Deduktivne baze podataka

Goran Brajdić

26. rujna 2016.

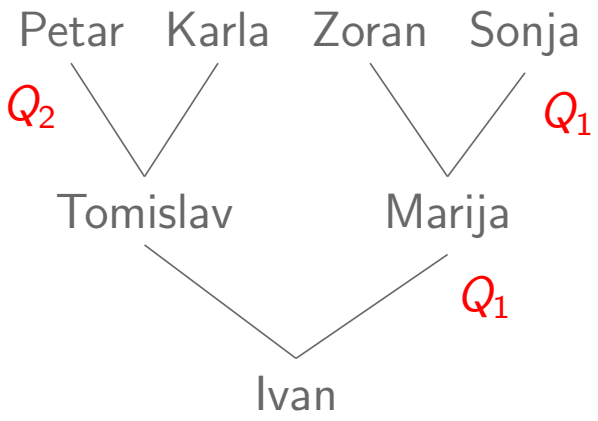
Sadržaj

Sadržaj

Deduktivne baze podataka – Uvod

Što su deduktivne baze podataka?

- Deduktivne baze podataka predstavljaju ekstenziju relacijskih baza podataka na način da podržavaju kompleksnije modeliranje podataka.
- Deduktivna baza podataka je napredna baza podataka proširena sustavom izvoda koji može izvoditi nove činjenice koristeći neka pravila nad kolekcijom podataka.
- Evaluacijom pravila nad činjenicama mogu se dobiti nove činjenice koje se tada mogu koristiti kao odgovor na upite.



BolestX			
Ime	**Roditelj**	Q_1	Q_2
Ivan	Tomislav	*Ne*	*Ne*
Ivan	Marija	*Da*	*Ne*
Tomislav	Petar	*Ne*	*Da*
Tomislav	Karla	*Ne*	*Ne*
Marija	Zoran	*Ne*	*Ne*
Marija	Sonja	*Da*	*Ne*

Obiteljsko stablo i genetske predispozicije

Postoji rizik za dobivanje neke genetske bolesti X ukoliko neka osoba naslijedi određene varijante gena Q_1 i Q_2. Postoji li rizik da Ivan dobije genetsku bolest X? U standardnom SQL:

$$(SELECT\ Ime\ FROM\ BolestX\ WHERE\ Q_1 =' Da')\ INTERSECT (SELECT\ Ime\ FROM\ BolestX\ WHERE\ Q_2 =' Da')$$

No, što ako Ivan može naslijediti gene od svih predaka?

Poznato je da neki geni mogu preskočiti generaciju, odnosno budu latentni kod određene osobe pa se manifestiraju tek kod nekog potomka. Postoji li tada rizik da Ivan dobije bolest X?

Tada bi bila potrebna ekstenzija modela

Tada bismo morali u tablici pohranjivati $(Ime,\ Predak,\ Q_1,\ Q_2)$. No to nije samo ekstenzija modela, već se javlja i potreba za promjenom sheme baze podataka. Također, zbog takve ekstenzije potrebno je i promijeniti sam sadržaj baze podataka (jer ne znamo ništa o tome tko su predci neke osobe), za što je potrebno mnogo više prostora za pohranu.

- Sada je jasno da relacijske baze podataka neće biti primarni izbor za naš skup problema.
- Razlog tome je što je znanje koje one reprezentiraju statičko.
- Da bismo mogli izvesti sve pretke neke osobe potrebna su nam sljedeća pravila dedukcije:

Pravila dedukcije

- Svaka osoba ima ime, roditelja i genetičke predispozicije.
- Svaki roditelj osobe je ujedno i predak te osobe.
- Svi roditelji predaka su predci.
- Za sve osobe postoji rizik da obole od bolesti X, ako neki predak ima varijantu gena Q_1 i neki predak ima varijantu gena Q_2.

Sadržaj

Deduktivne baze podataka – Datalog

- Deduktivne baze podataka mogu se realizirati u jeziku Datalog.
- Datalog je deklarativan logički programski jezik koji je u sintaktičkom smislu podskup Prologa.
- Razvili su ga Hervé Gallaire i Jack Minker 1987. godine.
- Koristio se prvo kao temelj za teoriju ekspertnih sustava tijekom osamdesetih godina prošlog stoljeća, a potom i kao temelj za teoriju i razvoj deduktivnih baza podataka.

Sadržaj

Alfabet nekog jezika logike prvog reda je unija sljedećih šest skupova:

1. $\Gamma = \{c_k : k \in K \subseteq \mathbb{N}\}$, neprazan skup čije elemente nazivamo **konstantski simboli**.
2. $\Omega = \bigcup_{n\in\mathbb{N}} \Omega_n$, disjunktna unija konačnih skupova Ω_n n-arnih **funkcijskih simbola**.
3. $\Pi = \bigcup_{n\in\mathbb{N}} \Pi_n$, disjunktna unija konačnih skupova Π_n n-arnih **predikatnih simbola**.
4. $X = \{x_1, x_2, x_3, \ldots\}$, prebrojiv skup čije elemente nazivamo **individualne varijable**.
5. $\{(\ ), \}$, skup **pomoćnih simbola** (lijeva i desna zagrada, te zarez).
6. $\{\neg, \vee, \wedge, \rightarrow, \leftrightarrow, \forall, \exists\}$, skup **logičkih simbola** koje redom nazivamo: negacija, disjunkcija, konjunkcija, kondicional, bikondicional, univerzalni i egzistencijalni kvantifikator.

Specifični jezik logike prvog reda definiramo kao uređenu četvorku $\mathcal{L} = (\Gamma, \Omega, \Pi, X)$.

Term je riječ definirana sljedećom induktivnom definicijom:

1) svaki konstantski simbol iz skupa Γ je term;
2) svaka individualna varijabla iz skupa X je term;
3) ako je $f \in \Omega_n$ n-arni funkcijski simbol i $t_1, \ldots, t_n$ su termi, tada je $f(t_1, \ldots, t_n)$ term.

Term koji ne sadrži individualne varijable nazivamo **osnovni term**. Skup svih terma jezika $\mathcal{L}$ označavamo sa $T_{\mathcal{L}}$.
Definiramo skup svih **atomarnih formula** jezika $\mathcal{L}$ kao

$$A_{\mathcal{L}} := \left\{ p(t_1, \ldots, t_n) : p \in \Pi_n,\ t_1, \ldots, t_n \in T_{\mathcal{L}} \right\}.$$

Atomarna formula koja se sastoji samo od osnovnih terma naziva se **osnovna atomarna formula**.

Pojam **formule** definiran je sljedećom induktivnom definicijom:

1) svaka atomarna formula je formula;
2) ako su W_1 i W_2 formule, tada su i $\neg W_1$, $W_1 \vee W_2$, $W_1 \wedge W_2$, $W_1 \rightarrow W_2$, $W_1 \leftrightarrow W_2$ također formule;
3) ako je W formula i x varijabla, tada su riječi $\forall x W$ i $\exists x W$ također formule.

Skup **literala** $L_{\mathcal{L}}$ sastoji se od svih atomarnih formula $A \in A_{\mathcal{L}}$, te odgovarajućih negiranih atomarnih formula $\neg A$. Atomarne formule nazivamo **pozitivni literali**. Negirane atomarne formule nazivamo **negativni literali**. Ukoliko neka atomarna formula ne sadrži varijable, tada se naziva **osnovni literal**.

- Univerzalno zatvorenje disjunkcije literala nazivamo **klauzula**, tj. $\forall\ (L_1 \vee L_2 \vee \ldots \vee L_n),\ L_i \in L_{\mathcal{L}}$.
- Klauzula koja sadrži najviše jedan pozitivni literal naziva se **Hornova klauzula**.

$$\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_{n-1} \vee A_n \Leftrightarrow (A_1 \wedge A_2 \wedge \ldots \wedge A_{n-1}) \rightarrow A_n$$

Hornove klauzule su zapravo implikacije sa skupom preduvjeta $\{A_1, \ldots, A_{n-1}\}$ i jednim zaključkom A_n.

Relacijske baze podataka razlikuju dva jezika:

- **Jezik za opis podataka** (eng. data description language – DDL) omogućuje stvaranje shema i pogleda (eng. views).
- **Jezik za manipulaciju podataka** (eng. data manipulation language – DML) omogućuje održavanje podataka i postavljanje upita.

U klasičnim relacijskim bazama podataka ta dva jezika uklopljena su u jedan jezik SQL.

Deduktivne BP – Sintaksa Dataloga

U deduktivnim bazama podataka nemamo striktne podjele na logički dio (DDL) i dio za manipulaciju podataka (DML) već su podaci i upiti specificirani logičkim formulama.

Svaki predikat je napisan Hornovim klauzulama oblika:

- $\forall\ (L_1 \vee L_2 \vee \cdots \vee L_n),\ L_i \in L_{\mathcal{L}}$ gdje postoji najviše jedan pozitivan literal L_j

Logičko programiranje uvodi malo drugačiju notaciju Hornovih klauzula:

- $L_j \leftarrow L_1, \ldots, L_{j-1}, L_{j+1}, \ldots, L_n.$ gdje '$\leftarrow$' predstavlja implikaciju, ',' konjunkciju, dok '.' predstavlja kraj klauzule.

Deduktivna baza podataka sastoji se od činjenica i pravila:

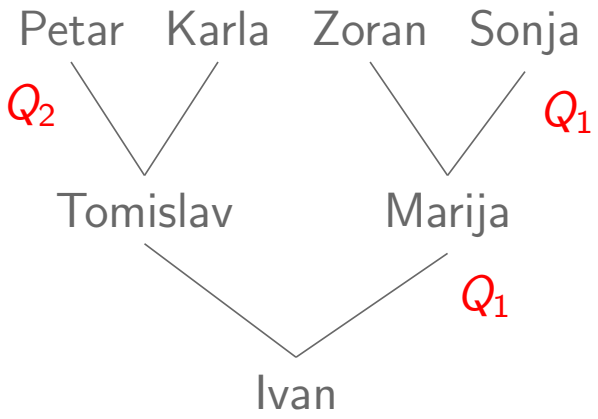
- Skup činjenica koji nazivamo **ekstenzijska baza podataka** (eng. extensional database – EDB) – ako se u činjenicama ne nalaze funkcije tada skup može biti pohranjen kao jednostavna relacijska baza podataka.
- Skup pravila koji nazivamo **intenzijska baza podataka** (eng. intentional database – IDB) – odražava ideju pogleda u relacijskim bazama podataka, ali povrh toga dopušta i rekurzije.

Klauzula baze podataka (eng. DB-clause) definirana je na sljedeći način:

- $A \leftarrow L_1, \ldots, L_n.$, gdje su $A \in A_{\mathcal{L}}$ atomarna formula, a $L_i \in L_{\mathcal{L}}$ literali. Atomarnu formulu A nazivamo **glava**, a $L_1, \ldots, L_n$ **tijelo** klauzule baze podataka. Alternativno, klauzulu baze podataka možemo pisati i kao $A :- L_1, \ldots, L_n.$ (Datalog notacija)
- Klauzule baze podataka gdje je $n > 0$ nazivamo **pravila**.
- Klauzule baze podataka gdje je $n = 0$ i A osnovna atomarna formula nazivamo **činjenice**.

- **Upit** (query) baze podataka definiramo kao: $?L_1, \ldots, L_n$, gdje su: $L_i \in L_{\mathcal{L}},\ n > 0$.
- Alternativni zapisi su: $\leftarrow L_1, \ldots, L_n$ ili $: - \ L_1, \ldots, L_n$.
- Upit koji se sastoji samo od pozitivnih literala naziva se **definitni upit**.
- Definitni upit gdje je $n = 1$ naziva se **Datalog upit**.

Deduktivne BP – Sintaksa Dataloga



činjenice (EDB)

$$\begin{cases} roditelj(\text{'Tomislav'}, \text{'Ivan'}). \\ roditelj(\text{'Marija'}, \text{'Ivan'}). \\ roditelj(\text{'Petar'}, \text{'Tomislav'}). \\ roditelj(\text{'Karla'}, \text{'Tomislav'}). \\ roditelj(\text{'Zoran'}, \text{'Marija'}). \\ roditelj(\text{'Sonja'}, \text{'Marija'}). \\ imaGenQ1(\text{'Sonja'}). \\ imaGenQ2(\text{'Petar'}). \\ imaGenQ1(\text{'Marija'}). \end{cases}$$

- Jedna od najvažnijih značajki Dataloga je mogućnost korištenja rekurzija

pravila (IDB)

$$\begin{cases} predak(X, Y) :- roditelj(X, Y). \\ predak(X, Y) :- roditelj(X, Z),\ predak(Z, Y). \\ rizik(X) :- predak(Z, X), predak(Y, X), imaGenQ1(Z), imaGenQ2(Y). \end{cases}$$

upiti

$$\begin{cases} ?predak(X, \text{'Ivan'}). \\ ?rizik(\text{'Ivan'}). \end{cases}$$

Ukoliko se predikatni simboli koji definiraju činjenice nikada ne pojavljuju u glavama pravila tada se skup klauzula baze podataka naziva **Datalog$^{f,\ neg}$ program**. U ovisnosti o tome dozvoljavamo li funkcije ili negacije možemo razlikovati nekoliko klasa Datalog jezika:

- **Datalog^{neg}** programi – ne sadrže funkcijske simbole
- **Datalog^{f}** programi – ne sadrže negativne literale.
- **Datalog** programi – ne sadrže niti negativne literale niti funkcijske simbole.

Sadržaj

Pravilo Datalog programa neće imati smisla ukoliko se varijable pojavljuju na čudan način. Primjerice:

- $S(X) :- R(Y).$
- $S(X) :- \neg R(X).$
- $S(X) :- R(Y),\ X < Y.$

U svakom od ovih slučaja rezultat je pretjerano velik ili beskonačan. U tu svrhu uzimamo u obzir samo sigurna pravila.

Pravilo $R \equiv B :- L_1, \ldots, L_n$ je sigurno ako:

- Svaka varijabla koja se nalazi u glavu pravila R, mora se pojaviti i u nekom pozitivnom (ne aritmetičkom) literalu u tijelu pravila R.
- Svaka varijabla koja se pojavljuje u negativnom literalu u tijelu pravila R mora se također pojaviti i u nekom pozitivnom literalu u tijelu pravila R.

Pretpostavka zatvorenog svijeta

Činjenica je istinita samo ako se može izvesti iz programa, odnosno ukoliko ne možemo izvesti određenu činjenicu iz programa, tada se ta činjenica smatra lažnom.

Datalog programe možemo razlikovati i po njihovoj ovisnosti među predikatima. **Graf predikatnih ovisnosti** nekog programa P sastoji se od:

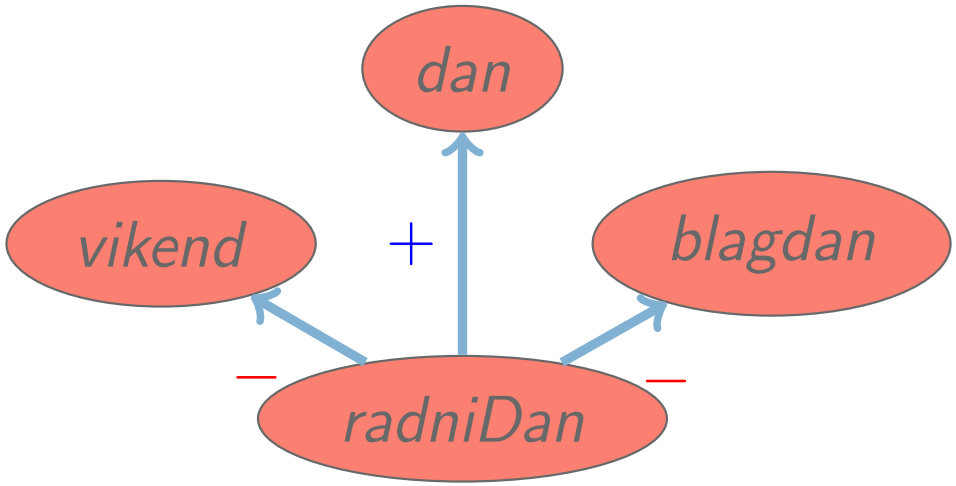
- **Čvorova** – svaki predikatni simbol $p \in P$ predstavlja jedan čvor u grafu.
- **Usmjerenih bridova** – od čvora p do čvora q ako se q pojavljuje u tijelu pravila sa predikatom p u glavi.
- Brid je **negativan** ako se q pojavljuje u negativnom literalu, u protivnom je **pozitivan**.

  – Program je **hijerarhijski** ukoliko graf predikatnih ovisnosti tog programa ne sadrži cikluse.

  – Ukoliko graf predikatnih ovisnosti nekog programa sadrži cikluse kažemo da je program **rekurzivan**.

  – Za program kažemo da je **stratificiran** ukoliko se ciklusi u grafu predikatnih ovisnosti tog programa sastoje samo od **pozitivnih** bridova.

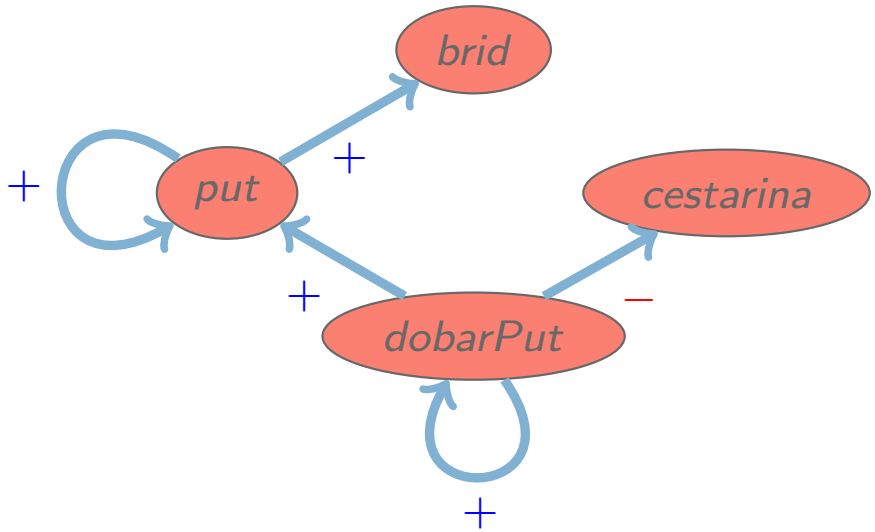
$$radniDan(X) :- dan(X),\ \neg vikend(X),\ \neg blagdan(X).$$



Slika: Graf predikatnih ovisnosti hijerarhijskog programa

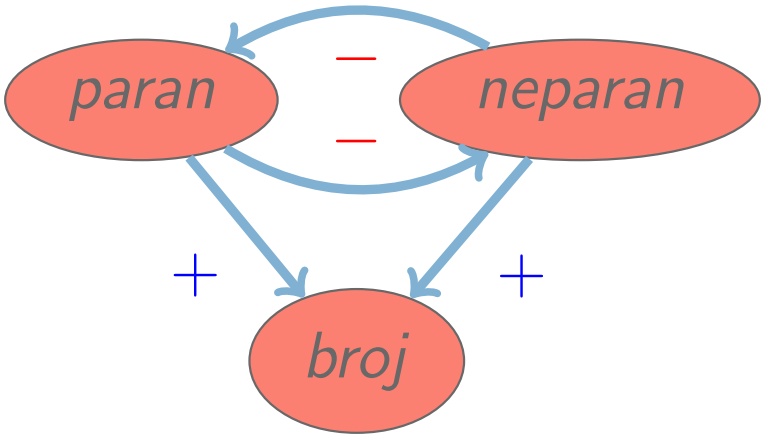
Graf predikatnih ovisnosti ne sadrži cikluse pa je stoga program hijerarhijski.

$$put(X, Y) :- brid(X, Y).$$
$$put(X, Y) :- brid(X, Z),\ put(Z, Y).$$
$$dobarPut(X, Y) :- put(X, Y),\ \neg cestarina(X, Y).$$
$$dobarPut(X, Y) :- dobarPut(X, Z),\ dobarPut(Z, Y).$$



Slika: Graf predikatnih ovisnosti rekurzivnog i stratificiranog programa

$$parni(X) :- broj(X),\ \neg neparan(X).$$
$$neparan(X) :- broj(X),\ \neg paran(X).$$



Slika: Graf predikatnih ovisnosti rekurzivnog i nestratificiranog programa

Graf predikatnih ovisnosti sadrži negativne cikluse, stoga je riječ o rekurzivnom i nestratificiranom programu.

Stratifikacija nekog programa P je particija $P = P_1 \cup \ldots \cup P_n$ od P u dijelove programa (strate) tako da vrijedi sljedeće:

- Definicija svakog predikatnog simbola je podskup neke strate.
- Definicija predikatnog simbola koji se pojavljuje u pozitivnom literalu tijela klauzule baze podataka u P_i pripada particiji P_j, za $j \leq i$.
- Definicija predikatnog simbola koji se pojavljuje u negativnom literalu tijela klauzule baze podataka u P_i pripada particiji P_j, za $j < i$.

Osnovna ideja stratifikacije je raslojavanje programa tako da su definicije predikata koji se pojavljuju u negativnim literalima uvijek prethodno dane u prijašnjim slojevima. Time efektivno isključujemo korištenje negacija unutar rekurzija.

Sadržaj

- **Herbrandov svemir (domena)** $U_{\mathcal{L}}$ jezika $\mathcal{L}$ sastoji se od svih osnovnih terma.
- **Herbrandova baza** $B_{\mathcal{L}}$ sastoji se od svih osnovnih atomarnih formula koje su formirane tako da kao argumente imaju elemente Herbrandovog svemira.

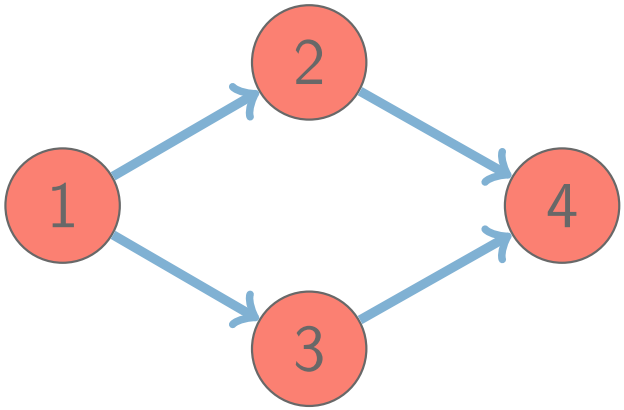
Evaluacija $Datalog^f$ programa koristi tzv. **iteraciju fiksne točke**. Osnovna ideja iteracije fiksne točke:

- Počinjemo sa praznim podskupom I_0 Herbrandove baze logičkog jezika koji se koristi u programu **P**.
- Transformiramo skup I_n u skup I_{n+1}, tj. $I_{n+1} := T_{\mathbf{P}}(I_n) := T_{\mathbf{P}}^{n+1}(I_0) := T_{\mathbf{P}}(\ldots T_{\mathbf{P}}(T_{\mathbf{P}}(I_0)))$, gdje je $T_{\mathbf{P}}$ neko pravilo transformacije.

Znati ćemo da smo dosegnuli fiksnu točku kada je $I_{n+1} = I_n$.

Elementarna produkcija je preslikavanje $T_{\mathbf{P}} : 2^{B_{\mathcal{L}}} \longrightarrow 2^{B_{\mathcal{L}}}$ koje preslikava element partitivnog skupa Herbrandove baze $B_{\mathcal{L}}$ u drugi element partitivnog skupa Herbrandove baze $B_{\mathcal{L}}$ na sljedeći način:

- $T_{\mathbf{P}} : I \mapsto \{B \in B_{\mathcal{L}} \ : \ \textit{postoji osnovna instanca}$ $B :- A_1, \ldots, A_n$ *klauzule programa tako da je* $\{A_1, \ldots, A_n\} \subseteq I\}$.
  - Klauzula programa $put(X, Y) :- brid(X, Y)$ znači da $\forall X, Y \ (put(X, Y) \vee \neg brid(X, Y))$.
  - Osnovna instanca je $put(v_1, \ v_2) \vee \neg brid(v_1, \ v_2)$, sa zamjenama varijabli $X|_{v_1}, \ Y|_{v_2}$.



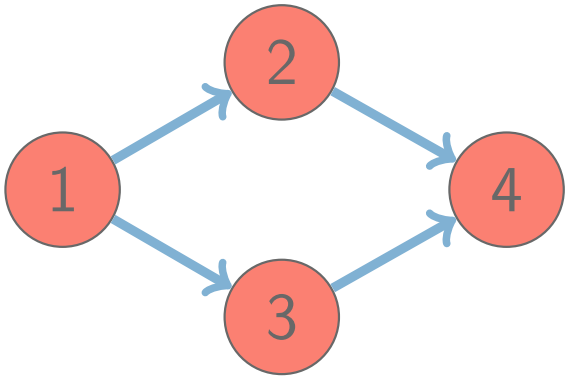
činjenice (EDB) $\begin{cases} brid(1,2). \\ brid(1,3). \\ brid(2,4). \\ brid(3,4). \end{cases}$

$$put(X,Y) :- brid(X,Y). \quad (1a)$$
$$put(X,Y) :- brid(X,Z),\ put(Z,Y). \quad (1b)$$

Ilustrirajmo iteraciju fiksne točke na danom primjeru.

- $I_0 := \{\}$.
- $I_1 := T_{\mathbf{P}}(I_0) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4)\}$
  - elementi od I_1 su osnovne činjenice.

DBP - Evaluacija Datalog programa



$$put(X, Y) :- brid(X, Y). \quad (1a)$$
$$put(X, Y) :- brid(X, Z),\ put(Z, Y). \quad (1b)$$

- Pravilo 1a.
- $I_2 := T_{\mathbf{P}}(I_1) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4),$ $put(1,2),\ put(1,3), put(2,4),\ put(3,4)\}$
- Pravilo 1b.
- $I_3 := T_{\mathbf{P}}(I_2) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4),\ put(1,2),\ put(1,3),$ $put(2,4),\ put(3,4),\ put(1,4)\}$
- $I_4 := T_{\mathbf{P}}(I_3) = I_3$, nemamo više novih putova, dakle dosegnuli smo fiksnu točku.

Za elementarnu produkciju T_P može se pokazati sljedeće:

- $I_n \subseteq I_{n+1}$, tj. unutar svake iteracije skup može samo rasti (evaluacija je monotona).
- postoji $f \geq 0$ tako da $\forall m \geq f$ vrijedi $I_m = I_{m+1}$, f se naziva fiksna točka. Vrijedi također $\forall m < f$, $I_m \subset I_{m+1}$.

- Iteraciju fiksne točke možemo shvatiti kao deduktivni sustav. Program **P** pruža aksiome, a jedino pravilo dedukcije je T_P.

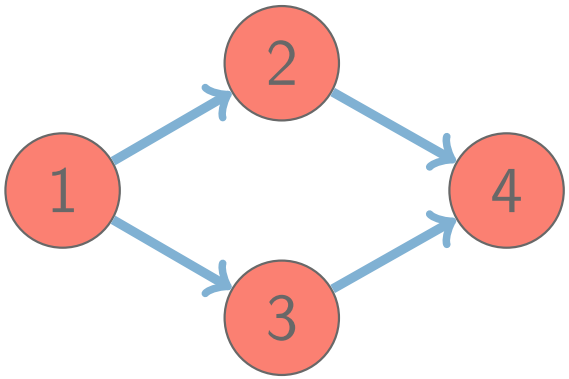
Neka je dan program **P**.

Kažemo da osnovnu činjenicu W možemo izvesti iz programa **P** u oznaci $\mathbf{P} \vdash W$ ako je $W \in \mathbf{P}$ ili ako se W može dobiti primjenom pravila dedukcije $T_{\mathbf{P}}$ nakon konačnog broja iteracija.

Naivni algoritam za izvršavanje upita

- Neka je dan Datalog program **P** i upit $Q \equiv A_1, \ldots, A_n$
- Započnimo iteraciju fiksne točke na programu **P**
  - Čim vrijedi da je $\mathbf{P} \vdash \neg Q$, vrati prazan skup kao rezultat.
  - Za svaku izvedenu osnovnu činjenicu W koja je osnovna instanca od Q, stavi W u rezultantni skup.
  - Ako smo dosegnuli fiksnu točku, vrati rezultantni skup.

DBP - Evaluacija Datalog programa



$$put(X, Y) :- brid(X, Y). \quad (1a)$$
$$put(X, Y) :- brid(X, Z),\ put(Z, Y). \quad (1b)$$
$$?put(1, X).$$

- $I_0 := \{\}$.
- $I_1 := T_{\mathbf{P}}(I_0) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4)\}$
  - Pravilo 1a.
- $I_2 := T_{\mathbf{P}}(I_1) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4),$ put(1,2), put(1,3), $put(2,4),\ put(3,4)\}$
  - Pravilo 1b.
- $I_3 := T_{\mathbf{P}}(I_2) = \{brid(1,2),\ brid(1,3),\ brid(2,4),\ brid(3,4),\ put(1,2),\ put(1,3),$ $put(2,4),\ put(3,4),$ put(1,4) $\}$,
- $I_4 := T_{\mathbf{P}}(I_3) = I_3$, nemamo više novih putova, dakle dosegnuli smo fiksnu točku.
- Rezultantni skup $= \{put(1,2),\ put(1,3),\ put(1,4)\}$

DBP - Evaluacija Datalog programa

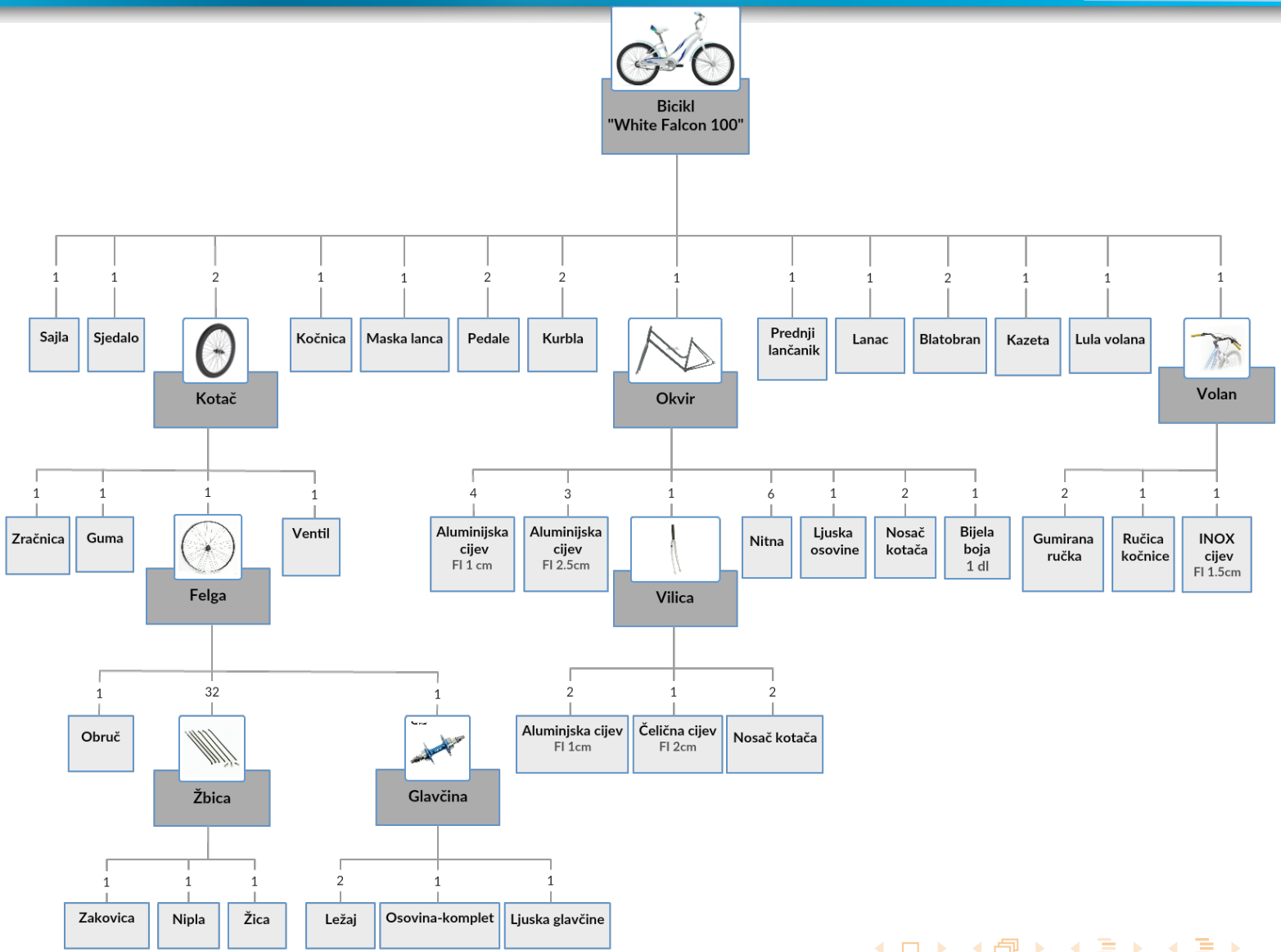
- Kako bismo evaluirali $Datalog^{neg}$ programe?
- Ideja bi bila iskoristiti rezultate stratifikacije.
- Sjetimo se **sigurne negacije** – svaka varijabla koja se pojavljuje u negativnom literalu u tijelu pravila mora se također pojaviti i u nekom pozitivnom literalu u tijelu pravila.
- Sada prvo možemo evaluirati pozitivne a zatim negativne literale. Da bismo to usvojili organizirano evaluaciju stratu po stratu.
- Ideja je da se modificira elementarna produkcija $T_{\mathbf{P}}$ tako da radi na stratama programa $\mathbf{P}$.

Sadržaj

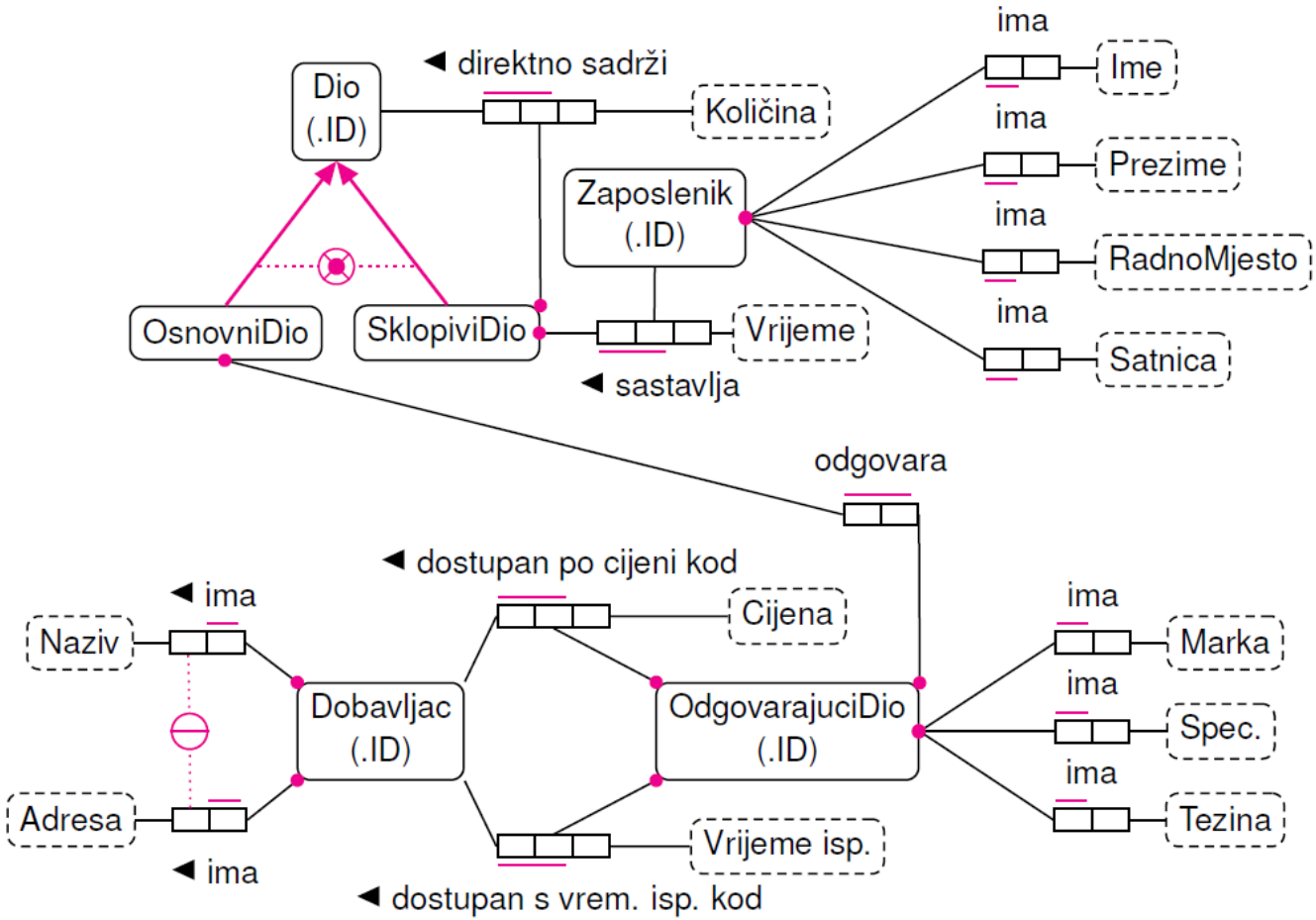
Primjer – Konstrukcijska sastavnica

- Konstrukcijska sastavnica ima znatnu primjenu u raznim prerađivačkim industrijama.
- Riječ je o bazi podataka koja sadrži sklopive dijelove koji se mogu sastojati djelomično od drugih sklopivih poddijelova koji se ponovo mogu sastojati od sklopivih poddijelova itd. sve dok se cijeli sastav ne razgradi u elementarne dijelove (koji se moraju nabaviti, primjerice od nekog dobavljača).
- Iz same formulacije problema možemo uočiti da će se javljati potreba za upitima koji koriste rekurzije.

Primjer – Konstrukcijska sastavnica



Primjer – Konstrukcijska sastavnica



Slika: Object Role Modeling dijagram za konstrukcijsku sastavnicu

Zaključak

- Deduktivne baze podataka sve do današnjice nisu doživjele veliki komercijalni uspjeh.
- I dalje se pretežno koriste relacijske baze podataka.
- Deduktivne baze podataka koriste se za stvaranje velikih baza znanja što je uglavnom izvan dosega stvarnih potreba većine današnjih aplikacija.
- No deduktivne baze podataka svakako su ostavile svoj utjecaj (SQL3 standard – Rekurzivni SQL).
- Razvoj u raznim drugim granama (rudarenje podataka, integracija podataka, mreže računala, analiza programa, sigurnost podataka te računanje u oblaku) polako ali sigurno proširuje spektar primjene deduktivnih baza podataka.

Literatura


Ralf Hinze (auth.) Armin B. Cremers, Ulrike Griefahn, *Deduktive datenbanken: Eine einführung aus der sicht der logischen programmierung*, Artificial Intelligence / Künstliche Intelligenz, Vieweg+Teubner Verlag, 1994.


K. R. Apt, H. A. Blair, and A. Walker, *Foundations of deductive databases and logic programming*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988, pp. 89–148.


T. A. Halpin, *Object-role modeling fundamentals : a practical guide to data modeling with orm*, Technics Publications, Basking Ridge, NJ, 2015.


LogicBlox, *LogiQL REPL (Read-Eval-Print-Loop)*, `https://repl.logicblox.com/?`, [Zadnji put pristupljeno 13. lipnja 2016.].


LogicBlox, *LogicBlox 4 Reference Manual*, `https://developer.logicblox.com/content/docs4/core-reference/html/index.html`, [Zadnji put pristupljeno 10. lipnja 2016.].


T A Halpin; Spencer Rugaber, *Logiql: A query language for smart databases*, Emerging Directions in Database Systems and Applications, CRC Press, 2014.


Letizia Tanca (auth.) Stefano Ceri, Georg Gottlob, *Logic programming and databases*, Surveys in Computer Science, Springer-Verlag Berlin Heidelberg, 1990.

Zahvaljujem na pažnji!

Pitanja ?

Goran Brajdić
gbrajdic@gmail.com