme) [Lite] - tikai ar 0 vai 1

OWL aksiomas

[Lite] subClassOf - tas atbilst ALC T-kastes aksiomai C1<=C2

[Lite] equivalentClass - tas atbilst ALC T-kastes aksiomai C1=C2

[DL] disjointWith - tas atbilst ALC T-kastes aksiomai C1<=-C2

[Lite] sameAs - tas atbilst ALC A-kastes aksiomai x1=x2 vai ALCO (t.i. ar nomināļiem) T-kastes aksiomai {x1}={x2}

[Lite] differentFrom - tas atbilst ALC A-kastes aksiomai -(x1=x2) vai ALCO (t.i. ar nomināļiem) T-kastes aksiomai {x1}<=-{x2}

[Lite] subPropertyOf - tas atbilst SH R-kastes aksiomai r1<=r2

[Lite] equivalentProperty - tas atbilst SH R-kastes aksiomai r1=r2

[Lite] inverseOf - tas atbilst SHI R-kastes aksiomai $r1=r2^{-1}$

[Lite] transitiveProperty - tas atbilst SH R-kastes aksiomai Trans(r)

[Lite] symmetricProperty - tas atbilst SHI R-kastes aksiomai $r^{-1} <= r$

[Lite] functionalProperty- tas atbilst SHF T-kastes aksiomai T<=(<=1 r.T)

[Lite] inverseFunctionalProperty - tas atbilst SHIF T-kastes aksiomai $T<=(<=1\ r^{-1}.T)$

OWL-Lite atbilst loģika SHIF

Sk. OWL-Lite Synopsis no W3C OWL Web Ontology Language Overview.

Ja apskata OWL-Lite atļautos līdzekļus, tad redzam, ka šai OWL versijai vislabāk atbilst loģika SHIF (sk. tālāk). Zināmu neatbilstības iespaidu rada tikai līdzekļi, kas OWL-Lite nav paredzēti, bet SHIF ir paredzēti: no Būla operācijām ar klasēm OWL-Lite ir paredzēts tikai intersectionOf, bet jau pat ALC ir paredzēti arī unionOf un complementOf.

SHIF ir loģikas SH paplašinājums divām svarīgām un praksē bieži sastopamām lietām: ar inverso lomu konstruktoru (I)
un funkcionālām lomām (F). Lomu r(x, y) sauc par funkcionālu lomu, ja tā katram x piekārto ne vairāk kā vienu y.
Piemēram, tevs un mate ir funkcionālas lomas, un OWL to pieraksta kā functionalProperty, kas pēc dabas būtu R-kastes aksioma.
Aprakstošo loģiku pasaulē to pašu var pierakstīt ar T-kastes aksiomu, izmantojot kvantoru <=1, piemēram,
T <= (<=1 tevs.T),
kur T ir totālā klase, jeb predikātu valodā: Ax(E<=1)y tevs(x, y). Te pirmais <= nozīmē apakškopu, bet otrais – ierobežoto kvantoru.

Šādas lomas parasti atbilst vai nu objektu atribūtiem (piemēram, personas dzimšanas datums) vai foreign key relāciju datubāzēs.

R-kastes aksioma r^{-1} <= r nosaka, ka r ir simetriska loma, t.i. loģika SHI atbalsta simetriskas lomas.

Ja funkcionalitātes aksiomās atļauj tikai t.s. vienkāršās lomas (t.i. tādas, kas pašas nav tranzitīvas un nesatur tranzitīvas apakšlomas), tad gan concept satisfiability, gan ABox consistency uzdevumi loģikai SHIF ir EXPTIME-complete. Finite model property ne vienmēr izpildās. Ir aizdomas, ka ja funkcionalitātes aksiomās atļausim patvaļīgas lomas, tad abi uzdevumi kļūs algoritmiski neatrisināmi (bet pierādīts tas vēl nav!).

Protams, SHIF ir vājāka par loģiku SHIQ (SHIF no ierobežotajiem kvantoriem ir atļauts tikai E<=1, bet SHIQ ir atļauti gan E<=n, gan E>=n jebkuram n). Ja skaitliskajos ierobežojumos (Q) atļauj tikaivienkāršas lomas, tad loģikai SHIQ concept satisfiability un ABox consistency uzdevumi arī ir EXPTIME-complete. Pozitīvo

daļu, t.i. algoritmu augšējam novērtējumam pirmo reizi izstrādājis S.Tobies savas disertācijas 6.nodaļā.

OWL-DL atbilst loģika SHOIN

SHOIN ir loģikas SH paplašinājums ar nomināļiem (O), inverso lomu konstruktoru (I) un t.s. nekvalificētajiem skaitliskajiem ierobežojumiem (N). Tas nozīmē, ka ir atļauti tikai ierobežotie kvantori >=n r.T un <=n r.T, kur T ir totālā klase (nevis patvaļīga klašu izteiksme). Valodā OWL to pieraksta
kā minCardinality un maxCardinality.

Tā kā loģika SHOIN ir SHO paplašinājums, tad concept satisfiability un ABox consistency uzdevumi tai ir ekvivalenti un no šī brīža mēs varam runāt vienkārsī par secināšanas uzdevumiem.

Ja skaitliskajos ierobežojumos (N) atļauj tikai t.s. vienkāršās lomas (t.i. tādas, kas pašas nav tranzitīvas un nesatur tranzitīvas apakšlomas), tad secināšanas uzdevumi loģikai SHOIN
ir NEXPTIME-complete. Ja skaitliskajos ierobežojumos atļausim patvaļīgas lomas, tad šie uzdevumi kļūs algoritmiski neatrisināmi – tas ir pierādīts rakstā

I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive Description Logics. *Logic Journal of the IGPL*, 2000, 8(3), pp.239-263.

Protams, SHOIN ir vājāka par loģiku SHOIQ, kurai (ja skaitliskajos ierobežojumos (Q) atļauj tikai vienkāršas lomas un skaitļus pieraksta unārajā sistēmā) secināšanas uzdevumi arī ir NEXPTIME-complete. Pozitīvo daļu, t.i. algoritmu augšējam novērtējumam pirmo reizi izstrādājis S.Tobies savas disertācijas 6.nodaļā. Ja skaitļus pieraksta binārajā sistēmā, tad novērtējums NEXPTIME loģikai SHOIQ pagaidām paliek nepierādīts.

OWL 1.1 atbilst loģika SROIQ

Raksta

Ian Horrocks, Oliver Kutz, and Ulrike Sattler. **The even more irresistible SROIQ**. *Principles of Knowledge Representation and Reasoning (*KR'2006*) (*pdf).

autori parāda, kādus vēl izteiksmes līdzekļus "uzmanīgi pievienojot"

loģikai SHOIQ, ir iespējams saglabāt secināšanas uzdevumu algoritmisko atrisināmību. Šie līdzekļi esot pieņemti par pamatu nākošajai OWL iterācijai – versijai 1.1.

Papildus loģikas SHOIQ iespējām loģika SROIQ satur šādus aksiomu veidus lomām (tie rada "Ziemassvētku eglītes" iespaidu, bet jāatceras, ka autori te balansē uz algoritmiskās neatrisināmības robežas!):

a) Disjoint roles: Dis(r, s) jeb r ^ s <= o (o – tukšā loma). Tikai vienkāršām lomām. Piemēram, tevs un mate ir nesavienojamas lomas.

b) Reflexive roles, irreflexive roles, antisymmetric roles: Ref(r), Irr(r), Asy(r). Tikai vienkāršām lomām. Piemēram, paziņa varētu būt refleksīva (un simetriska) loma, tevs un mate - nerefleksīvas un antisimetriskas lomas.

c) A-kastē ir atļauti negatīvi apgalvojumi par lomām: ~r(c, d). Piemēram, (John, Paris): -patik (t.i. nepatīk). [Man liekas, ka šis papildinājums bija izdarāms jau ALC līmenī, bez jebkādām sekām uzdevumu sarežģītībai.]

d) Complex role inclusions: r o s <= s un r o s <= r, kur o ir lomu savienošana. Piemēram, ja gribam pateikt, ka "veselā" īpašnieks ir arī tā sastāvdaļu īpašnieks, tad varam rakstīt:
ipasnieks o sastavdala <= ipasnieks
No tā varēsim izsecināt, ka automašīnas īpašniekas ir arī tās dzinēja īpašnieks.

Diemžēl, atļaujot šādas lomu aksiomas, autori ir bijuši spiesti pieprasīt acikliskumu ne tikai šīm aksiomām, bet arī visām lomu hierarhijas aksiomām! Vai tas nav īstais iemesls, kāpēc E.Zoļins SROIQ raksta ar mazo burtu – kā sROIQ?

e) Universal role U, t.i. loma, kam AxAy U(x, y). [Piemērs, kad to vajag?]

f) Jēdzienu konstruktors E r.Self, kas nozīmē {x | r(x, x) }. J.Zoļins iesaka Er.Self vietā rakstīt Diag(r), jo Self nav likumīga klašu izteiksme. Es teiktu, ka Diag(r) arī nav īsti labs apzīmējums, jo rezultāts tomēr ir klase. Piemērs no raksta: E patik.Self – visi "narcisisti". [Ka būtu vispareizāk?]

Minētajā rakstā ir pierādīts, ka loģikai SROIQ secināšanas uzdevumi ir algoritmiski atrisināmi, bet nav noskaidrota to sarežģītības klase.

Tālāk sk.

OWL DL ontoloģijas un datorlingvistika (3 MB, prof. Gunta Bārzdiņa lekcijas slaidi 29.03.2006)

Uz nodaļas satura rādītāju