# Nested Transactions in a Logical Language for Active Rules

Bertram Ludäscher  Wolfgang May  Georg Lausen

Institut für Informatik, Universität Freiburg, Germany

## Overview

- Introduction: Deductive vs Active Rules
- Flat Statelog and Friends: Unifying Active and Deductive Rules
- Procedures and Nested Transactions
- Conclusion and Outlook

**Deductive Rules** (aka Datalog$^{\neg}$)

+ powerful query language, "declarative" semantics

+ IC *checking*

– static description of the modeled world ⇒ no support for updates and (re)active behavior

**Active Rules**

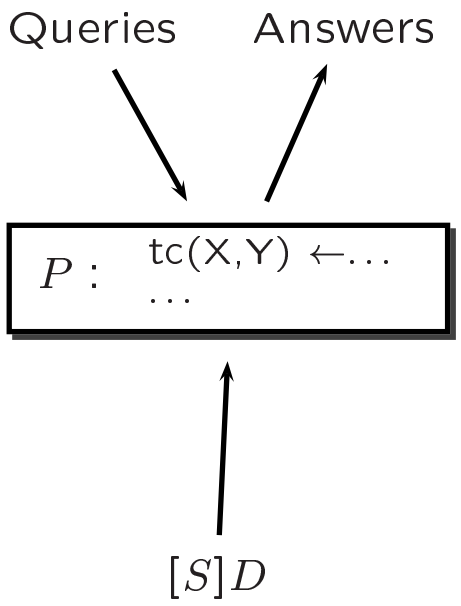
+ (re)active behavior, especially:updates

+ view maintenance, IC *enforcement*, monitoring applications, ...

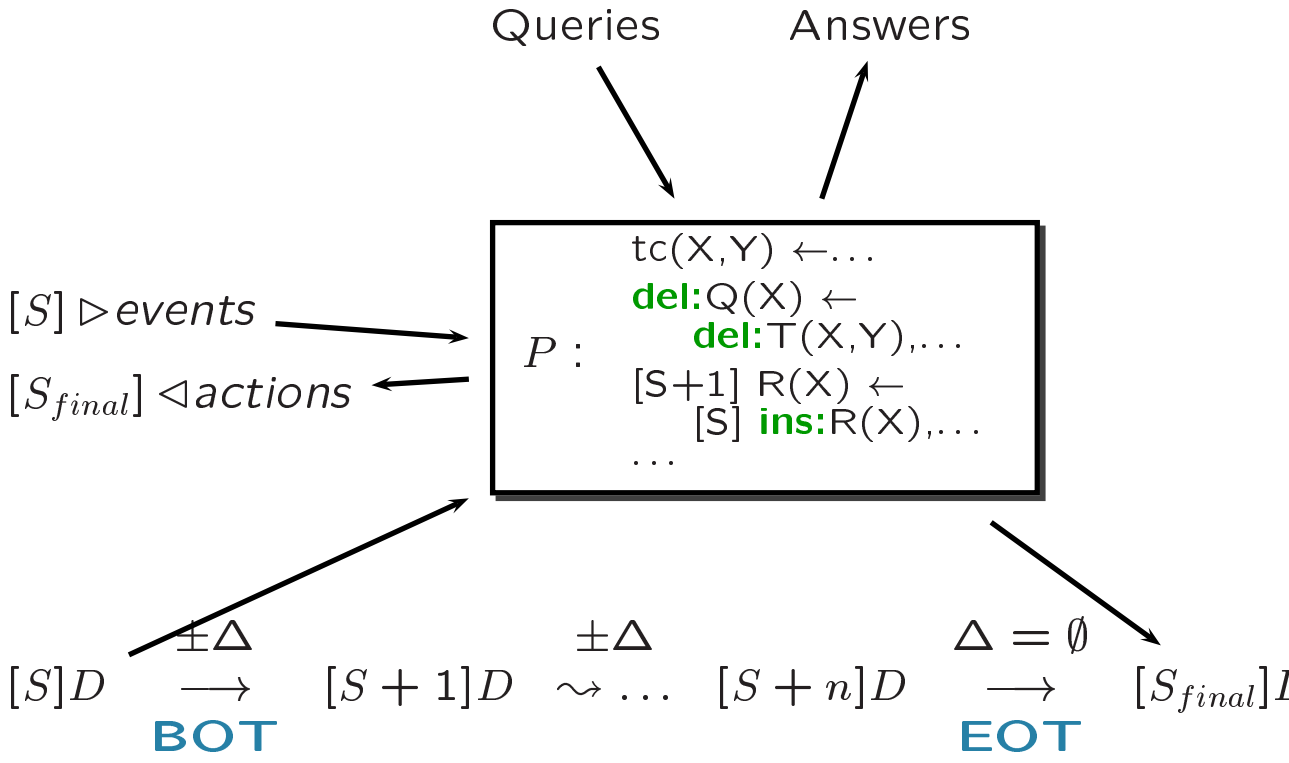
– semantics, predictability of rule effects, termination

⇒ **a Unified Framework for Active and Deductive Rules**

| Update / Active Rules | = | Logic / Datalog$^{\neg}$ | + X |
|---|---|---|---|




**Datalog**




**(Flat) Statelog**

## User-Defined Rules

*Queries:* $[S]$ tc(X,Y) $\leftarrow$ tc(X,Z), tc(Z,Y).

*Integrity Constraints:* $[S]$ **abort** $\leftarrow$ ...

*Updates:*
*% emp.D REFERENCES dept.D ON DELETE CASCADE*
$[S]$ **del:**emp(E,Sal,D) $\leftarrow$ **del:**dept(D,_), emp(E,Sal,D).

## System-Defined Rules

*Frame Rules:*
$[S+1]$ R(X) $\leftarrow$ $[S]$ R(X), **not** **del:**R(X).
$[S+1]$ R(X) $\leftarrow$ $[S]$ **ins:**R(X).

*Integrity Constraints:* $[S]$ (**abort** $\leftarrow$ **ins:**R(X), **del:**R(X)).

## Standard LP Semantics

$[S+1]$ p(X) $\leftarrow$ $[S]$ **not** q(Y) $\equiv$ p(succ(S),X) $\leftarrow \neg$ q(S,Y)
$\equiv$ $\Box$ ($\circ$ p(X) $\leftarrow \neg$ q(Y))

## Expressiveness/Complexity

- Flat Statelog $\equiv$ While/PFP ($\equiv$ PSPACE on ordered DBs)
- $\Delta$-monotone Statelog $\equiv$ Fixpoint/LFP ($\equiv$ PTIME on ordered DBs) (termination guaranteed)
- Guarded Statelog $\equiv$ Stratified Datalog ($\equiv$ PTIME on ordered DBs)

## Related Approaches

*XY-Datalog* [Zaniolo], *Datalog$_{1S}$/Templog* [Chomicki/Baudinet],
*ELS-Datalog* [Kemp-Ramamohanarao-Stuckey],
*Datalog$^{\neg\neg}$* [Abiteboul-Vianu],
*Heraclitus[Alg,C]* [Ghandeharizadeh et.al.]
...

Hire employee $E$ with salary *Sal* for department *Dept* provided the average salary *after* the update does not exceed a certain limit:

[$S$] **ins:**empsal(E,Sal),**ins:**empdep(E,Dept), [$S+1$] checksal(Dept) $\leftarrow$
    [$S$] $\triangleright$hire(E, Sal, Dept).
[$S$] check_ok $\leftarrow$ checksal(Dept), avg(Dept, Amt), Amt $<$ 50000.
[$S+1$] **del:**empsal(E,Sal),**del:**empdep(E,Dept) $\leftarrow$
    [$S$] $\triangleright$hire(E,Sal,Dept), [$S+1$] **not** check_ok.

## Problems

- Undoing the effect of changes has to be programmed explicitly.
- There is no structure which allows grouping of semantically closely related rules.
- The effects of *ephemeral updates* [zaniolo-DOOD-95], and *hypothetical updates* are visible to other rules, since there is no encapsulation.

$\Rightarrow$ group rules into **procedures** which execute as **nested transactions**

- A *Statelog procedure* $\pi$ is a set of local Statelog rules.
- A procedure $\pi$ defines a *transaction* $T_\pi$ at runtime. The behavior of $\pi$ is encapsulated:
  - If $\pi$ calls $\mu$, then $T_\mu$ executes as *subtransaction*.
  - Simultaneously called procedures $\pi$ and $\rho$ execute *independently* and in *isolation*.
- (Sub-)Transactions execute *atomically* (*all-or-nothing*).

## Procedures

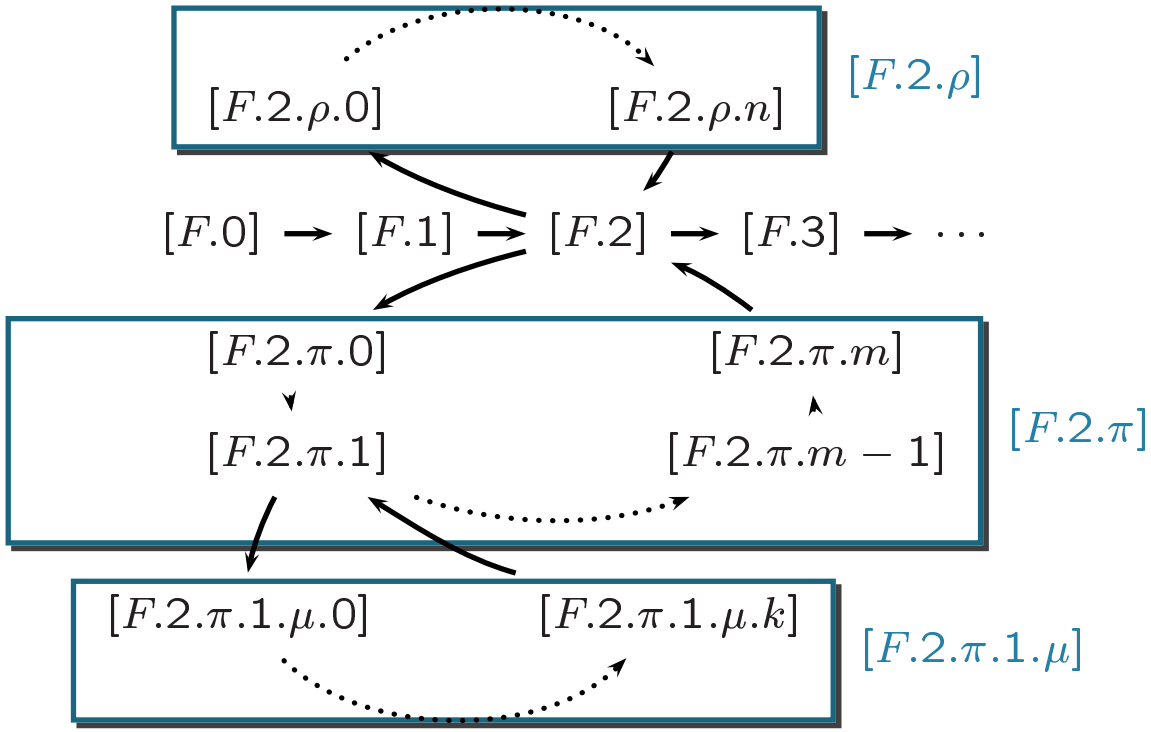
**proc** $\pi(A_1, \ldots, A_n)$; $\nabla I_1, \ldots, I_k$; $\Delta O_1, \ldots, O_l$;
**initial:** $P_{initial}(\pi)$; **always:** $P_{always}(\pi)$; **final:** $P_{final}(\pi)$
**endproc**

$\nabla R$: import EDB/IDB relation $R$ $\Delta R$: export EDB relation $R$

## Rules

| | | |
|---|---|---|
| *Views:* | $[S]\ V(\bar{X}) \leftarrow \ldots$ | $V \in IDB(P)$ |
| *Change Requests:* | $[S]$ **ins:**$R(\bar{X}) \leftarrow \ldots$ | |
| | $[S]$ **del:**$R(\bar{X}) \leftarrow \ldots$ | $R \in EDB(P)$ |
| | $[S]$ **mod:**$R(\bar{X}/\bar{X}') \leftarrow \ldots$ | |
| *Procedure Calls:* | $[S]\ \pi(\bar{X}) \leftarrow \ldots$ | $\pi \in Proc(P)$ |
| *External Actions:* | $[S]\ \triangleleft A(\bar{X}) \leftarrow \ldots$ | $\triangleleft A \in Act(P)$ |
| *Transaction Control:* | $[S]$ **abort** $\leftarrow \ldots$ | **abort** $\in Ctl(P)$ |

$\Rightarrow$ all user-defined rules are *local*.




(Transaction) Frames:
$[F]$, $[F.2.\rho]$, $[F.2.\pi]$, ...

States:
$[F.n]$, $[F.2.\rho.n]$, $[F.2.\pi.m]$, ..

*Deltas:*

$[S+1]\ R(\bar{X}) \leftarrow [S]$ **ins:**$R(\bar{X})$, **not** **EOT**.
$[S+1]\ R(\bar{X}) \leftarrow [S]\ R(\bar{X})$, **not** **del:**$R(\bar{X})$, **not** **EOT**.

*Protocol Relations:*

$[S+1]$ **insd:**$R(\bar{X}) \leftarrow [S]$ **ins:**$R(\bar{X})$, **not** **EOT**.
$[S+1]$ **insd:**$R(\bar{X}) \leftarrow [S]$ **insd:**$R(\bar{X})$, **not** **del:**$R(\bar{X})$, **not** **EOT**.
$[S+1]$ **deld:**$R(\bar{X}) \leftarrow [S]$ **del:**$R(\bar{X})$, **not** **EOT**.
$[S+1]$ **deld:**$R(\bar{X}) \leftarrow [S]$ **deld:**$R(\bar{X})$, **not** **ins:**$R(\bar{X})$, **not** **EOT**.

*Control:*

$[S]$ **running** $\leftarrow [S]$ **ins:**$R(\bar{X})$, **not** $R(\bar{X})$.
$[S]$ **running** $\leftarrow [S]$ **del:**$R(\bar{X}), R(\bar{X})$.

$[S]$ **EOT** $\leftarrow [S]$ **BOT**, **not** **running**.
$[S+1]$ **EOT** $\leftarrow [S]$ **running**, **not abort**, $[S+1]$ **not** **running**.

*Procedures:*

$[S.\pi(\bar{X}).0]$ **BOT** $\leftarrow [S]\ \pi(\bar{X})$.

$[S]$ **committed:**$\pi(\bar{X}) \leftarrow [S]\ \pi(\bar{X}), [S.\pi(\bar{X}).N]$ **EOT**, **not abort**.
$[S]$ **aborted:**$\pi(\bar{X}) \leftarrow [S]\ \pi(\bar{X}), [S.\pi(\bar{X}).N]$ **EOT**, **abort**.

. . .

- Statelog programs $P$ can be directly translated into logic programs (with function symbols):

$$[S+1]\ \mathsf{R(X)} \leftarrow [S]\ \textbf{ins:}\mathsf{P(Y)},\ \mathsf{Q(X,Y)}.$$

$$\mapsto$$

$$\mathsf{R}([S+1],\mathsf{X}) \leftarrow \textbf{ins:}\mathsf{P}([S],\mathsf{Y}),\ \mathsf{Q}([S],\mathsf{X},\mathsf{Y}),\ \textbf{state}([S]).$$

- Definition of states:

  $\textbf{state}([\varepsilon.0])$.
  $\textbf{state}([S.\pi(\bar{X}).0]) \leftarrow \pi([S],\bar{X})$.
  $\textbf{state}([S+1]) \leftarrow \textbf{state}([S]),\ \textbf{alive}([S])$.

- if finitely many states are created by $P \cup EDB \cup EB$ (**E**vent **B**ase), then $P$ *terminates*:

  $\textbf{alive}([S]) \leftarrow \textbf{BOT}([S])$.
  $\textbf{alive}([S+1]) \leftarrow \textbf{running}([S]),\ \textbf{not}\ \textbf{EOT}([S])$.

- In every frame $[F]$ there is at most one state $[F.n]$ s.t. $\mathcal{M}(P, EDB, EB) \models \textbf{EOT}([F.n]$

**proc** hire(E,Sal,Dept); ▽empsal,empdep; △empsal,empdep;
  **initial:** newemp(E,Sal,Dept) ⊗ checksal(Dept) ← .
  **always:** **abort**← **aborted:**checksal(Dept).
**endproc**
**proc** newemp(E,Sal,Dept); △empsal,empdep;
  **initial:** **ins:**empsal(E,Sal) ←.
  **ins:**empdep(E,Dept) ←.
**endproc**
**proc** checksal(Dept); ▽empdep,empsal;
  **initial:** **abort**← avg(Dept,Amt), **not** Amt<50000.
**endproc**

## Statelog

- "declarative" semantics for active rules and updates
- classes of terminating active rules
- increased expressive power: checking and *enforcing* of static and *dynamic* ICs
- **Integration and formalization of nested transactions:**

  (i) Logic programming semantics

  (ii) Kripke-style semantics: formalizes the *conceptual model* of nested transactions in Statelog.

## Outlook

- Expressive power with nested transactions
- Goal-oriented evaluation
- Implementation
- Reasoning about transactions:
  $(P, EDB, EB) \models \varphi_{IC}$ for all $EB$ and all $EDB$ *reachable via* $P$.

**Def.** A *Statelog Kripke structure* over signature $\Sigma$ is a tuple $\mathcal{K} = (\mathcal{G}, \mathcal{A}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{U}, \mathcal{M}, \mathcal{P})$, where

$\mathcal{G}$ is a set of states,

$\mathcal{A}$ (actions) is a set of procedure names,

$\mathcal{Q}, \mathcal{S} \subseteq \mathcal{G} \times \mathcal{A} \times \mathcal{U}^\omega \times \mathcal{G}$, are two marked accessibility relations between states representing the procedure-call resp. -return relation:

$\mathcal{Q}(g, \pi(\bar{x}), g')$: the first state of the subtransaction induced by $\pi(\bar{x})$ is $g'$.

$\mathcal{S}(g', \pi(\bar{x}), g)$: $g'$ is the final state of the subtransaction induced by $\pi(\bar{x})$ in $g$.

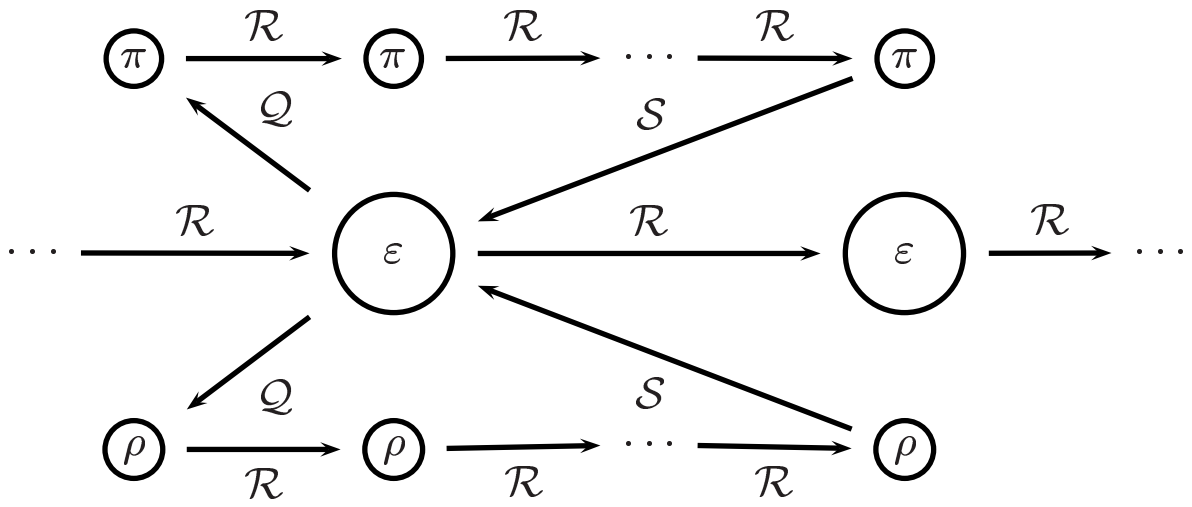
$\Rightarrow$ results of subtransactions are communicated via $\mathcal{S}$

$\mathcal{R} \subseteq \mathcal{G} \times \mathcal{G}$ models the temporal successor relation,

$\mathcal{U}$ is the universe of elements,

$\mathcal{M}$ maps states to first-order interpretations,

$\mathcal{P}$ is a function which maps every $g \in \mathcal{G}$ to a set of local rules (the rules visible in $g$).


π
π
π
ε
ε
ρ
ρ
ρ
$\mathcal{R}$
$\mathcal{Q}$
$\mathcal{S}$
$\cdots$

**Theorem (Adequacy).**

- *EDB relations are changed exactly via requests.*
- *Every state contains all requests contributed by subtransactions.*
- *IDB relations are derived locally by user-defined rules.*
- *Requests are derived by user-defined rules or contributed by subtransactions.*
- *In all states the protocol relations contain all non-revoked changes of the corresponding subtransactions.*