Production Rules

Contents

- Introduction: Production Rules as Active Rules
- Example: Integrity Enforcement
- Datalog with Fixpoint Semantics (Datalog, . . ., Datalog$^{\neg\neg}$)
- Expressive Power, Complexity
- Production Rules vs Deductive Rules: Handling of Negation

The active rules considered so far have very complex, procedural aspects (cf. execution model, coupling modes, ...)

$\Rightarrow$ conceptually simpler **production rules** of the form:

$$\textbf{if } \langle condition \rangle \textbf{ then } \langle action \rangle$$

- $\langle condition \rangle$ is a query matched against the *working memory* (current database state)
- $\langle action \rangle$ are internal (database updates) or external operations

Other terms: *production system* (= rules + working memory + inference engine), *condition-action rules*, *forward-chaining rules*

Basic evaluation procedure: (*recognize-act-cycle*)

match (find eligible rules $\Rightarrow$ candidate set)
while $\langle$*matched rules* $\neq \emptyset\rangle$ do {
1. select a rule (conflict resolution)
2. execute (act)
3. match

}

Languages/Systems:

- AI community (main memory based system):
  OPS-5 (instance-oriented) [**?**]
- adaption to databases:
  ARIEL (set-oriented) [Han96]
  (uses *discrimination networks*: Rete [For82], TREAT [Mir87], A-TREAT [WH92, Han92])
- RDL1 [dMS88, KdMS90], A-RDL (Active RDL = RDL with **delta relations**) [SKdM92, SK96]

$\Rightarrow$ "pure" version: Datalog$^{\neg\neg}$ with procedural (fixpoint) semantics [AV91]

Motivation

- concise, elegant query language
- extends SQL/FO by recursion
- investigation of expressive power, complexity (relation to *finite model theory*)
- updates ($\Rightarrow$ programmable reactive behavior $\Rightarrow$ active rules)

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

Inclusion dependencies: emp[DeptNo] $\subseteq$ dept[DeptNo], dept[MgrNo] $\subseteq$ emp[EmpNo]

Active Rule: (pidgin language)

on	delete from emp
if	E in emp.deleted **and** E manages D **and** E1 works for D
then	delete D from dept, delete E1 from emp

Production Rule: $\langle action \rangle \leftarrow \langle condition \rangle$

% external events $\triangleright e(\bar{X})$ are mapped to db-relations:
del:emp(E,S,D) $\leftarrow$ $\triangleright$**del**:emp(E,S,D)

% main rule:
del:dept(D,E), **del**:emp(E1,Sal1,D) $\leftarrow$ **del**:emp(E,_,_), dept(D,E), emp(E1,Sal1,D)

% frame rules:
$\neg$ dept(D,E) $\leftarrow$ **del**:dept(D,E)
$\neg$ emp(E,S,D) $\leftarrow$ **del**:emp(E,S,D)

▷ **del**:emp(md1, 10000,d1) $\leftarrow$ *% external event has occurred!*

del:dept(D,EM), **del**:emp(E1,Sal1,D) $\leftarrow$ **del**:emp(EM,_,_), dept(D,EM), emp(E1,Sal1,D)

$\neg$ dept(D,E) $\leftarrow$ **del**:dept(D,E)
$\neg$ emp(E,S,D) $\leftarrow$ **del**:emp(E,S,D)

emp

EmpNo	*Salary*	*Dept*
ed1	7000	d1
md1	10000	d1
md2	1000	d1
ed2	6000	d2
md2	8000	d2
md3	9000	d3

dept

DeptNo	*Manager*
d1	md1
d2	md2
d3	md3

⋮

emp

EmpNo	*Salary*	*Dept*
md3	9000	d3

dept

DeptNo	*Manager*
d3	md3

Syntax

- A **Datalog**$^{\neg}$ rule is an expression of the form

$$H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m$$

  where H, B_i, C_j are atoms
- $m = 0 \Rightarrow$ **Datalog** rule
- $n, m = 0 \Rightarrow$ **fact** (i.e., empty body evaluates to **true**)
- H possibly negated atom $\neg A \Rightarrow$ **Datalog**$^{\neg\neg}$ rule
- a Datalog/Datalog$^{\neg}$/Datalog$^{\neg\neg}$ **program** P is a finite set of Datalog/Datalog$^{\neg}$/Datalog$^{\neg\neg}$ rules
- a **database** $\mathcal{D}$ is a finite set of facts

Interpretations

Only **Herbrand interpretations** ("syntactical interpretations") are considered:

- **Herbrand universe** $\mathcal{U}_P := \{a, b, c, \ldots\}$ (constants of P)
- **Herbrand base** $\mathcal{B}_P := \{p(a,a), p(a,b), q(a), \ldots\}$ (*ground atoms* constructible from relation symbols and constants of P)
- **Herbrand interpretation** $\mathcal{I} \subseteq \mathcal{B}_P$: those atoms which are interpreted as **true** ($\mathcal{B}_P \setminus \mathcal{I} \Rightarrow$ **false** atoms)

Let P be a Datalog$^\neg$ program **including** a database $\mathcal{D}$ as a set of facts, i.e. $P = P' \cup \mathcal{D}$.

The operator T_P [vEK76, CH85] maps a database $\mathcal{I}$ to

$$T_P(\mathcal{I}) := \{head(r) \mid \mathcal{I} \models body(r), r \in \mathtt{ground}(P)\}$$

i.e., to the **immediate consequences** of $\mathcal{I}$.

Thus every program P (with database $\mathcal{D}$) induces a sequence of databases $\mathcal{I}_0, \mathcal{I}_1, \ldots$ by setting

$$\mathcal{I}_0 := \emptyset, \qquad \text{and} \qquad \mathcal{I}_{i+1} := T_P(\mathcal{I}_i)$$

If $\mathcal{I}_0 \subseteq \mathcal{I}_1 \subseteq \cdots$ then the sequence is called **inductive** and reaches the least fixpoint

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

(note that we consider **finite** structures)

P:

P':	tc(X,Y)	$\leftarrow$	e(X,Y).	$\mathcal{D}$:	e(a,b). e(b,c). e(c,b).
	tc(X,Y)	$\leftarrow$	e(X,Z), tc(Z,Y).		

stage i	$\mathcal{I}_i$
0	$\emptyset$
1	e
2	$e \cup \{tc(a,b), tc(b,c), tc(c,b)\}$
3	$e \cup \{tc(a,b), tc(b,c), tc(c,b), tc(a,c), tc(b,b), tc(c,c)\}$
4	$e \cup \{tc(a,b), tc(b,c), tc(c,b), tc(a,c), tc(b,b), tc(c,c)\}$
$\vdots$	$\vdots$

where $e = \{e(a,b), e(b,c), e(c,b)\}$ (new facts are displayed in green)

NB: if $tc(x,y) \in \mathcal{I}_{d+1} \setminus \mathcal{I}_d$ then $|x\ y| = d$ (= shortest distance from x to y)

When is the sequence $\langle \mathcal{I}_n \rangle$ inductive?

- If P is a **Datalog** program, then $\mathcal{I} \subseteq \mathcal{J}$ implies $T_P(\mathcal{I}) \subseteq T_P(\mathcal{J})$ (i.e., T_P is **monotone**)
  $\Rightarrow \langle \mathcal{I}_n \rangle$ is inductive $\Rightarrow$ **least fixpoint semantics**
- However, if P is a **Datalog$^\neg$** program, T_P becomes **nonmonotonic:**

p_minus_q(X) $\leftarrow$ p(X), $\neg$ q(X).

Here, increasing q may decrease p_minus_q. Also, the least fixpoint may no longer exist:

p(X) $\leftarrow$ q(X), $\neg$ p(X).

Given the database $\mathcal{D} = \{q(a)\}$ this gives rise to the sequence

$$\emptyset, \{q(a)\}, \{q(a), p(a)\}, \{q(a)\}, \{q(a), p(a)\}, \ldots$$

$\Rightarrow$ the sequence $\langle \mathcal{I}_n \rangle$ may oscillate $\Rightarrow$ NI-Datalog$^\neg$ (**noninflationary** Datalog$^\neg$) semantics [AV91]

One can enforce termination by turning T_P into an **inflationary** operator [AV88, KP88]:

$$T_P^+(\mathcal{I}) := \mathcal{I} \cup T_P(\mathcal{I})$$

$\Rightarrow \mathcal{I} \subseteq T_P^+(\mathcal{I}) \Rightarrow T_P^+$ is inductive and the sequence $\langle \mathcal{I}_n \rangle$ defined using T_P^+ reaches a fixpoint

$\Rightarrow$ I-Datalog$^\neg$ (**inflationary** Datalog$^\neg$) semantics

Example:

For $\mathcal{D} = \{q(a)\}$ and P: p(X) $\leftarrow$ q(X), $\neg$ p(X).

we get

$$\emptyset, \{q(a)\}, \{q(a), p(a)\}, \{q(a), p(a)\}, \ldots$$

Basic idea: interpret negation in the head as **deletion**:

$$\begin{aligned} T_P^{ins}(\mathcal{I}) &:= \{H \mid \mathcal{I} \models body, (H \leftarrow body) \in \mathtt{ground}(P))\} \\ T_P^{del}(\mathcal{I}) &:= \{H \mid \mathcal{I} \models body, (\neg H \leftarrow body) \in \mathtt{ground}(P))\} \end{aligned}$$

Using these, several variants of $T_P^{\neg\neg}$ can be defined [AV91]:

$$\begin{aligned} T_P^{\neg\neg}(\mathcal{I}) &:= (\mathcal{I} \cup T_P^{ins}(\mathcal{I})) \setminus T_P^{del}(\mathcal{I}) && \textit{\% priority to deletions} \\ T_P^{\neg\neg}(\mathcal{I}) &:= (\mathcal{I} \setminus T_P^{del}(\mathcal{I})) \cup T_P^{ins}(\mathcal{I}) && \textit{\% priority to insertions} \end{aligned}$$

other variants exist for handling $T_P^{ins} \cap T_P^{del} \neq \emptyset$, e.g. keep/complement the previous value, “abort”

$\Rightarrow$ **Expressive power** of the various variants?

Query: *generic* mapping from instances of an *input schema* to instances of an *output schema*:

transitive closure:

$$\begin{array}{rcl} q_{tc}: \quad \mathtt{inst}(e/2) & \rightarrow & \mathtt{inst}(tc/2) \\ \{e(a,b), e(b,c)\} & \mapsto & \{tc(a,b), tc(b,c), tc(a,c)\} \\ & \vdots & \end{array}$$

Update: *generic* mapping between instances of the *same* schema:

delete 2-cycles:

$$\begin{array}{rcl} u_{del-2}: \quad \mathtt{inst}(e/2) & \rightarrow & \mathtt{inst}(e/2) \\ \{e(a,b), e(b,a), e(a,c)\} & \mapsto & \{e(a,c)\} \\ & \vdots & \end{array}$$

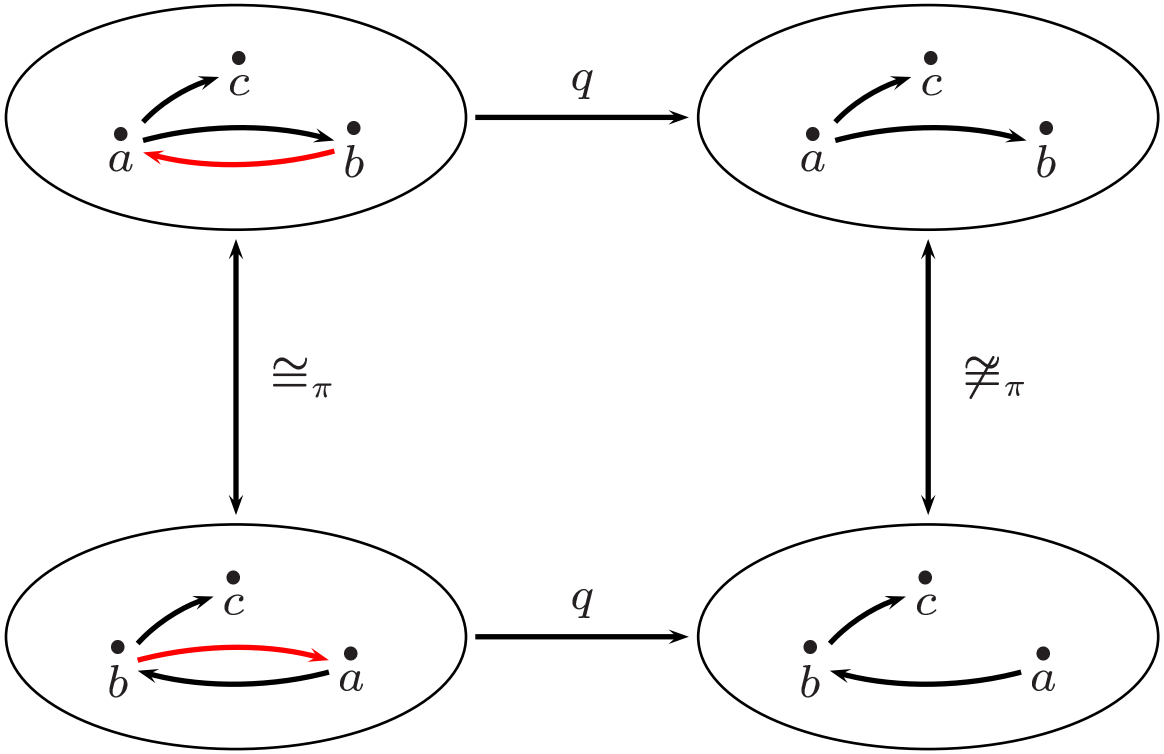
The **expressive power** of a language $\mathcal{L}$ is the class of db-mappings definable in $\mathcal{L}$.

- *"Isomorphic inputs are mapped to isomorphic outputs."*
- *"Names of constants is irrelevant."*

$\Rightarrow$ only "conceptual" information may be accessed.

Formally, the db-mapping q is generic if $q(\pi(\mathcal{D})) = \pi(q(\mathcal{D}))$ for all databases $\mathcal{D}$ and isomorphisms π of $\mathcal{D}$.

A mapping q which is **not generic:** "remove all edges pointing to node a":


c
a
b
q
c
a
b
$\cong_\pi$
$\not\cong_\pi$
c
b
a
q
c
b
a

- **Conjunctive Queries:** FO($\exists, \wedge$), RA($\sigma, \pi, \bowtie, \delta$), *Query-By-Example*

  $answer$(X,Z) $\leftarrow$ p(X,Y),q(Y,Z)
- **Positive-Existential:** FO($\exists, \wedge, \vee$), RA($\sigma, \pi, \bowtie, \delta, \cup$)

  $answer$(X,Z) $\leftarrow$ p(X,Y),q(Y,Z)

  $answer$(X,Z) $\leftarrow$ r(X,Z)
- **First-Order:** FO($\forall, \exists, \wedge, \vee, \neg$), RA($\sigma, \pi, \bowtie, \delta, \cup, \setminus$)
- **Datalog**

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

  tc(X,Y) $\leftarrow$ e(X,Z), tc(Z,Y)
- **stratified Datalog**$^{\neg}$
- **Fixpoint:** I-Datalog$^{\neg}$
- **While:** NI-Datalog$^{\neg}$, Datalog$^{\neg\neg}$
- **All:** NI-Datalog$^{\neg}{}_{new}$, Datalog$^{\neg\neg}{}_{new}$

- Computing with **order** or **nondeterministic extensions** has a great impact on expressive power (cf. [AHV95])

- **Data Complexity:** complexity of evaluating a **fixed** program P wrt. the size of the database $\mathcal{D}$
- **Expression (Query) Complexity:** complexity of evaluating a **fixed** database $\mathcal{D}$ wrt. the size of the programs P

In the database context: $|\mathcal{D}| \gg |P| \Rightarrow$ data complexity is more relevant.

- in **LOGSPACE:**
  conjunctive, positive-existential, FO
- in **PTIME:**
  Datalog, S-Datalog$^\neg$, I-Datalog$^\neg$
- in **PSPACE:**
  NI-Datalog$^\neg$, Datalog$^{\neg\neg}$

(cf. [AHV95]; recently: *parametric complexity*, cf. [PY97])

Summary

- simple, clear semantics
- external events $\triangleright e(\bar{X}) \rightsquigarrow$ db-events (relations)
- FO conditions
- db-actions (insert/delete requests)
- set-oriented (all rules are fired "in parallel")
  (there are instance-oriented variants)
- deterministic semantics $\Rightarrow$ confluence is guaranteed
  (for nondeterministic variants $\Rightarrow$ confluence is undecidable)
- always terminate (Datalog, I-Datalog$^{\neg}$)
  (no direct support for deletions $\Rightarrow$ not appropriate for active rules)
- may not terminate (NI-Datalog$^{\neg}$, Datalog$^{\neg\neg}$)
  - the class of NI-Datalog$^{\neg}$ programs P which terminate for **all** databases $\mathcal{D}$ is undecidable
  - there are decidable fragments having the same expressive power

Different "readings" (interpretations) of a rule

$$r: \qquad H \leftarrow B, \neg C$$

Production Rule: (**if** *condition* **then** *action*)

if r fires, H is derived. In the final state

- H may no longer hold, since $B, \neg C$ has become false:
  if *low stock* **then** *increase stock*
- H may still hold, although no rule supports this (inflationary semantics):
  $\neg C$ may become false **later**, yet the "rash (premature)" conclusion H is not withdrawn!

Deductive Rule: (**if** *premise* **then** *conclusion*)

$\approx$ "reasoning process": knowledge is only increased
$\Rightarrow$ conclusions (and hence rule conditions) remain valid

Crux: different handling of **negation**!

- Consider the program $\boxed{\mathsf{p} \leftarrow \neg\,\mathsf{p}}$
  - NI-Datalog$^\neg$: no fixpoint
  - I-Datalog$^\neg$: fixpoint $= \{p\}$, but not **supported** (no rule "justifies" p!)
- Does the following program compute the complement of transitive closure?

$$\boxed{\begin{array}{lcl} \mathsf{tc(X,Y)} & \leftarrow & \mathsf{e(X,Y)} \\ \mathsf{tc(X,Y)} & \leftarrow & \mathsf{e(X,Z),\ tc(Z,Y)} \\ \mathsf{ntc(X,Y)} & \leftarrow & \neg\,\mathsf{tc(X,Y)} \end{array}}$$

  - I-Datalog$^\neg$: no! ntc(X,Y) is "concluded" too early, i.e., before tc is completely "known"
  - NI-Datalog$^\neg$: yes! but adding the tautology "ntc(X,Y) $\leftarrow$ ntc(X,Y)" spoils the intended semantics

$\Rightarrow$ focus of production rules was not on "logical" reading of rules and "declarative" handling of negation

$\Rightarrow$ declarative semantics for Datalog$^\neg$ (logic programming)

What does this program compute?

$P_?$:

$$
\boxed{
\begin{array}{lcl}
\mathrm{tc(X,Y)} & \leftarrow & \mathrm{e(X,Y)} \\
\mathrm{tc(X,Y)} & \leftarrow & \mathrm{e(X,Z),tc(Z,Y)} \\
\mathrm{q(X,Y,\ X',Y')} & \leftarrow & \mathrm{tc(X,Y),\ \neg\, tc(X',Y')}
\end{array}
}
$$

In order to compute ntc with **inflationary** Datalog$^\neg$, the following program has to be used:

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

prev(X,Y)   ←  tc(X,Y)
prev_but_last(X,Y)   ←  tc(X,Y), e(X',Z'), tc(Z',Y'), ¬ tc(X',Y')
ntc(X,Y)   ←  ¬ tc(X,Y), prev(X',Y'), ¬ prev_but_last(X',Y')
```