Deductive Rules

Contents

- Introduction, Motivation: Deductive Databases
- Example
- Minimal Models
- Stratified Models
- Well-Founded Semantics
- Stable
- Conclusion

Remarks

- most semantics/results will be given for Datalog$^{\neg}$
  (but are also applicable to logic programs with function symbols (Prolog) unless stated otherwise)
- we consider only Herbrand interpretations (= syntactical interpretation of terms)

Motivation for Deductive Rules

- *declarative* rule language based on first-order logic
- extends SQL/RA by *recursion* (& various types of negation)
- clear and concise representation of *knowledge*
- unifies
  - (recursive) **queries**
  - (static) **integrity constraints**

Some Major Research Issues

- handling/meaning of programs with negation (as failure)
  (min-mod, stratified, well-founded, stable)
- updates and change (deductive rules do no “speak” about change)
- efficient implementation
  (top-down vs bottom-up partially reconciled: magic sets, SLG resolution)
- extensions
  (disjunctions, explicit negation, constraints, ...)

Syntax: Datalog$^{\neg}$

Schema: emp(EmpNo, Salary, DeptNo), dept(DeptNo, MgrNo)

Queries/Views:

highpaid(E)	$\leftarrow$	emp(E,S,_), S > 50000	*% two simple*
boss(E,B)	$\leftarrow$	emp(E,_,D), dept(D,B)	*% ... views, and*
bom(X,Y)	$\leftarrow$	part(X,Y)	*% ... a recursively*
bom(X,Y)	$\leftarrow$	part(X,Z),bom(Z,Y)	*% ... defined one.*
p(X,Y)	$\leftarrow$	q(X), r(Y)	*% represent regular data concisely*

Integrity Constraints:

false	$\leftarrow$	emp(E,_,D), $\neg$ dept(D,_)	*% emp[DeptNo] $\subseteq$ dept[DeptNo]*
false	$\leftarrow$	dept(D,M), $\neg$ emp(M,_,_)	*% dept[MgrNo] $\subseteq$ emp[EmpNo]*

NB: the anonymous variable "_" is viewed as **existentially** quantified in the innermost context, i.e., $\neg$ dept(D,_) $\Leftrightarrow$ $\neg\,\exists$M: dept(D,M)

Starting point:

Read the program P:

$$\boxed{\begin{array}{l} \text{human(plato)} \leftarrow \\ \text{human(sokrates)} \leftarrow \\ \text{mortal(X)} \leftarrow \text{human(X)} \end{array}}$$

as the logic formula

$$\boxed{human(plato) \land human(sokrates) \land \forall X \; (mortal(X) \leftarrow human(X))}$$

⊕ In every model of P we have $mortal(plato)$ and $mortal(sokrates)$ (classical consequence relation: $\{Ax_i\} \models F$ if *every* model of $\{Ax_i\}$ is also a model of F)

⊖ We are interested in a single "intended" model, but there are unintended models M, e.g. where $M \models mortal(zeus)$

$\Rightarrow$ look at **minimal models**

Let P be a Datalog program, e.g.:

```
tc(X,Y)  ←  e(X,Y)
tc(X,Y)  ←  e(X,Z), tc(Z,Y)
```

Fixpoint/Production Rule Semantics: Let $\mathcal{I}_0 := \emptyset$, $\mathcal{I}_{i+1} := T_P(\mathcal{I}_i)$, and $\mathcal{I}_\omega := \bigcup_{n \geq 0} \mathcal{I}_n$:

$$lfp(T_P) = \mathcal{I}_\omega$$

For finite structures (Datalog) there is some $n \in \mathbb{N}$ s.t.

$$\mathcal{I}_\omega = \mathcal{I}_n = \mathcal{I}_{n+1} = \cdots$$

Minimal Model Semantics

$$\mathcal{M}_P := \bigcap_{M \models P} M \qquad \text{(intersection of all Herbrand models)}$$

Theorem. For positive programs: $\mathcal{M}_P = lfp(T_P) = \mathcal{I}_\omega$

Why consider negation (in rules)?

- at least SQL/FO queries should be expressible:
  - $P := Q \setminus R \quad \widehat{=} \quad$ p(X) $\leftarrow$ q(X), $\neg$ r(X)
  - $P \subseteq Q? \quad \widehat{=} \quad \forall X (p(X) \rightarrow q(X)) \quad \widehat{=}$
    p_sub_q $\leftarrow$ $\neg$ not_p_sub_q
    not_p_sub_q $\leftarrow$ p(X), $\neg$ q(X)
- close the language under negation:
  if $\mathcal{M}_P \models \neg\, tc(x, y)$ it should be possible to write ntc(X,Y) $\leftarrow$ $\neg$ tc(X,Y)
- nonmonotonic reasoning in general:
  flies(X) $\leftarrow$ bird(X), $\neg$ abnormal(X)
  bird(tweety) $\leftarrow$
  Thus $P \mid\!\sim$ flies(tweety), but $P \cup$ {abnormal(tweety)} $\not\mid\!\sim$ flies(tweety)

How to derive negative information?

General idea: read "$\neg\, A$" as "A cannot be proven" (*negation as failure*, *CWA*)

For positive programs: If $A \in \mathcal{B}_P \setminus \mathcal{M}_P$ then conclude $\neg\, A$

Logic programming drosophila:

P_1: $\boxed{\mathsf{p} \leftarrow \neg\, \mathsf{q}}$

P_2: $\boxed{\mathsf{p} \leftarrow \neg\, \mathsf{p}}$

P_3: $\boxed{\begin{array}{lcl} \mathsf{p} & \leftarrow & \neg\, \mathsf{q} \\ \mathsf{q} & \leftarrow & \neg\, \mathsf{p} \end{array}}$

Proposals/Solutions:

- ostrich policy: *"negative cycles don't occur"*; however they do: games, argumentations, ...
- **minimal models**: $\{q, \neg\, p\}$ for P_1 and $\{p\}$ for P_2 are minimal but not supported!
- avoidance: **stratified semantics** *"no recursion through negation"* $\Rightarrow$
  $\Rightarrow$ unique intended model (generally accepted) for programs like P_1
- **well-founded semantics**: unique 3-valued model, extends stratified semantics:
  if no well-founded derivation of p and $\neg\, p$ exists $\Rightarrow$ set interpret p (and $\neg\, p$) as **undefined**
- **stable models**:P_2 has no stable model, P_3: $\{p, \neg\, q\}$ and $\{q, \neg\, p\}$

Semipositive Datalog = Datalog$^\neg$ where only **extensional** (edb) relations may appear negated:

P: p(X) $\leftarrow$ q(X), $\neg$ r(X) *% input (edb): q,r; output (idb): p*

($\neg$) $\uparrow$

$\mathcal{D}$: q(a). q(b). r(a).

Generalization: **Stratified Datalog$^\neg$**

P_2: ntc(X,Y) $\leftarrow$ $\neg$ tc(X,Y) *% in: tc; out: ntc*

$\neg$ $\uparrow$

P_1: tc(X,Y) $\leftarrow$ e(X,Y) *% in: e; out: tc*

tc(X,Y) $\leftarrow$ e(X,Z),tc(Z,Y)

$\uparrow$

$\mathcal{D}$: e(a,b). e(b,c). *% out: e*

A rule R_1 **depends** (**negatively**) on a rule R_2 if

$$\begin{array}{llcl} R_1: & \ldots & \leftarrow & \ldots, (\neg)\, q(\ldots), \ldots \\ R_2: & q(\ldots) & \leftarrow & \ldots \end{array}$$

A program P is called **stratified** if there is a level-mapping $l: \; P \rightarrow \mathbb{N}$ s.t.

$l(R_1) > l(R_2)$ if R_1 depends **negatively** on R_2
$l(R_1) \geq l(R_2)$ if R_1 depends on R_2

and **locally stratified** if `ground`(P) is stratified.

% locally stratified (not stratified):		
even(0)	$\leftarrow$	
even(succ(X))	$\leftarrow$	$\neg$ even(X)

% not locally stratified:			
p(X)	$\leftarrow$	$\neg$ p(X)	*% a simple negative cycle*
even(0)	$\leftarrow$		*% unfounded negative*
even(X)	$\leftarrow$	$\neg$ even(succ(X))	*% ... dependencies*

If P is (locally) stratified

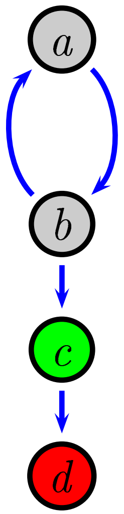
- rules can be partitioned into strata $P_0 \prec P_1 \prec P_2 \prec \cdots$ s.t.
  - the rules of P_i depend only on P_j for $j \leq i$
  - within P_i there are no negative dependencies
- if there are only finitely many rules (Datalog$^\neg$):
  P is stratified $\Leftrightarrow$ no rule of P depends negatively on itself
- there is a **unique intended model** $\mathcal{M}_P^{supp}$ [ABW88]

$\Rightarrow$ problem of finding the intended model is solved for (locally) stratified programs!

Consider the program P :

$$win(X) \leftarrow move(X, Y), \neg\, win(Y)$$

P describes a **game** between two players I and II: I (II) cannot move $\Rightarrow$ II (I) wins!



move relation

$win(d)$ is	**false**	(d is lost)
$win(c)$ is	**true**	(c is won)
$win(a), win(b)$ are	**undef**	(a, b are drawn)

$$win_{i+1}(X) \quad \leftarrow \quad move(X,Y), \neg\, win_i(Y)$$

Crux: if win_i is "too small" (underestimate) $\Rightarrow$ win_{i+1} "too big" (overestimate) and vice versa!

underestimates		overestimates	
i	$win_i(x)$	$i+1$	$win_{i+1}(x)$
0	$\{\neg\, a, \neg\, b, \neg\, c, \neg\, d\}$	1	$\{a, b, c, \neg\, d\}$
2	$\{\neg\, a, \neg\, b, c, \neg\, d\}$	3	$\{a, b, c, \neg\, d\}$
4	$\{\neg\, a, \neg\, b, c, \neg\, d\}$	5	$\{a, b, c, \neg\, d\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- atoms true in the underestimate, false in overestimate $\Rightarrow$ definite knowledge
- underestimates $\mathcal{I}_{2k}$ converge to $\mathcal{I}^{\textbf{true}}$ from below
- overestimates $\mathcal{I}_{2k+1}$ converge to $\mathcal{I}^{\textbf{true}} \cup \mathcal{I}^{\textbf{undef}}$ from above

Let P be a ground instantiated program.

1. Assume (guess) an interpretation $\mathcal{I} \subseteq \mathcal{B}_P$ for P.
2. Replace every **negative** literal by its truth-value wrt. $\mathcal{I} \Rightarrow$ positive program $P/\mathcal{I}$ (Stable reduction/ Gelfond-Lifschitz transformation of P wrt. $\mathcal{I}$ [GL88])
3. $P/\mathcal{I}$ has a unique minimal model $\mathcal{M}_{P/\mathcal{I}}$.
4. $\mathcal{I}$ is **stable model** of P if $\mathcal{I} = \mathcal{M}_{P/\mathcal{I}}$ (the set of beliefs $\mathcal{I}$ "reproduces" itself)

Example: P:

p	$\leftarrow$	$\neg$ q
q	$\leftarrow$	q, $\neg$ p

1. $\mathcal{I} = \{p, \neg\, q\}$
2. $P/\mathcal{I}$: p $\leftarrow$ **true**
q $\leftarrow$ q, **false**
3. $\mathcal{M}_{P/\mathcal{I}} = \{p, \neg\, q\}$
4. $\Rightarrow \mathcal{I}$ is stable!

Q: what about $\mathcal{I} = \{q, \neg\, p\}$?

P_1: $\boxed{\mathsf{p} \leftarrow \neg\, \mathsf{q}}$

P_2: $\boxed{\mathsf{p} \leftarrow \neg\, \mathsf{p}}$

P_3: $\boxed{\begin{array}{lcl} \mathsf{p} & \leftarrow & \neg\, \mathsf{q} \\ \mathsf{q} & \leftarrow & \neg\, \mathsf{p} \end{array}}$

Win-Move Example:

Stable Models		Well-Founded Model		
true	**false**	**true**	**false**	**undef**
win(a), win(c)	win(b), win(d)	win(c)	win(d)	win(a), win(b)
win(b), win(c)	win(a), win(d)	—	—	—

Well-founded semantics:

- unique 3-valued model [VGRS88, VG89]
- extends the stratified semantics
- generally accepted as "sceptical" semantics
- complexity for Datalog$^{\neg}$: **PTIME-complete** (expresses **Fixpoint**)
- normal form for Datalog$^{\neg}$: $win(\bar{X}) \leftarrow move(\bar{X}, \bar{Y}), \neg\, win(\bar{Y})$ [FKL97]

Stable semantics: [GL88]

- set of stable models (possibly empty)
- every stable model (provided there is one!) extends the well-founded semantics
- "brave" semantics
- complexity for Datalog$^{\neg}$ [MT91, Sch95]
  - credulous reasoning (derive A if A holds in **some** stable model): **NP-complete**
  - cautious reasoning: (derive A if A holds in **all** stable models) **coNP-complete**
  - concise, elegant representation of hard problems (planning, optimization, ...)

(Active) Production rule semantics

⊕ (re)active behavior, especially: updates

⊕ *enforcement* of static & dynamic integrity constraints

⊖ procedural semantics, handling of negation

⊖ termination not guaranteed

Deductive semantics

⊕ programs have a "declarative" logical semantics:
  - positive programs: minimal model semantics (= least fixpoint)
  - stratified ("layered") negation: stratified semantics
  - negative cyclic dependencies: well-founded, stable semantics

⊕ *checking* of static integrity constraints

⊖ static description of the modeled world: the intended model(s) correspond to single state (*snapshot*) of the modeled world!

⊖ no updates

$\Rightarrow$ integrate active features of production rules with declarative semantics of deductive rules

$\Rightarrow$ **Statelog** (= **State**s + Data**log**)

References

[ABW88] K. R. Apt, H. Blair, and A. Walker. Towards a Theory of Declarative Knowledge. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89 – 148. Morgan Kaufmann, 1988.

[FKL97] J. Flum, M. Kubierschky, and B. Ludäscher. Total and Partial Well-Founded Datalog Coincide. In F. Afrati and P. Kolaitis, editors, *Proc. 6th Intl. Conference on Database Theory (ICDT)*, volume 1186 of *LNCS*, pages 113–124, Delphi, Greece, 1997. Springer.

[GL88] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *Proc. Intl. Conference on Logic Programming*, pages 1070–1080, 1988.

[MT91] V. W. Marek and M. Truszczynski. Autoepistemic Logic. *Journal of the ACM*, 38(3):588–619, 1991.

[Sch95] J. S. Schlipf. Complexity and Undecidability Results in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 15(III-IV), 1995.

[VG89] A. Van Gelder. The Alternating Fixpoint of Logic Programs with Negation. In *Proc. ACM Symposium on Principles of Database Systems*, pages 1–10, 1989.

[VGRS88] A. Van Gelder, K. Ross, and J. Schlipf. Unfounded Sets and Well-Founded Semantics for General Logic Programs. In *Proc. ACM Symposium on Principles of Database Systems*, pages 221–230, 1988.