Declarative semantics for active rules

Sergio Flesca, Sergio Greco

Aktiivreeglite deklaratiivne semantika

Raivo Laanemets

Sissejuhatus (1)

- *Aktiivne reegel* - mingi reegel, mis kirjeldab mingisugust tingimuslikku tegevust, mis sooritatakse teatud sündmuse toimumise tõttu.

- "Normaalses" andmebaasis *(RDBMS)* nimetame selliseid reegleid *triggeriteks.*

- Loogilises andmebaasis *(Prolog, Datalog)* on nende nimeks lihtsalt *aktiivreeglid.*

- Sündmuse tõttu toimuvat andmebaasi muutust $D \rightarrow D'$ vaatleme kui mingite teiste faktide kustutamist/lisamist.

Sissejuhatus (2)

Kuna andmebaas võib sisaldada rohkem kui ühte aktiivreeglit, siis tekivad küsimused:

1. Kuidas mitut reeglit rakendada?

2. Kas rakendamisel on reeglite järjekord tähtis?

3. Kas reeglite rakendusena tekib lõpmatu tsükkel?

4. Kas reeglid $r_1, .., r_n$ ja andmebaas D määravad üheselt D'?

5. Milline on reeglite $r_1, .., r_2$ rakenduse arvutuslik efektiivsus?

Siin uurime põhiliselt probleeme (1,2,3,4).

Aktiivreeglite süntaks (1)

Lisaks tavalistele aatomitele võtame kasutusele aatomid, millega saab väljendada andmebaasi fakti lisamist ja fakti kustutamist.

Aatomid kujul $-A, +A$ reeglis r:

- Kui sellised aatomid asuvad r päises, nimetame neid *andmebaasi muutvateks aatomiteks*. *(update atom).* Kui $+A$ ($-A$) on tuletatav reeglite rakendamisel, siis lisatakse (kustutatakse) andmebaasi(st) A.

- Kui sellised aatomid asuvad r keha osas, nimetame neid *tegevust kajastavateks literaalideks. (update literal)*

Aktiivreeglite süntaks (2)

Reeglid on kujul

$$A \leftarrow B_1, .., B_n, C_1, .., C_m,$$

kus A on aatom, $B_1, .., B_n$ on tegevust kajastavad literaalid ja $C_1, .., C_n$ on normaalsed literaalid. Aatom võib olla nii *tavaline* aatom või aatom kujul $-A$, $+A$. Reeglit nimetatakse *aktiivseks*, kui tema päises on andmebaasi muutev aatom (*update atom*). Reeglit nimetame *tavaliseks*, kui tema päises on normaalne aatom.

Protseduraalne semantika (1)

Olgu antud reeglid:

$r_1 : -mgr(X, P, D) \leftarrow -proj(P), mgr(X, P, D).$
$r_2 : +mgr(X, P, D) \leftarrow -mgr(X, P, D), \neg diff\_mgr(X, D).$
$r_3 : -mgr(X, P, D) \leftarrow, +mgr(X, P, D), \neg proj(P).$
$diff\_mgr(X, D) \leftarrow mgr(X', P, D), X \neq X.$

ja koosnegu andmebaas faktidest $proj(p), mgr(x, p, d)$.

Näide 1: Kasutaja teeb päringu $\delta = \{-proj(p)\}$ ehk soovib kustutada projekti p.

Protseduraalne semantika (2)

Reegel r_1 : $-mgr(X, P, D) \leftarrow -proj(P), mgr(X, P, D)$ tähendab: kui projekt P kustutatakse, siis kõik projekti P projektijuhid tuleb kustutada.

Reegel r_2 : $+mgr(X, P, D) \leftarrow -mgr(X, P, D), \neg diff\_mgr(X, D)$ tähendab: kui projekti P juht X kustutatakse osakonnas D, ja selles osakonnas pole ühtegi teist juhti, siis juht X tuleb alles jätta.

Reegel r_3 : $-mgr(X, P, D) \leftarrow +mgr(X, P, D), \neg proj(P)$ tähendab: kui projekti P juhti X osakonnast D tahetakse lisada andmebaasi, aga projekt P ei ole andmebaasis, siis juhti X ei lisata.

Protseduraalne semantika (3)

Eespool vaadeldud reeglid moodustavad projektide ja projektijuhtide andmebaasil terviklikkust (*integrity*) garanteerivad kitsendused.

Protseduraalne semantika tähendab: päringu δ saamisel rakenda reegleid $r_1, .., r_3$ järjekorras: kõigepealt r_1, siis r_2 ja lõpuks r_3, kusjuures üks reegel võib vallandada mingi teise reegli rakendumise. Tegevust kajastavad literaalid määratakse konkreetse transaktsiooni (δ rakendamine algusest lõpuni) piires.

Protseduraalne semantika (4)

Vaatleme näites (1) tehtud päringut $\delta = \{-proj(p)\}$, kasutades protseduraalset semantikat. Andmebaasis on alguses faktid $\{proj(p), mgr(x, p, d)\}$. Kõigepealt kustutame $proj(p)$. r_1 tulemusena saame $-mgr(x, p, d)$, siis r_2 rakendamisel saame $+mgr(x, p, d)$, r_3 korral $-mgr(x, p, d)$, mis käivitab uuesti r_2. Nii tekib r_2 ja r_3 kaudu lõpmatu tsükkel, mis järjest lisab ja kustutab fakti $mgr(x, p, d)$.

Protseduraalne semantika (5)

Võibolla oleksime suutnud lõpmatut tsüklit vältida, kasutades veidi teistsuguseid reegeid.

Näites (1) oli ainult 3 aktiivset reeglit. Kui neid on 30, siis tsüklite vältimine ja andmebaasi terviklikkuse tagamine on väga raske.

Üldisel juhul on n reegli tsüklilisuse kontrollimine mittelahenduv ülesanne.

Järeldus: seega peame valima mingi parema viisi aktiivreeglite rakendamiseks andmebaasis.

Deklaratiivne semantika (1)

Loogilistel programmidel (keeles $Prolog$ või $Datalog$) on läbiuuritud semantika.

Et aktiivreeglitega programmile $\mathcal{AP}$ rakendada deklaratiivset semantikat, tuleb $\mathcal{AP}$ teisendada loogiliseks programmiks $\mathcal{DP}$.

Selleks on välja pakutud mitu lahendust. Meie vaatleme *S.Flesca* ja *S.Grecko* esitatud algoritmi.

Deklaratiivne semantika (2)

Andmebaasiks nimetame paari $D = (D^+, \overline{D})$. D^+ tähistab kõikide tõeste faktide hulka andmebaasis D ja $\overline{D}$ kõikide tundmatute faktide hulka andmebaasis D. D^- tähistab kõikide D väärade faktide hulka.

Päringut M nimetame (siin räägime lisamisest või kustutamisest)

1. *Konfliktide vabaks,* kui M ei sisalda seisu uuendavaid fakte kujul $+A$ ja $-A$ (korraga ei lisa ja kustuta ühte fakti).

2. *Mittevasturäägivaks*, kui iga M aatomi $+A$ ($-A$) jaoks on selge, et $\neg - A \in M$ ($\neg + A \in M$).

Deklaratiivne semantika (3)

Olgu $D = (D^+, \overline{D})$ andmebaas ja M mittevasturääkiv päring. Siis päringu M *rakendamine* $D_1 = M(D)$ andmebaasile D on defineeritud kui:

1. $p \in D_1^+$, kui $+p \in M$ või $p \in D^+$ ja $\neg -p \in M$

2. $p \in \overline{D_1}$, kui kehtib üks järgmistest:

   (a) $p \in \overline{D}$ ja $+p, -p \notin M$

   (b) $p \in D^+$ja $-p \in \overline{M}$

   (c) $p \in D^-$ ja $+p \in \overline{M}$

Deklaratiivne semantika (4)

Kus $\circ p \in \overline{M}, \circ \in \{+, -\}$ kui $-p \notin M$ ja $-p \notin M$, st. p kustutamine või lisamine ei ole M järgi leitav.

$M(D)$ definitsiooni järgi saame algoritmi aktiivprogrammi $\mathcal{AP}$ koos päringuga δ teisendamiseks loogiliseks programmiks $\mathcal{DP}$.

Teisendusalgoritm (1)

Sisend: $\mathcal{UP} = (\delta, \mathcal{AP})$
Väljund: loogiline programm $st(\mathcal{UP})$
1. Lisa $st(\mathcal{UP})$-sse $\mathcal{AP}$ mitteaktiivsed reeglid.
2. Iga $\mathcal{AP}$ reegli $\circ a(t) \leftarrow B$ jaoks lisa $st(\mathcal{UP})$-sse reegel $\circ a(t) \leftarrow B, \neg ck\_a(t)$.
3. Iga reegli $r \in st(\mathcal{UP})$ korral asenda r kehas $\circ p(t)$ $\circ p(t) \vee p'(t)$-ga.
4. Iga sammus (2) saadud n-aarse predikaadi ck_a jaoks lisa $st(\mathcal{UP})$-sse reegel $ck\_a(X_1, .., X_n) \leftarrow +a(X_1, .., X_n), -a(X_1, .., X_n)$.
5. Iga $\circ p \in \delta$ korral lisa $st(\mathcal{UP})$-sse fakt $\circ p'$ (vt. samm (3)).
6. Teisenda $st(\mathcal{UP})$-st sammus (3) tekkinud disjunktsioonid.
7. Tagasta $st(\mathcal{UP})$, kus $\circ p$ interpreteeritakse kui tavalised predikaatsümbolid.

Teisendusalgoritm (2)

Algoritmi väljundina saadud programmis $st(\mathcal{UP})$ loodud predikaatsümboleid $+p(t)$ ($-p(t)$) interpreteerime kui faktide lisamist (kustutamist).

Rakendame teisendusalgoritmi näites (1) toodud aktiivprogrammile ja päringule δ.

Deklaratiivne semantika (5)

Enne algoritmi sammu 6 rakendamist on meil programm:

$$-mgr(X,P,D) \leftarrow (-proj(P) \vee proj'(P)),$$
$$\quad mgr(X,P,D), \neg ck\_mgr(X,P,D).$$
$$+mgr(X,P,D) \leftarrow (-mgr(X,P,D) \vee mgr''(X,P,D)),$$
$$\quad \neg diff\_mgr(X,P,D), \neg ck\_mgr(X,P,D).$$
$$-mgr(X,P,D) \leftarrow (+mgr(X,P,D) \vee mgr'(X,P,D)),$$
$$\quad \neg proj(P), \neg ck\_mgr(X,P,D).$$
$$proj'(p).$$
$$ck\_mgr(X,P,D) \leftarrow +ck\_mgr(X,P,D), -ck\_mgr(X,P,D).$$
$$diff\_mgr(X,D) \leftarrow mgr(X',P,D), X \neq X.$$

Deklaratiivne semantika (6)

Eelmisel slaidil saadud programm tuleb veel ümber kirjutada ilma disjunktsioonideta reeglite kehas.

Mis saab programmist $st(\mathcal{UP})$ edasi, sõltub programmi loogilisest semantikast.

Programmi peatumine sõltub vastava loogilise programmi peatumisest.

Samuti sõltub saadava andmebaasi D_1 ühesus (*confluence*) $st(\mathcal{UP})$ ühesusest konkreetses semantikas.

Stabiilse mudeli semantika (1)

Stabiilse mudeli semantika (*stable model* semantics) kasutab suletud maailma eeldust. Eitus on realiseeritud kui *negation as failure*.

Kui programmil $st(\mathcal{UP})$ ei leidu ühtegi stabiilset mudelit, siis programmiga midagi edasi teha ei saa.

Kui programmil $st(\mathcal{UP})$ leidub täpselt üks stabiilne mudel, siis ütleme, et semantika on determineeritud (*deterministic*). Sellisel juhul on saadav andmebaas üheselt määratud (*confluent*).

Kui programmil $st(\mathcal{UP})$ on rohkem kui üks stabiilne mudel, siis on meil mitteühene semantika (*nonconfluent semantics*).

Stabiilse mudeli semantika (2)

Artiklis vaadeldakse erinevaid stabiilse mudeli semantikaid.

Üldiselt jõutakse järeldusele, et suure väljendusvõimsusega semantikat (suurem võimalus, et programmil leidub mudel) kasutades on tulemuse arvutuse efektiivsus väiksem ja vastupidi.

Artiklis uuritakse *well-founded, max-deterministic, total-deterministic* jt. semantikaid. Sealjuures vaadeldakse, millal vastava mudeli leidumine on garanteeritud.

Kokkuvõte

Vaatlesime erinevaid semantikaid aktiivreeglitega loogiliste andmebaaside jaoks.

Erinevatel semantikatel on erinevad probleemid.

Protseduraalne semantika ei ole deklaratiivne ja tema probleemideks on lõpmatud tsüklid ja mitte üheselt defineeritud lõpptulemus.

Deklaratiivsete lähenemise korral teisendame aktiivprogrammi koos päringuga loogiliseks programmiks ja rakendame mõnda loogilist semantikat. Nii saame mõned probleemid lahendada, aga mõned jäävad alles.