Referential Actions as Logical Rules

Bertram Ludäscher Wolfgang May Georg Lausen

Institut für Informatik, Universität Freiburg, Germany

Overview

- Introduction: Referential Integrity Constraints (*ric*'s) and Referential Actions (*rac*'s)
- Ambiguities (Examples)
- Abstract Semantics (Maximal Admissible Sets)
- Translation to Logic Programs and Declarative Semantics
- Conclusion

Syntax:

$$R_C.\vec{F} \rightarrow R_P.\vec{K}$$

Foreign key $\vec{F}$ of the **child relation** R_C references a **(candidate) key** $\vec{K}$ of the **parent relation** R_P.

Semantics:

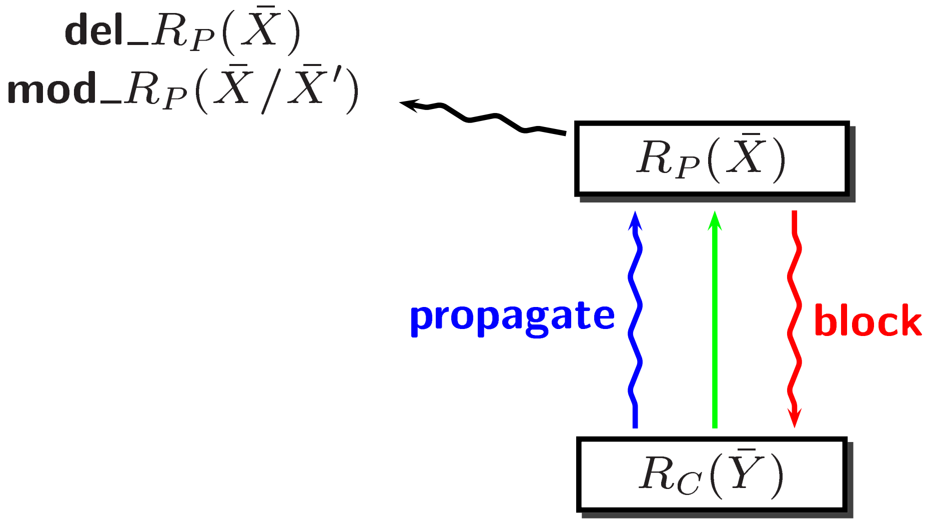
$$\forall \bar{X}\,(R_C(\bar{X}) \rightarrow \exists \bar{Y}\,(R_P(\bar{Y}) \wedge \bar{X}[\vec{F}] = \bar{Y}[\vec{K}]))$$

Example:

Emp.(Dept,Proj) $\rightarrow$ Projects.(DNo,PNo)

Emp			
ENo	**Dept**	Sal	**Proj**
...	...	...	...
...	...	...	...

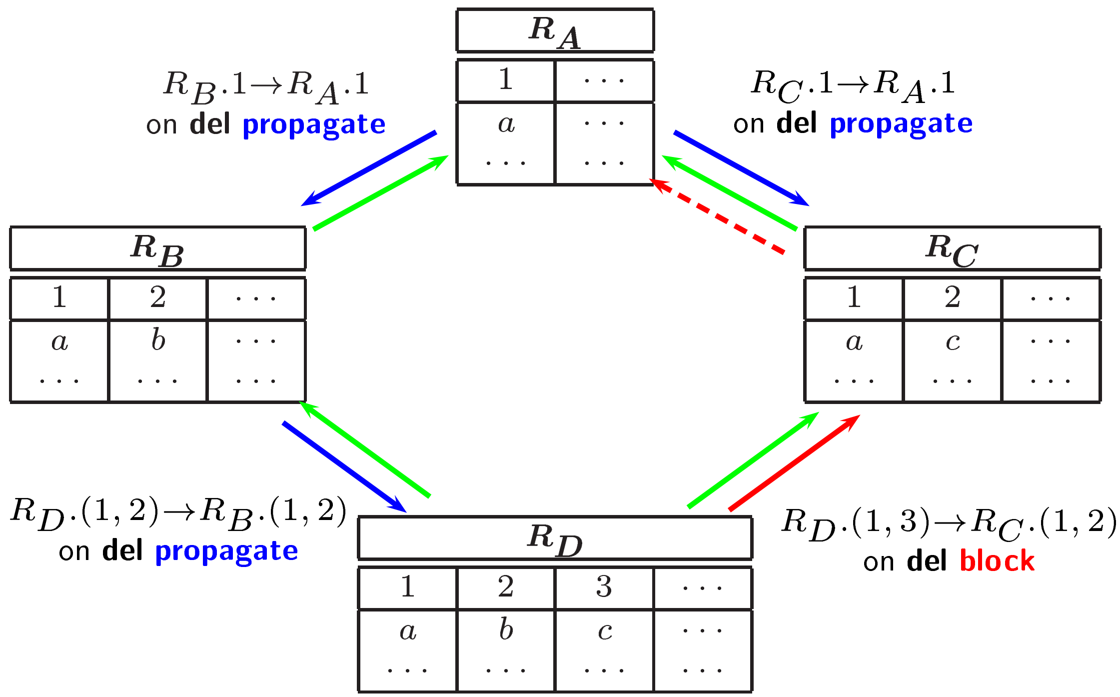
Projects			
DNo	**PNo**	...	...
...	...	...	...
...	...	...	...



Syntax: $R_C.\vec{F} \rightarrow R_P.\vec{K}$ on {**ins** | **del** | **mod**} {parent | child} {**propagate** | **restrict** | **wait**}

rac	≈ SQL	R_P			R_C		
		ins	**del**	**mod**	**ins**	**del**	**mod**
propagate	`CASCADE`	*ok*	•	•	—	*ok*	—
restrict	`RESTRICT`	*ok*	•	•	•	*ok*	•
wait	`NO ACTION`	*ok*	•	•	•	*ok*	•

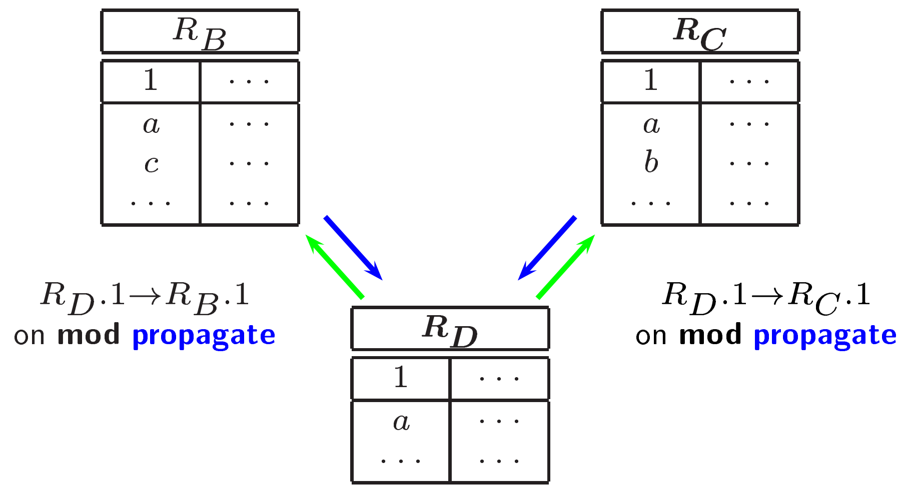
- *rac*'s only specify **local behavior**, but **global semantics** is not clear $\Longrightarrow$ ambiguities, conflicts.



$$
\begin{array}{lll}
U_{\triangleright} & = & \{\triangleright \mathbf{del\_}R_A(a,\ldots)\} \quad (\widehat{=} \text{ given } \textbf{user requests}) \\
 & (1) & R_A \rightsquigarrow R_C \rightsquigarrow R_D \\
 & \Rightarrow & \text{reject } \mathbf{del\_}R_A(a,\ldots) \\
 & (2) & R_A \rightsquigarrow R_B \rightsquigarrow R_D \\
 & \Rightarrow & \text{accept } \mathbf{del\_}R_A(a,\ldots) \\
 & & \\
 & \Rightarrow & \text{ambiguity!}
\end{array}
$$

Logic programming analogue:

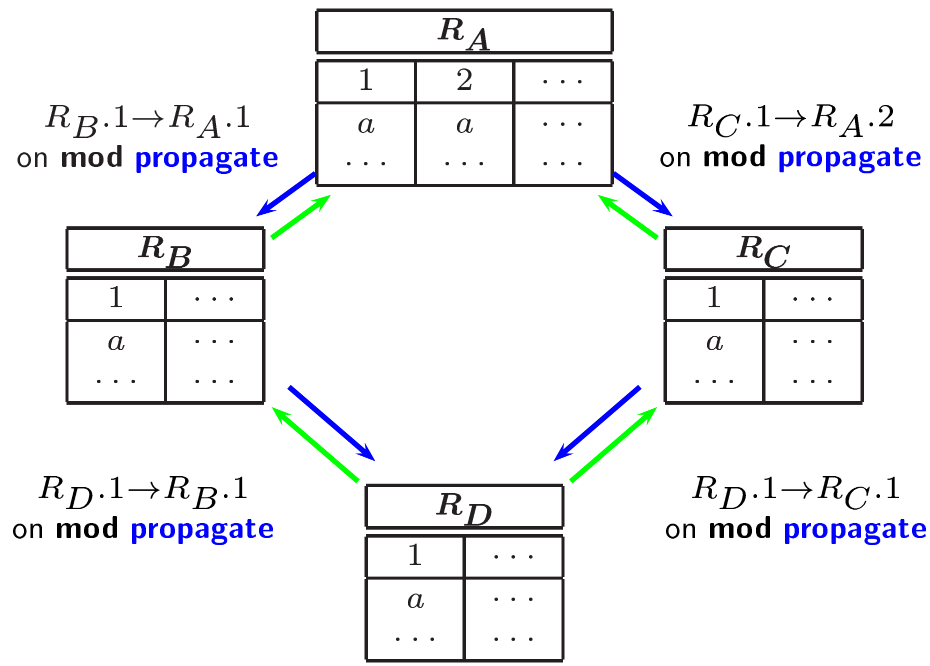
$$
\boxed{\begin{array}{lll} exec & \leftarrow & \neg\, block. \\ block & \leftarrow & \neg\, exec. \end{array}}
$$



Logic programming analogue:

$$
\begin{aligned}
U_{\triangleright} \quad &= \quad \{\triangleright\mathbf{mod\_}R_B(a/b,\cdots), \triangleright\mathbf{mod\_}R_C(a/c,\cdots)\} \\
&\rightsquigarrow \quad \mathbf{mod\_}R_D(a/b,\ldots), \mathbf{mod\_}R_D(a/c,\ldots) \\
&\Rightarrow \quad \text{ambiguity!}
\end{aligned}
$$

$$
\boxed{
\begin{array}{lcl}
exec_1 & \leftarrow & \neg\, block_1. \\
exec_2 & \leftarrow & \neg\, block_2. \\
block_1 & \leftarrow & exec_2. \\
block_2 & \leftarrow & exec_1.
\end{array}
}
$$



$$
\begin{aligned}
U_{\triangleright} \quad &= \quad \{\triangleright\textbf{mod\_}R_A(a/b, a/c, \ldots)\} \\
&\rightsquigarrow \quad \textbf{mod\_}R_B(a/b, \ldots), \textbf{mod\_}R_C(a/c, \ldots) \\
&\rightsquigarrow \quad \textbf{mod\_}R_D(a/b, \ldots), \textbf{mod\_}R_D(a/c, \ldots) \\
&\Rightarrow \quad \text{ambiguity!}
\end{aligned}
$$

Logic programming analogue:

$$
\boxed{\begin{aligned} exec &\leftarrow \neg block. \\ block &\leftarrow exec. \end{aligned}}
$$

Given:

- a set of *rac*'s RA (used to maintain some *ric*'s RI)
- a set of user requests $U_\triangleright$, e.g. $\{\triangleright\textbf{del\_}R_1(a,b), \triangleright\textbf{ins\_}R_2(b,a,c), \ldots\}$
- the current database instance D

Questions:

- Which $U \subseteq U_\triangleright$ can be executed safely, and
- what updates Δ are induced by U and RA?

More formally: Find all **maximal** $U \subseteq U_\triangleright$ such that

- the **induced updates** $\Delta(U)$ preserve RI in $D' := D \pm \Delta(U)$, and
- $\Delta(U)$ reflects the intended meaning of RA.

An admissible set of updates Δ must be "well-behaved" wrt. $U_\triangleright$, RA, and D.

Definition (Admissible Delta): A set of updates Δ is

- **founded** if every $upd \in \Delta$ is "justified" by some $\triangleright upd' \in U_\triangleright$ and propagations using RA
- **complete** if all induced propagations are in Δ
- **feasible** if on ... { restrict | wait } actions are obeyed
- **coherent** if Δ contains no contradicting updates (like e.g. **ins_**$R(\bar{x})$ and **del_**$R(\bar{x})$)
- **key-preserving** if in $D' := D \pm \Delta$ all key dependencies are satisfied
- **admissible** if Δ is founded, complete, feasible, coherent, and key-preserving.

Definition (Intended Semantics): Fix $U_\triangleright$, RA, and D.

- Let $U \subseteq U_\triangleright$. The set $\Delta(U)$ of **induced updates** is the least complete set $\Delta \supseteq U$.
- $U \subseteq U_\triangleright$ is **admissible** if $\Delta(U)$ is admissible.
- The **intended semantics** are the maximal admissible $U \subseteq U_\triangleright$.

Idea: Formalize *rac*'s RA as a logic program P_{RA} (preserves *locality principle*).
The declarative semantics of P_{RA} yields a "reasonable" global semantics.

$\boxed{\text{User Requests and Final Updates}}$ $:\Longleftrightarrow$

$\text{pot\_del\_}R(\bar{X}) \;\leftarrow\; \triangleright\text{del\_}R(\bar{X}).$
$\textbf{del\_}R(\bar{X}) \;\leftarrow\; \triangleright\text{del\_}R(\bar{X}), \neg\,\text{blk\_del\_}R(\bar{X}).$

$\boxed{R_C.\vec{F} \rightarrow R_P.\vec{K} \text{ on } \textbf{del propagate}}$ $:\Longleftrightarrow$

$\textbf{del\_}R_C(\bar{X}) \;\leftarrow\; \textbf{del\_}R_P(\bar{Y}),\; R_C(\bar{X}),\; \bar{X}[\vec{F}] = \bar{Y}[\vec{K}].$
$\text{pot\_del\_}R_C(\bar{X}) \;\leftarrow\; \text{pot\_del\_}R_P(\bar{Y}),\; R_C(\bar{X}),\; \bar{X}[\vec{F}] = \bar{Y}[\vec{K}].$
$\text{blk\_del\_}R_P(\bar{Y}) \;\leftarrow\; \text{pot\_del\_}R_P(\bar{Y}),\; \text{blk\_del\_}R_C(\bar{X}),\; \bar{X}[\vec{F}] = \bar{Y}[\vec{K}]\;.$

$\boxed{R_C.\vec{F} \rightarrow R_P.\vec{K} \text{ on } \textbf{del block}}$ $:\Longleftrightarrow$

$\text{blk\_del\_}R_P(\bar{Y}) \;\leftarrow\; \text{pot\_del\_}R_P(\bar{Y}),\; \text{is\_ref'd\_}R_P.\vec{K}\text{\_by\_}R_C.\vec{F}(\bar{Y}[\vec{K}])\;.$ % *restrict*
$\text{blk\_del\_}R_P(\bar{Y}) \;\leftarrow\; \text{pot\_del\_}R_P(\bar{Y}),\; \text{rem\_ref'd\_}R_P.\vec{K}\text{\_by\_}R_C.\vec{F}(\bar{Y}[\vec{K}])\;.$ % *wait*

Diamond:
$$\begin{aligned} exec &\leftarrow \neg\, block. \\ block &\leftarrow \neg\, exec. \end{aligned}$$

Mutex:
$$\begin{aligned} exec_1 &\leftarrow \neg\, block_1. \\ exec_2 &\leftarrow \neg\, block_2. \\ block_1 &\leftarrow exec_2. \\ block_2 &\leftarrow exec_1. \end{aligned}$$

Self-Attack:
$$\begin{aligned} exec &\leftarrow \neg\, block. \\ block &\leftarrow exec. \end{aligned}$$

- Conflicts of type **self-attack** prevent existence of stable models $\Rightarrow$ **partial stable models** $\mathcal{PS}$.
- The **well-founded model** $\mathcal{W}$ assigns *undefined* to all controversial requests (most sceptic $\mathcal{PS}$).
- **Maximal stable** models work for **mutex** and **self-attack** but not for **diamond** $\Rightarrow$ preference: **exec** $\succ$ **block** $\Rightarrow$ M-stable models $\mathcal{AS}$.

Theorem (Correctness & Completeness).

- For every P-stable model $\mathcal{PS}$ of $P_{RA} \cup D \cup U_{\triangleright}$:
  - $\Delta^{\boldsymbol{true}}_{\mathcal{PS}}$ is admissible,
  - $\Delta^{\boldsymbol{true}}_{\mathcal{PS}} = \Delta(U^{\boldsymbol{true}}_{\mathcal{PS}})$,
  - $U^{\boldsymbol{true}}_{\mathcal{PS}}$ is admissible.

  (Special case: $\mathcal{PS} = \mathcal{W}$)
- For every maximal admissible $U \subseteq U_{\triangleright}$, there is an M-stable model $\mathcal{MS}$ s.t. $U = U^{\boldsymbol{true}}_{\mathcal{MS}}$ and $\Delta(U) = \Delta^{\boldsymbol{true}}_{\mathcal{MS}}$.

Application-specific preference: $\mathcal{PS}_1 <_a \mathcal{PS}_2 \;:\Leftrightarrow\; U^{\boldsymbol{true}}_{\mathcal{PS}_1} \subset U^{\boldsymbol{true}}_{\mathcal{PS}_2}$.

Theorem (Maximality).

- The maximal admissible sets $U \subseteq U_{\triangleright}$ are given by the M-stable models of $P_{RA} \cup D \cup U_{\triangleright}$ which are maximal wrt. $<_a$.

Summary.

- *rac*'s can be used to maintain *ric*'s
- global effect of *rac*'s is unclear

$\Rightarrow$ definition of an abstract, intended semantics ($\widehat{=}$ several "equally justified" outcomes)

$\Rightarrow$ constructive semantics results from specifying a set RA of *rac*'s as a logic program P_{RA} with declarative semantics

$\Rightarrow$ general solution to the meaning of *rac*'s

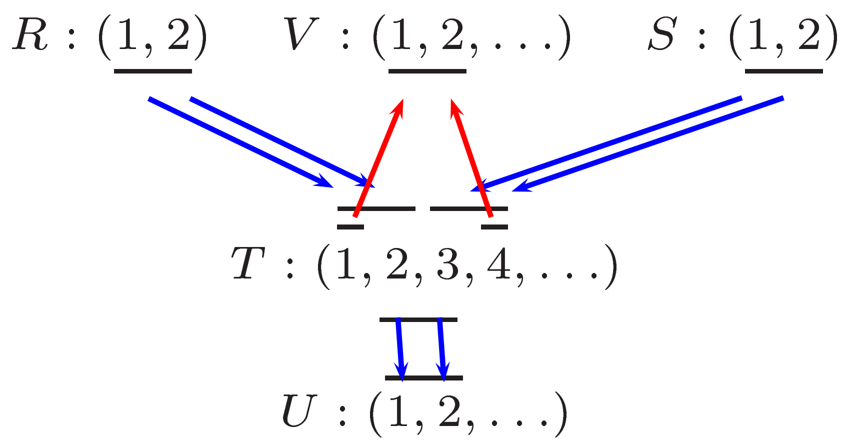
Possible Applications.

- general framework: as an analysis & explanation tool
- restricted framework (using simplifying assumptions on interplay of *rac*'s): executable logical specification $\Rightarrow$ basis for a procedural implementation

Definition (P-, M-Stable Models):

Let $I = \langle I^{\boldsymbol{true}}, I^{\boldsymbol{false}} \rangle$ be a 3-valued interpretation. The reduction P/I of a ground instantiated logic program P is obtained by replacing every negative literal in P by its truth-value wrt. I. Thus, P/I is positive and has a unique minimal (wrt. the truth-order $\boldsymbol{false} <_t \boldsymbol{undef} <_t \boldsymbol{true}$) 3-valued model $\mathcal{M}_{P/I}$.

I is a *P-stable* model, if $\mathcal{M}_{P/I} = I$. A P-stable model I is *M-stable* (*maximal stable*) if there is no P-stable model $J \neq I$ such that $J^{\boldsymbol{true}} \supseteq I^{\boldsymbol{true}}$ and $J^{\boldsymbol{false}} \supseteq I^{\boldsymbol{false}}$.



Depending on the database state D, changes on R and S
(i) must not be merged, or
(ii) have to be merged on T.

Given **mod_**$R(a/a', b/b')$ and **mod_**$S(c/c', d/d')$. Then

(i) D contains $R(a, b), S(c, d), T(a, b, c, d), U(b, c), V(a, d), V(a', d), V(a, d')$

(ii) similar to (i) but $V(a, d), V(a', d')$.