Transaction Logic with Defaults and Argumentation Theories

Paul Fodor       Michael Kifer

State University of New York at Stony Brook
Stony Brook, NY 11794, U.S.A

ICLP 2011,
Stony Brook, NY 11794, U.S.A.
Overview

- **Transaction Logic**
  - Integrates declarative and procedural knowledge
  - Defeasible reasoning
    - **Logic programs with defaults and argumentation theories** (LPDA)
      - Argumentation theories: simple “meta-” rules in the LP itself to specify when rules ought to be defeated

- **Transaction Logic with Defaults and Argumentation Theories** ($\mathcal{T} \mathcal{R}^{DA}$)
  - Unifies the two streams of research
  - Applications: specification of defaults in action theories and heuristics for directed search in AI problems (planning)
  - Well developed model theory, reducible to Sequential Transaction Logic
  - Experiments: show heuristics expressed as defeasible actions (reduce the search space, the execution time and space requirements)
Outline

- Introduction
  - Transaction Logic
  - LPDA Defeasible reasoning
- \(\mathcal{TR}^{DA}\) framework
  - Syntax
  - Semantics
- \(\mathcal{TR}^{DA}\) Evaluation
Transaction Logic

- Logic for programming state-changing actions and reasoning about their effects
- The set of predicate symbols of the program are partitioned into:
  - **fluents**: facts stored in database states or derived predicates that do not change the state of the database
  - **actions**: procedures that change states
    - User defined actions
    - **Elementary actions**: built-in actions for basic manipulation of states: \(\text{delete}(f)\) and \(\text{insert}(f)\) for every fluent instance \(f\)
- Connectives: classical logical connectives \((\land, \lor, \neg)\) and additional connectives \((\otimes, \oplus, \lozenge, \circ, |)\)
  - \(\phi : \neg \gamma\) define \(\gamma\) to be an execution of \(\phi\); \hspace{1cm} (Horn
  - \(\phi \otimes \psi\) means: execute \(\phi\), then execute \(\psi\); \hspace{1cm} Serial TR
  - \(\lozenge \psi\) (the modal operator of hypothetical execution) means: \hspace{1cm} hypothetically
    - testing whether \(\phi\) can be executed at the current state, but no actually state changes take place
A formula $\phi$ in Transaction Logic is a transaction and has a truth value over execution paths (sequences of states).

States: $s_1$ (initial state), $\ldots$, $s_n$ (final state)

Semantics: $\phi$ executes along $\pi$ iff $\phi$ is true on $\pi$

Proof theory: executes $\phi$ along $\pi$ as it proves $\phi$
Defeasible Reasoning

- Defeasible Reasoning
  - common sense reasoning where rules can be true by default BUT may have contradicting results
  - some rules may be defeated
  - useful when many rules exist in the system and they cannot all check each other’s preconditions

- Existing approaches:
  - Courteous Logic Programs (Grosof in IBM Common Rules 1999)
  - Defeasible logic (Nute et al. 1993+)
  - Prioritized defaults (Gelfond & Son 1997)
  - Preferred answer sets (Brewka & Eiter 2004)
  - Compiling preferences (Delgrande et al. 2003)
LPDA Defeasible Reasoning

- **LPDA**
  - Unifies almost all previous defeasible LP approaches in one KR
  - Reuses most previous LP algorithms and optimizations
  - Generalizes defeasible LP to HiLog-style higher-order logics and F-logic style object-oriented features
  - Implemented as extension to Flora-2 and part of Vulcan’s SILK project (Semantic Inferencing on Large Knowledge)
    - [http://silk.semwebcentral.org](http://silk.semwebcentral.org)
  - Multiple argumentation theories, exclusion constraints, multi-way conflicts, omni rules
Transaction Logic with Defaults and Argumentation Theories:

- $\mathcal{TR}^{DA}$ rules are tagged with terms (defeasible reasoning)
- The predicate $!$ opposes is used to specify that some rules are incompatible with others
- The predicate $!$ overrides specifies that some actions have higher priority than other actions.
- Cycles are permitted in the definition of actions
Example 1

Stock market actions: weigh recommendations and make decisions about buying and selling stocks

@buy_actbuy(Stock, Amount) : recommendation(buy, Stock) ⊗ owns(Stock, Qty) ⊗ delete(owns(Stock, Qty)) ⊗ insert(owns(Stock, Qty + Amount)).
@sell_actsell(Stock, Amount) : recommendation(sell, Stock) ⊗ owns(Stock, Qty) ⊗ delete(owns(Stock, Qty)) ⊗ insert(owns(Stock, Qty - Amount)).
!
!opposes(sell(Stock), buy(Stock)).
!overrides(sell_action, buy_action).
recommendation(buy, C) : − services(X).
recommendation(sell, C) : − media(X).
services(acme).
media(acme).
owns(acme, 100).
trade(Stock, Amount) : − buy(Stock, Amount).
trade(Stock, Amount) : − sell(Stock, Amount).

Selling and buying the same stock as part of the same decision is contradictory - rules in conflict
Safety: The rule sell overrides the rule buy
An existential goal (∃) trade(acme, 100)
  Without the !opposes and !overrides information = two non-deterministic possible executions
Selling 100 stocks is preferred
Planning for building pyramids of blocks:

\[
\text{\@mv\_rule}(\text{Block}, \text{To}) \text{move}(\text{Block}, \text{From}, \text{To}) : - \\
(\text{on}(\text{Block}, \text{From}) \land \text{larger}(\text{To}, \text{Block})) \otimes \\
\text{pickup}(\text{Block}, \text{From}) \otimes \text{putdown}(\text{Block}, \text{To}).
\]

\[
\text{pickup}(X, Y) : -(\text{clear}(X) \land \text{on}(X, Y)) \otimes \\
\text{delete}(\text{on}(X, Y)) \otimes \text{insert}(\text{clear}(Y)).
\]

\[
\text{putdown}(X, \text{table}) : -(\text{clear}(X) \land \text{not on}(X, Z)) \\
\otimes \text{insert}(\text{on}(X, \text{table})).
\]

\[
\text{putdown}(X, Y) : -(\text{clear}(X) \land \text{clear}(Y) \land \text{not on}(X, Z)) \\
\otimes \text{delete}(\text{clear}(Y)) \otimes \text{insert}(\text{on}(X, Y)).
\]

\[
\text{\@opposes}(\text{move}(B1, F1, T1), \text{move}(B2, F2, T2)) : -B1 \neq B2.
\]
Heuristics: cut down on the number of plans

- Move-actions that move bigger blocks are preferred to move-action that move smaller blocks (unless the blocks are moved down on the table surface)

\[
!\text{overrides}(mv\_rule(B2, To), mv\_rule(B1, To)) : \neg \text{larger}(B2, B1) \land To \neq \text{table}.
\]

- The heuristic for moving blocks to the table surface: prefer unstacking smaller blocks

\[
!\text{overrides}(mv\_rule(B2, \text{table}), mv\_rule(B1, \text{table})) : \neg \text{larger}(B1, B2).
\]
**\( \mathcal{T}R^D_A \) Example 2**

- **Starting configuration of blocks:**

  \[
  \text{on(} blk_1, blk_4 \text{). on(} blk_2, blk_5 \text{).}
  \text{on(} blk_4, \text{table). on(} blk_5, \text{table). on(} blk_3, \text{table).}
  \text{clear(} blk_1 \text{). clear(} blk_2 \text{). clear(} blk_3 \text{).}
  \text{larger(} blk_2, blk_1 \text{). larger(} blk_3, blk_1 \text{). larger(} blk_3, blk_2 \text{). larger(} blk_4, blk_1 \text{). larger(} blk_5, blk_2 \text{). larger(} blk_5, blk_4 \text{).}
  \]

- Both \( blk_1 \) and \( blk_2 \) can be moved on top of \( blk_3 \), moving \( blk_2 \) has higher priority because it is larger.

- Unstacking \( blk_1 \) to the table surface is preferable to unstacking \( blk_2 \) (\( blk_1 \) is a smaller block)
  - creates an opportunity to move \( blk_4 \) on top of \( blk_2 \) and subsequently put \( blk_1 \) on top of \( blk_4 \).
Use the heuristic preference rules in order to improve the performance of a pyramid building

\( \text{stack}(0, \text{Block}). \)
\( \text{stack}(N, X) : - N > 0 \otimes \text{move}(Y, X) \otimes \text{stack}(N - 1, Y) \)
\( \otimes \text{on}(Y, X). \)
\( \text{stack}(N, X) : - N > 0 \otimes \text{on}(Y, X) \otimes \text{unstack}(Y) \)
\( \otimes \text{stack}(N, X). \)
\( \text{unstack}(X) : - \text{on}(Y, X) \otimes \text{unstack}(Y) \otimes \text{unstack}(X). \)
\( \text{unstack}(X) : - \text{isclear}(X) \land \text{on}(X, \text{table}). \)
\( \text{unstack}(X) : - (\text{isclear}(X) \land \text{on}(X, Y) \land Y \neq \text{table}) \)
\( \otimes \text{move}(X, \text{table}). \)
\( \text{unstack}(X) : - \text{on}(Y, X) \otimes \text{unstack}(Y) \otimes \text{unstack}(X). \)
Workflow modeling and execution: a *buy* transaction is designed to make a financial transaction and a delivery

\[
\begin{align*}
\text{buy} & : - \text{pay} \otimes \text{delivery}. \\
\text{buy} & : - \text{delivery} \otimes \text{pay}. \\
@b1\text{delivery} & : - \text{gold}\_\text{member} \otimes \text{express}\_\text{mail}. \\
@b2\text{delivery} & : - \text{ground}\_\text{mail}. \\
@b3\text{pay} & : - \text{pay}\_\text{credit}\_\text{card}. \\
@b4\text{pay} & : - \text{pay}\_\text{cheque}. \\
\text{express}\_\text{mail} & : - \text{insert}\(\text{delivered}\_\text{express}\_\text{mail}\). \\
\text{ground}\_\text{mail} & : - \text{insert}\(\text{delivered}\_\text{ground}\_\text{mail}\). \\
\text{pay}\_\text{credit}\_\text{card} & : - \text{credit}\_\text{card}\_\text{credentials} \otimes \text{insert}\(\text{credit}\_\text{card}\_\text{payment}\). \\
\text{pay}\_\text{cheque} & : - \text{bank}\_\text{account} \otimes \text{insert}\(\text{bank}\_\text{payment}\). \\
\text{credit}\_\text{card}\_\text{credentials}.\text{bank}\_\text{account}.\text{gold}\_\text{member}. \\
\end{align*}
\]
A **fluent literal** is either an atomic fluent or \( \text{neg atm} \) ("strong", classical negation), \( \text{not atm} \) (default negation), \( \text{not neg atm} \)

An **action literal** is an action atomic formula or \( \text{not } \alpha \) (not possible to execute \( \alpha \) starting from the current state)

A **database state** is a set of ground base fluents.

**Elementary state transitions:**
- \( \text{insert}(f) \) causes a transition from \( D \) to the state \( D \cup \{f\} \setminus \{\text{neg } f\} \); and
- \( \text{delete}(f) \) causes a transition from \( D \) to \( D \setminus \{f\} \cup \{\text{neg } f\} \).

Serial goals are defined recursively as follows:
- If \( P \) is a fluent or an action literal then \( P \) is a serial goal.
- If \( P \) is a serial goal, then so are \( \text{not } P \) and \( \Diamond P \).
- If \( P_1 \) and \( P_2 \) are serial goals then so are \( P_1 \otimes P_2 \) and \( P_1 \land P_2 \).
A tagged rule is:
\[ @r \; H \; : \; \neg \; B. \]
the tag \( r \) of a rule is a term
\( H \) is a \textit{not}-free literal and \( B \) is a serial goal

- **Fluent rules:** \( H \) is a derived fluent or \textit{neg} \( H \) and
  \[ B = f_1 \otimes \ldots \otimes f_n, \] where each \( f_i \) is a fluent literal (\( \otimes \) could be replaced with \( \land \))

- **Action rules:** \( H \) must be an atomic action formula, while the body of the rule, \( B \), is a serial goal

\[ \text{handle}(r,H) \] is the \textit{handle} of that rule.

An **argumentation theory**, \( AT \), is a set of untagged rules

- The rules \( AT \) are used to specify how the rules in \( P \) get defeated.
- A unary predicate: \$\textit{defeated}_{AT} \] (may not appear in the transaction base)
- May contain auxiliary predicates used in axioms that define \$\textit{defeated}_{AT} \]
**$\mathcal{TR}^{DA}$ Well-Founded Semantics**

- **Herbrand base $\mathcal{B}$:**
  - $\mathcal{B}_F$, the **Herbrand Base of fluents** is a subset of $\mathcal{B}$ that consists of the fluent-literals
  - $\mathcal{B}_{EU}$, the **Herbrand Base of elementary updates** is a subset of ground insert- and delete-literals that are used for elementary transitions
  - $\mathcal{B}_A$, the **Herbrand Base of actions** is the subset of $\mathcal{B}$ that consists of action-literals

- A **partial Herbrand interpretation** is a mapping $\mathcal{H}$ that assigns $f$, $u$ or $t$ to every formula $L$ in $\mathcal{B}$.

- A **path** is a finite sequence of states, $\pi = \langle D_1 \ldots D_k \rangle$, where $k \geq 1$
\( \mathcal{TR}^{\text{DA}} \) Well-Founded Semantics

A **partial Herbrand Path Structure** is a mapping \( I \) that assigns a partial Herbrand interpretation to every **path**

1. \( I(\langle D \rangle)(d) = t \), if \( d \in D \);
   \( I(\langle D \rangle)(d) = f \), if \( d \notin D \);
   \( I(\langle D \rangle)(d) = u \), otherwise, for every ground base fluent literal \( d \) and every database state \( D \).

2. \( I(\langle D_1, D_2 \rangle)(\text{insert}(p)) = t \) if \( D_2 = D_1 \cup \{ p \} \setminus \{ \text{neg} \, p \} \) and \( P \) is a ground fluent literal;
   \( I(\langle D_1, D_2 \rangle)(\text{insert}(p)) = f \), otherwise.

3. \( I(\langle D_1, D_2 \rangle)(\text{delete}(p)) = t \) if \( D_2 = D_1 \setminus \{ p \} \cup \{ \text{neg} \, p \} \) and \( P \) is a ground fluent literal;
   \( I(\langle D_1, D_2 \rangle)(\text{delete}(p)) = f \), otherwise.
**Transaction Logic**

**Well-Founded Semantics**

- **Split**: A split of $\pi$ is any pair of sub paths $\pi_1 \circ \pi_2$, such that $\pi_1 = \langle D_1 \ldots D_i \rangle$ and $\pi_2 = \langle D_i \ldots D_k \rangle$.

- **Propositional Constants**: Special propositional constants: $u^\pi$ and $t^\pi$, for each path $\pi$ (propositional transaction that are undefined or true over the path $\pi$ and false on all other paths).

- **Truth Valuation**: Truth valuation in path structures:
  - If $\phi$ and $\psi$ are serial goals and $\pi = \pi_1 \circ \pi_2$ then $I(\pi)(\phi \otimes \psi) = \min(I(\pi_1)(p), I(\pi_2)(q))$.
  - For any path $\pi$:
    - $I(\pi)(t^\pi) = t$ and $I(\pi')(t^\pi) = f$, if $\pi' \neq \pi$;
    - $I(\pi)(u^\pi) = u$ and $I(\pi')(u^\pi) = f$, if $\pi' \neq \pi$. 
Well-Founded Semantics

Truth valuation in path structures:

- If $\phi$ is a serial goal then $I(\pi)(\neg \phi) = \sim I(\pi)(\phi)$, where $\sim t = f$, $\sim f = t$, and $\sim u = u$.
- If $\phi$ is a serial goal and $\pi = \langle D \rangle$, where $D$ is a database state, then
  
  $I(\pi)(\diamond \phi) = \max \{ I(\pi')(\phi) \mid \pi' \text{ is a path that starts at } D \}$
  $I(\pi)(\diamond \phi) = f$, otherwise.
- For a untagged serial rule $F : - G$, $I(\pi)(F : - G) = t$ iff $I(\pi)(F) \geq I(\pi)(G)$.
- For a defeasible rule $\@ r F : - G$,
  
  $I(\pi)(\@ r F : - G) = t$ iff
  
  $I(\pi)(F) \geq \min (I(\pi)(G), I(\pi)(\neg \diamond ~ defeated(handle(r, F))))$. 
A path structure, $I$, is a model of a transaction formula $\phi$ if $I, \pi \models \phi$ for every path $\pi$.

A path structure $I$ is a model of a serial $\mathcal{TR}^{DA}$ transaction base $P$ if all the rules in $P$ are satisfied in $I$.

A path structure $M$ is a model of $P$ with respect to the argumentation theory $AT$, $M \models (P, AT)$, if $M \models P$ and $M \models AT$. 

\[ \mathcal{TR}^{DA} \text{ Well-Founded Semantics} \]
**$\mathcal{T}R^{DA}$ Well-Founded Semantics**

- Order on Herbrand partial interpretations:
  - $\sigma_1 \preceq \sigma_2$ if all $\text{not}$-free literals that are true in $\sigma_1$ are also true in $\sigma_2$ and all $\text{not}$-literals that are true in $\sigma_2$ are also true in $\sigma_1$.
  - $\sigma_1 \leq \sigma_2$ if all $\text{not}$-free literals that are true in $\sigma_1$ are also true in $\sigma_2$ and all $\text{not}$-literals that are true in $\sigma_1$ are also true in $\sigma_2$. 
**Order on Path Structures:**

- $M_1 \preceq M_2$ if $M_1(\pi) \preceq M_2(\pi)$ for every path $\pi$
- $M_1 \preceq M_2$ if $M_1(\pi) \preceq M_2(\pi)$ for every path $\pi$

A model $M$ of $P$ is **minimal** with respect to $\preceq$ iff for any other model, $N$, of $P$ $N \preceq M$ implies $N = M$.

The **least** model of $P$ is a minimal model that is unique.

If $P$ is a **not**-free $\mathcal{TR}$ program, then $P$ has a least Herbrand model, denoted $LPM(P)$. 
\( \mathcal{TR}^{DA} \) Well-Founded Iterated Least Model

- The \( \mathcal{TR}^{DA} \) quotient of \( P \) by \( I \), \( \frac{P}{I} \):
  1. First, each occurrence of every \textit{not} -literal of the form \( \text{not} \; L \) in \( P \) is replaced by \( t^\pi \) for every path \( \pi \) such that \( I(\pi)(\text{not} \; L) = t \) and with \( u^\pi \) for every path \( \pi \) such that \( I(\pi)(\text{not} \; L) = u \).
  2. Replace each labeled rule of the form \( @r \; L : - \; \text{Body} \) with:

\[
\begin{align*}
L : & - t^{\langle D_t \rangle} \otimes \text{Body} \\
L : & - u^{\langle D_u \rangle} \otimes \text{Body}
\end{align*}
\]

for each database state \( D_t \) such that
\[
\max \{ I(\pi)(\text{defeated}(\text{handle}(r, L))) \mid \pi \text{ starts a } D_t \} = t
\]
and each database state \( D_u \) such that
\[
\max \{ I(\pi)(\text{defeated}(\text{handle}(r, L))) \mid \pi \text{ starts a } D_u \} = u
\]

3. Remove the labels from the remaining rules.
The well-founded model of a $\mathcal{TR}^D\mathcal{A}$ transaction base $P$ with respect to the argumentation theory $AT$, $WFM(P, AT)$, is the limit of the transfinite induction:

1. $I_0$ — the path structure that maps each path $\pi$ to the empty Herbrand interpretation
2. If $I_n$ has already been defined for every $n < m$,
   - $I_m = LPM\left(\frac{P \cup AT}{I_{m-1}}\right)$, if $m$ is a non-limit ordinal
   - For every path $\pi$ and every literal $L$
     \[
     I_m(\pi)(L) = \begin{cases} 
     t & \text{if } I_n(\pi)(L) = t \text{ for some } n < m \\
     f & \text{if } I_n(\pi)(L) = f \text{ for some } n < m \\
     u & \text{otherwise (if } I_n(\pi)(L) = u \text{ for all } n < m) 
     \end{cases}
     \]
     if $m$ is a limit ordinal.
\( I_0 = \{ \} \)

\[ P \sqcup AT \]

\( I_0 \)

Quotient

Least partial model

\( I_1 = \text{LPM}( P \sqcup AT \mid I_0 ) \)

\[ P \sqcup AT \]

\( I_1 \)

Iteration of least models

\[ I_n = \text{LPM}( P \sqcup AT \mid I_{n-1} ) \]

\[ P \sqcup AT \]

\( I_n \)

Quotient Iteration of least models

Least partial model

\[ I_{n+1} = \text{LPM}( P \sqcup AT \mid I_n ) = I_n \]
The transfinite sequence of Herbrand path structures $\langle I_0, I_1, \ldots \rangle$ is always increasing ($\leq$) and has a (unique) limit reached for some ordinal, $\alpha$, such that $I_\alpha = I_{\alpha+1}$

$WFM(P, AT)$

$WFM(P, AT)$ coincides with the well-founded model of the $\mathcal{T}R$ program $P' \cup AT$, $P'$ is obtained from $P$ by changing every defeasible rule $(\varnothing r L :- Body) \in P$ to the plain rule $L :- \text{not}(\diamond$ defeated(handle(r,L))) $\otimes Body$ and removing all the tags
The $GCLP^{TR}$ Courteous Argumentation Theory

- Extends *generalized courteous logic programs* (GCLP) (Grosof 1999) to $TR$ under the $TR^{DA}$ framework
  - $\triangleleft$ *opposes* and $\triangleright$ *overrides* are user defined relations specified over rule handles (rules are in opposition or prefered)
  - *candidate* rule handler is a rule instance whose body is hypothetically true in the current database state

$$\text{$\textit{candidate}$(R)} : - \quad \text{$\textbf{body}$(R, B)} \otimes \text{$\textit{call}$(B)}.$$

- $\text{refutes}$: a higher-priority rule implies a conclusion that is incompatible with the conclusion implied by the another rule
- $\text{rebuts}$: a pair of rules assert conflicting conclusions without being able to select a conclusion “more important” than the other conclusion
The $GCLP^{TR}$ Courteous Argumentation Theory

$\text{defeated}(R) \quad : \quad \neg \text{refutes}(S, R) \land \neg \text{compromised}(S).

$\text{defeated}(R) \quad : \quad \neg \text{rebuts}(S, R) \land \neg \text{compromised}(S).

$\text{defeated}(R) \quad : \quad \neg \text{disqualified}(R).

$\text{refutes}(R, S) \quad : \quad \neg \text{conflict}(R, S) \land \neg \text{overrides}(R, S).

$\text{rebuts}(R, S) \quad : \quad \text{candidate}(R) \land \text{candidate}(S) \land \
\quad \neg \text{opposes}(R, S) \land \neg \text{compromised}(R) \land \
\quad \neg \text{refutes}(_, R) \land \neg \text{refutes}(_, S).

$\text{compromised}(R) \quad : \quad \text{refuted}(R) \land \text{defeated}(R).

$\text{disqualified}(X) \quad : \quad \text{defeats}_{tc}(X, X).

$\text{defeats}_{tc}(X, Y) \quad : \quad \text{defeats}(X, Y).

$\text{defeats}_{tc}(X, Y) \quad : \quad \text{defeats}_{tc}(X, Z) \land \text{defeats}(Z, Y).

\neg \text{opposes}(\text{handle}(_, H), \text{handle}(_, \neg H)).
## Evaluation - The blocks world planning example

<table>
<thead>
<tr>
<th>World size</th>
<th>No heuristics</th>
<th>With preferential heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Plans</td>
<td>Time (sec.)</td>
</tr>
<tr>
<td>10 blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 blocks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discussion and related work

  - Consider planning with complete information on finite domains and deterministic actions
  - Use answer set planning (Subrahmanian, 1995)
  - Preferences on trajectories or choice and temporal order of achievement of fluent goals

- $\mathcal{TR}^{DA}$ is a full procedural logic with priorities between actions
  - $\mathcal{TR}^{DA}$ allows infinite domains for planning
  - $\mathcal{TR}^{DA}$ allows function symbols
  - $\mathcal{TR}^{DA}$ allows non-deterministic actions
  - Well-founded semantics (Van Gelder et al., 1991)

- Preferences in modeling, execution and verification of workflows (Governatori, 2006)
Developed a theory of defeasible reasoning for Transaction Logic, a purely declarative extension of classical logic for defining state-changing transactions

Courteous style of defeasible reasoning in declarative and procedural knowledge

Defined the well-founded semantics (new) for Transaction Logic and its $TR^{DA}$ extension

Future work
  - Application to business process management
Thank you!

Questions?

Disclaimer: The preceding slides represent the views of the authors only.

All brands, logos and products are trademarks or registered trademarks of their respective companies.