Integration of Active and Deductive Database Rules

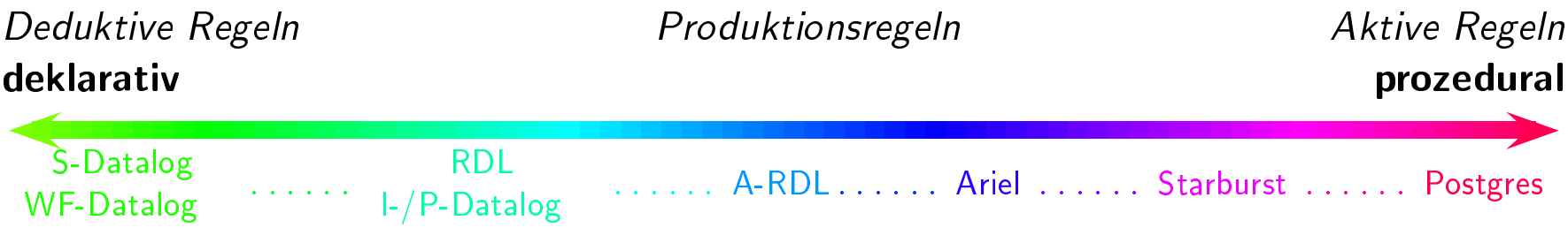
Bertram Ludäscher

Institut für Informatik, Universität Freiburg

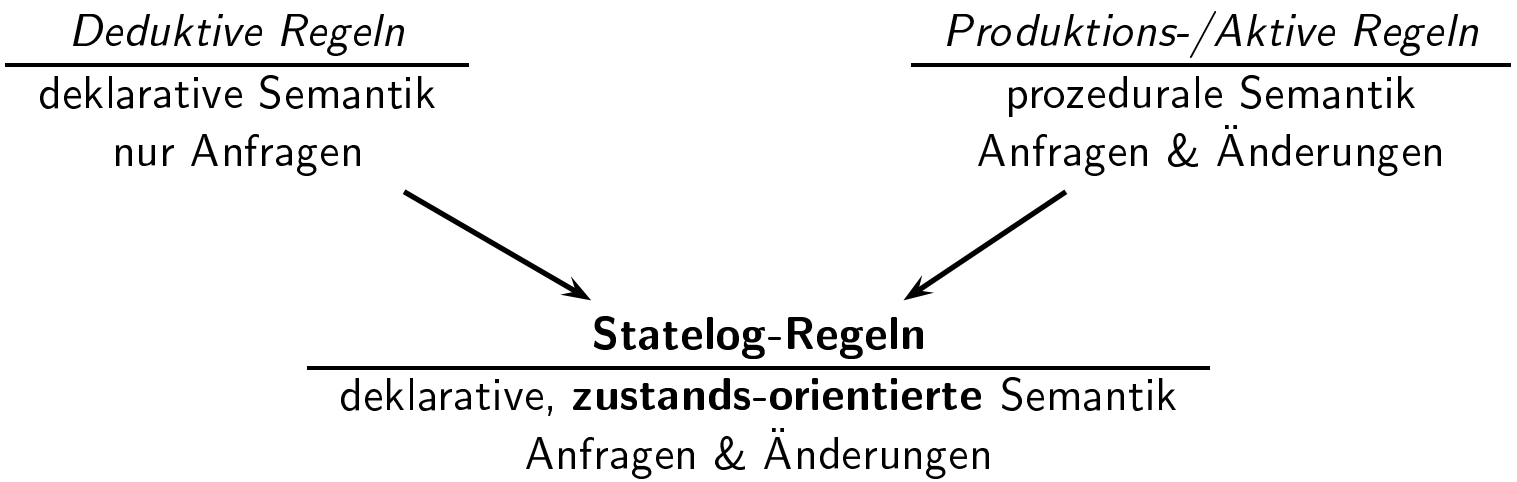
Übersicht

- Motivation; Spektrum von Regelsprachen: *Aktive Regeln—Produktionsregeln—Deduktive Regeln*
- Statelog
- Terminierung
- Normalformen $\Rightarrow$ Ausdrucksstärke, Komplexität
- Vergleich
- Zusammenfassung

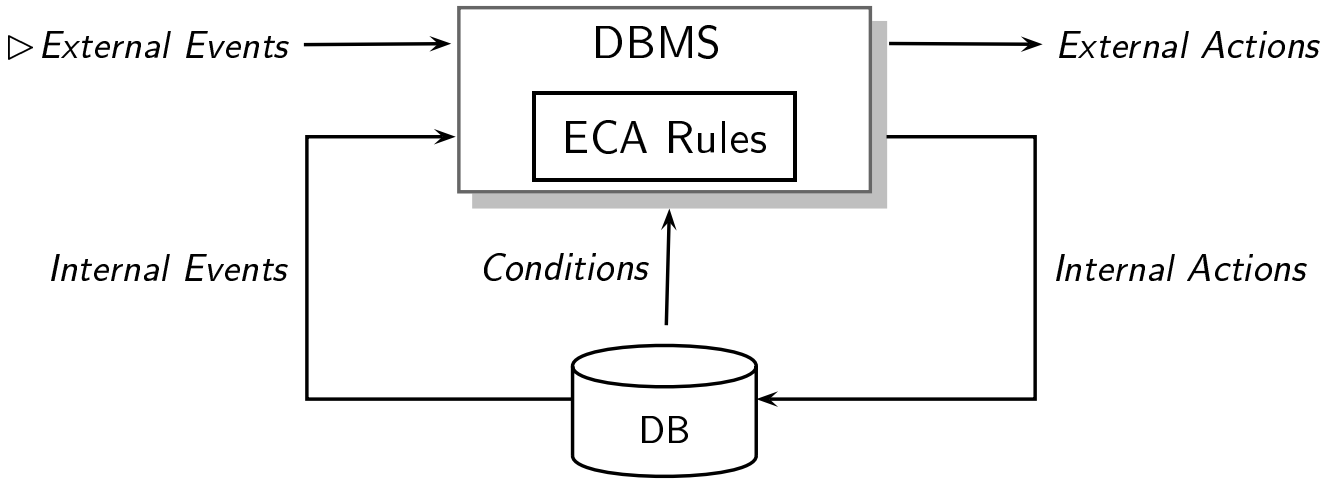
Spektrum von Datenbank-Regelsprachen [Widom93]:



Ziel:



Aktives Datenbanksystem:



ECA-Regeln:

ON ⟨*Event*⟩ **IF** ⟨*Condition*⟩ **THEN** ⟨*Action*⟩

- ⊕ automatische Änderungen (updates)
- ⊕ vielseitig einsetzbar (integrity maintenance, monitoring, ...)
- ⊖ komplexe, **prozedurale** Semantik (Kopplungsmodi, Konfliktauflösung, ...)
- ⊖ formale Eigenschaften kaum untersucht; Terminierung, Konfluenz

Produktionsregeln:

IF ⟨*Condition*⟩ **THEN** ⟨*Action*⟩

Z.B. P-*Datalog* (nicht-inflationäres Datalog):

⟨*Action*⟩	←	⟨*Condition*⟩
del:emp(E,S,D)	←	▷**del**:emp(E,S,D)
del:dept_mgr(D,E), **del**:emp(E1,Sal1,D)	←	**del**:emp(E,_,_), dept_mgr(D,E), emp(E1,Sal1,D)

- ⊕ automatische Änderungen ausdrückbar
- ⊕ klare Fixpunkt-Semantik
- ⊕ formale Eigenschaften (Ausdrucksstärke, Komplexität, Terminierung) genau untersucht [AV]
- ⊕⊖ einfacher als aktive Regeln
- ⊖ Negationsbehandlung ($\neg$ tc) $\Rightarrow$ keine deklarative Semantik

Deduktive Regeln:

IF ⟨*Premise*⟩ **THEN** ⟨*Conclusion*⟩

Z.B. *WF-Datalog* (wohlfundiertes Datalog):

$$
\begin{array}{lcl}
\text{tc(X,Y)} & \leftarrow & \text{e(X,Y).} \\
\text{tc(X,Y)} & \leftarrow & \text{tc(X,Z),tc(Z,Y).} \\
\text{non-tc(X,Y)} & \leftarrow & \neg\, \text{tc(X,Y).} \\
\hline
\text{win(X)} & \leftarrow & \text{move(X,Y)}, \neg\, \text{win(Y).}
\end{array}
$$

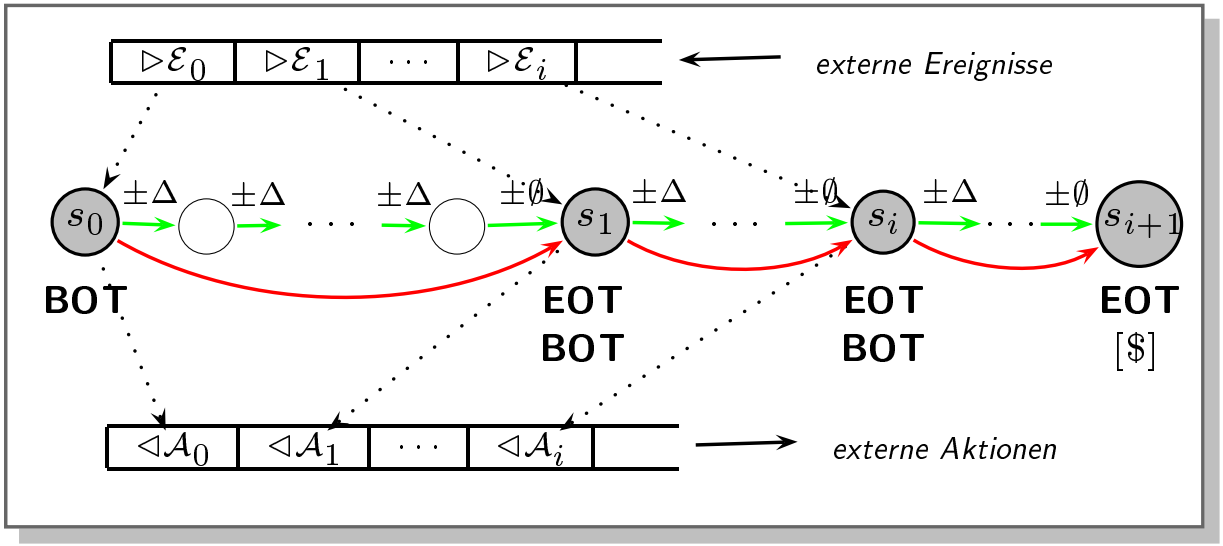
- ⊕ intuitive, deklarative Semantik (Prinzipien: MIN, SUPP, TAUT,...), insbesondere für Negation
- ⊕ formale Eigenschaften bekannt
- ⊖ Änderungsoperationen nicht direkt unterstützt

Grundidee: Erweiterung von Datalog um Zustände:

$$[S{+}k]\, H \;\leftarrow\; [S{+}l]\, B, [S{+}m] \neg C.$$

- $k = l = m \Rightarrow$ *lokale* Regeln $\approx$ Anfragen $\Rightarrow$ kein Zustandsübergang
- $k \geq l, m \Rightarrow$ *progressive* Regeln $\Rightarrow$ definieren Transitionen

Ausführungsmodell:



- Transition: einzelner Zustandsübergang $[s_n] \rightarrow [s_n{+}1]$
- Transaktion: Gesamtübergang $[s_n] \rightsquigarrow [\$]$

Frame-Regeln:

$[S{+}1]\ R(\mathsf{X}) \leftarrow [S]\ R(\mathsf{X}),\ \neg\ \mathbf{del}{:}R(\mathsf{X}).$
$[S{+}1]\ R(\mathsf{X}) \leftarrow [S]\ \mathbf{ins}{:}R(\mathsf{X}).$

ICs (Integritätsbedingungen):

$[S]\ \mathbf{abort} \leftarrow [S]\ \mathbf{ins}{:}R(\mathsf{X}),\ \mathbf{del}{:}R(\mathsf{X})).$

Anfragen:

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

Änderungen:

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

Dynamische ICs und komplexe Ereignisse: $e(X,Y) := (f(X)\,;\, g(Y))$

in PFOTL: $\varphi_e(X,Y) := \blacklozenge(f(X)) \wedge g(Y)$

Statelog Programm P_e:

$[S{+}1]$ **detd_**$\blacklozenge$f(X) $\leftarrow$ $[S]$ f(X).
$[S{+}1]$ **detd_**$\blacklozenge$f(X) $\leftarrow$ $[S]$ **detd_**$\blacklozenge$f(X).
$[S]$ **detd_**e(X,Y) $\leftarrow$ $[S]$ g(Y), **detd_**$\blacklozenge$f(X).

Def. Statelog-Programm P ist zustandsstratifiziert, wenn keine negativen Abhängigkeitszyklen *innerhalb eines Zustands* auftreten.

Satz. *Wenn P progressiv und zustandsstratifiziert $\Rightarrow$ P ein eindeutiges perfektes Modell:*

$$\mathcal{M}_{P\cup\mathcal{D}} = (\mathcal{M}_{P\cup\mathcal{D}}[0]\,, \mathcal{M}_{P\cup\mathcal{D}}[1]\,, \ldots).$$

- P terminiert für $\mathcal{D}$, wenn $\mathcal{M}_{P\cup\mathcal{D}}[n] = \mathcal{M}_{P\cup\mathcal{D}}[n{+}1]\,;\, n \geq n_0$.

Allgemein:
- $\mathcal{M}_{P\cup\mathcal{D}}$ *ist periodisch (ultimately periodic):* $\mathcal{M}_{P\cup\mathcal{D}}[n] = \mathcal{M}_{P\cup\mathcal{D}}[n{+}T]\,;\; n \geq n_0$

$\Rightarrow$ endlich repräsentierbar.

Def. Sei P ein Statelog-Programm:

- P terminiert für $\mathcal{D}$ wenn Periode $T = 1$, ($\text{Term}_{\mathcal{D}}$)
- P terminiert manchmal/immer, wenn $\exists\mathcal{D}/\forall\mathcal{D}$: P terminiert für $\mathcal{D}$. ($\text{Term}_{\exists/\forall}$)

Satz.

- ($\text{Term}_{\mathcal{D}}$) ist **PSPACE**-vollständig ($n = |\mathcal{D}|$).
- ($\text{Term}_{\exists/\forall}$) ist unentscheidbar.
- ($\text{Term}_{\mathcal{D}}$) ist in Statelog ausdrückbar ($P \rightsquigarrow P^{\downarrow}$).

Satz.
Bewachte (G-Statelog) und Δ-monotone Programme (Δ-Statelog) terminieren in **PTIME**.
(Idee: verhindere "Oszillation" von Änderungen: $ins \rightsquigarrow del \rightsquigarrow ins \rightsquigarrow \cdots$)

- ***NF-Statelog***: nur Regeln der Form

$$\begin{array}{rrcl} \textit{1-}(P)\textit{rogressiv}: & [S{+}1]\,H & \leftarrow & [S]\,B, [S]\,\neg\, C. \\ (L)\textit{okal}: & [S]\,H & \leftarrow & [S]\,B, [S]\,\neg\, C. \end{array}$$

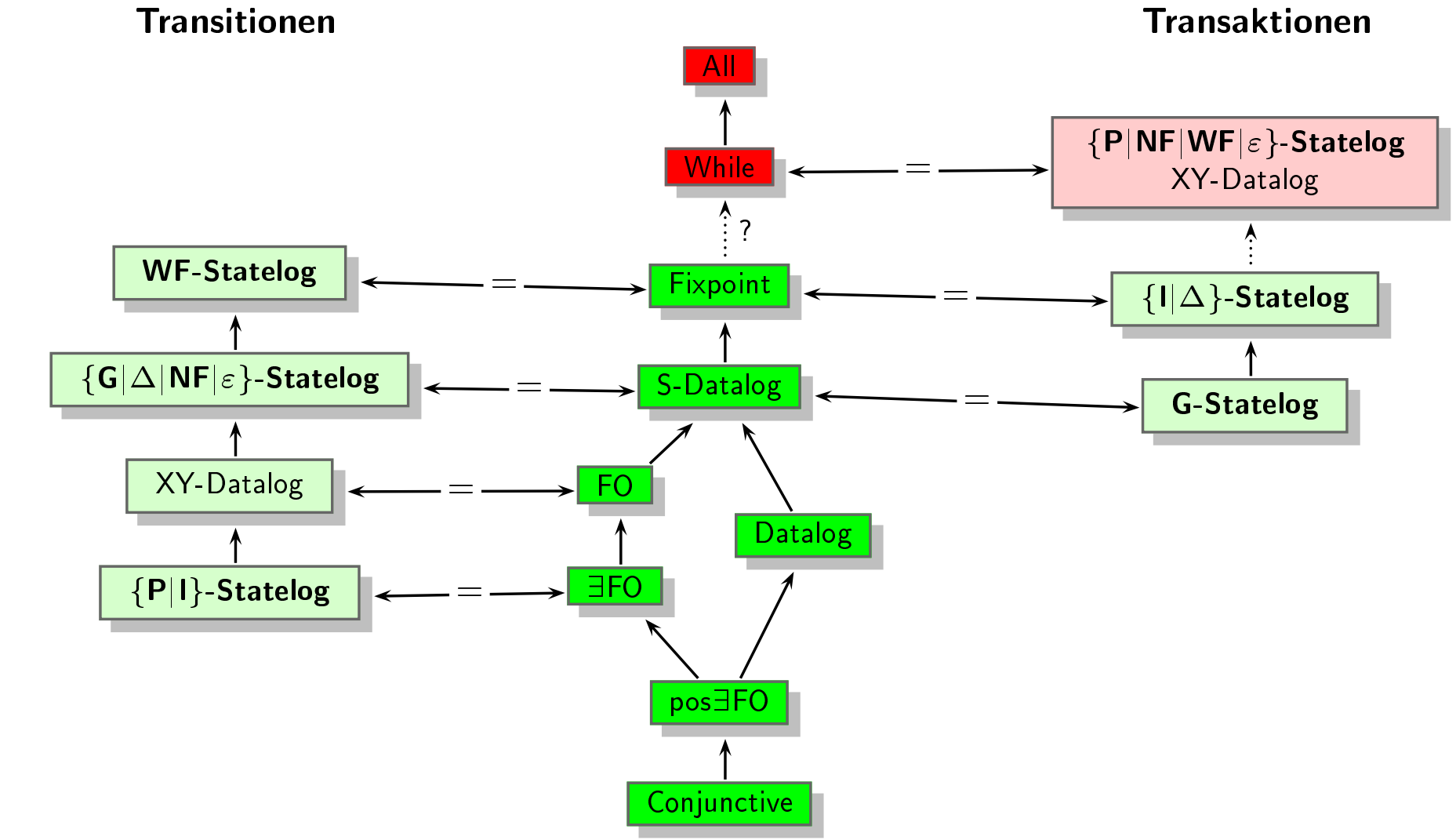
- ***P-Statelog***: nur Regeln vom Typ (P).

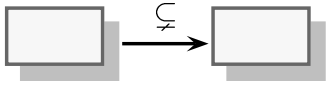
Satz.

Statelog $\equiv_\infty$ NF-Statelog	% *evolutionsäquivalent*
Statelog $\equiv_\$$ P-Statelog	% *transaktionsäquivalent*

Produktionsregeln & deduktive Regeln in Statelog:

$$H \leftarrow B, \neg C \quad \Leftrightarrow \quad \begin{cases} \text{P-Datalog}: & [S{+}1]\,H \leftarrow [S]\,B, [S]\,\neg\, C \\ \text{I-Datalog}: & [S{+}1]\,H \leftarrow [S]\,B, [S]\,\neg\, C \\ & [S{+}1]\,H \leftarrow [S]\,H \\ \hline \text{S-Datalog}: & [S]\,H \leftarrow [S]\,B, [S]\,\neg\, C \\ \text{WF-Datalog}: & [S{+}1]\,H \leftarrow [S{+}1]\,B, [S]\,\neg\, C \end{cases}$$





Daten-Komplexität: ⊇DB-**PSPACE**, ⊆DB-**PTIME**

Daten-Komplexität:

- TRANSITION: geg.: $p(\bar{x})$, $\mathcal{M}_{P\cup\mathcal{D}}[s]$ ges.: $\mathcal{M}_{P\cup\mathcal{D}}[s{+}1] \stackrel{?}{\models} p(\bar{x})$
- TRANSAKTION: geg.: $p(\bar{x})$, $\mathcal{M}_{P\cup\mathcal{D}}[0]$ ges.: $\mathcal{M}_{P\cup\mathcal{D}}[\$] \stackrel{?}{\models} p(\bar{x})$

Satz.

- TRANSITION ist **PTIME**-vollständig.
- TRANSAKTION ist **PSPACE**-vollständig.
  $(n = |\mathcal{D}|)$

- **Datalog**$_{1S}$ [Chomicki]:
  - ***temporale*** *Anfragesprache*, periodisch, Auswertung: **PSPACE**-vollständig,
  - Ausdrucksstärken nicht direkt vergleichbar:
    - Statelog: *relationale* DB, Datalog$_{1S}$: *temporale* DB
    - erlaubt: Rückwärts-Regeln
    - keine Zustandsstratifizierung, Änderungen/Transitionen/Transaktionen

- **XY-Datalog** [Zaniolo]:
  - *Integration von aktiven & deduktiven Regeln*
  - komplexe Ereignisse
  - keine Resultate bzgl. Ausdrucksstärke/Komplexität/Terminierung

- **I-/P-Datalog** [Abiteboul/Vianu]:
  - *Anfrage- & Änderungssprachen*, Produktionsregel-Semantik, umfassende Theorie
  - keine Behandlung von aktiven Regeln
  - Teilklassen von Statelog

Ergebnisse

Statelog integriert

- *aktive* Regeln (automatische Änderungsoperationen, Kopplungsmodi, komplexe Ereignisse; nicht jedoch: "decoupled", "chronicle"-Kontext, ...), und
- *deduktive* Regeln (formale, deklarative Semantik):

$$\textit{Aktive Regeln} = \textit{Deduktive Regeln} + \textit{Zustände}.$$

- *deklarative Semantik*: zustandsstratifiziertes Modell $\mathcal{M}_{P \cup \mathcal{D}}$
- Statelog-Normalformen
  $\Rightarrow$ Charakterisierung der Ausdrucksstärke, Komplexität (bzgl. *Transitionen* und *Transaktionen*)
- Rahmen für aktive Regeln:
  Δ-Relationen, Konfliktbehandlung, Δ-monotones Statelog, dynamische ICs, komplexe Ereignisse
- prototypisch implementiert