RIF RuleML Rosetta Ring: Round-Tripping the Dlex Subset of Datalog RuleML and RIF-Core

The International RuleML Symposium on Rule Interchange and Applications
Las Vegas, Nevada, USA, Nov 5-7, 2009

Harold Boley

Institute for Information Technology
National Research Council of Canada
Fredericton, NB, E3B 9W4, Canada

Abstract

The RIF RuleML overlap area is of broad interest for Web rule interchange. Its kernel, Dlex, is defined syntactically and semantically as the common sublanguage of Datalog RuleML and RIF-Core restricted to positional arguments and non-conjunctive rule conclusions, and allowing equality plus externals in rule premises (only). Semantics-preserving mappings are defined between the Dlex subset of the RIF Presentation Syntax, RIF/XML, RuleML/XML, and the Prolog-like RuleML/POSL. These are the basis for RIF RuleML feature comparison and translation. The slightly augmented mappings can be composed into a ('Rosetta') ring for round-tripping between all pairs of Dlex representations

Introduction: Dlex Kernel of RuleML and RIF

- RuleML moves to version 1.0, and RIF to Recommendation
- Overlap area between RIF and RuleML is of broad interest for Web rule interchange
- Focus on Dlex kernel:
  - Common sublanguage of Datalog RuleML and RIF-Core
  - restricted to positional arguments and non-conjunctive rule conclusions
  - allowing equality plus externals – built-ins as defined in RIF-DTB – in rule premises (only)

Introduction: Mappings Between Dlex Representations

- Advance RIF RuleML development and interoperation:
  Semantics-preserving mappings within Dlex
- Mappings between:
  - RIF Presentation Syntax and RIF/XML
  - RIF/XML and RuleML/XML
  - RuleML/XML and Prolog-like RuleML/POSL
- Can be composed into ('Rosetta') ring for round-tripping between all pairs of Dlex representations

Introduction: Why Study Dlex Kernel and Mappings?

RIF RuleML ...

- ... feature comparison
- ... concrete translator specification
- ... language convergence

Example: Definitions of ‘even’ predicate:

`http://ruleml.org/rif/dlex/RIFRuleMLRosettaEven`

Prepares identifying largest common subset of Datalog RuleML and RIF-Core, also to include “slotted” (RuleML) or “named” (RIF) arguments as well as objects/frames

Introduction: First Part – Specify Dlex Language

- Formalize syntax and semantics of Dlex by specializing relevant definitions of RIF-BLD
- Both new RIF dialect, RIF-Dlex, and new RuleML sublanguage, Dlex RuleML
- Focus on Dlex since
  1. Contains Datalog, which is like relational databases with recursive views
  2. Adding equality plus externals $\rightsquigarrow$ relevant to commercial rule systems
  3. Helps to understand and converge larger RIF and RuleML subsets
- Because RIF-Core (even RIF-BLD) does not support negation, Dlex does not either

Introduction: Second Part – Define Mapping Ring

- Enables round-tripping between all Dlex languages using compositions that chain one to four mappings
- Mappings with XML sources can be refined to XSLT translators
- Focus on mapping
  - $\chi_{\texttt{dlex}}$ from RIF Presentation Syntax to RIF/XML
  - $\xi_{\texttt{dlex}}$ from RIF/XML to Fully Striped RuleML/XML
  - $\pi_{\texttt{dlex}}$ from Stripe-Skipped RuleML/XML to RuleML/POSL

Introduction: Second Part – Define Mapping Ring (Cont'd)

- For ***closed ring***, final (text-to-text) mapping $\omega_{\texttt{dlex}}$ from RuleML/POSL to RIF Presentation Syntax can omit/change separators and insert RIF `Document`s, `Group`s, etc.
- For ***open ring***, 4-mapping composition $\pi_{\texttt{dlex}} \circ \sigma_{\texttt{dlex}} \circ \xi_{\texttt{dlex}} \circ \chi_{\texttt{dlex}}$ complemented by inverse composition $\chi_{\texttt{dlex}}{}^{-1} \circ \xi_{\texttt{dlex}}{}^{-1} \circ \sigma_{\texttt{dlex}}{}^{-1} \circ \pi_{\texttt{dlex}}{}^{-1}$

Syntax of Dlex

Definition (Alphabet)

The ***alphabet*** of the Dlex presentation syntax consists of

- a countably infinite set of ***constant symbols*** `Const`
- a countably infinite set of ***variable symbols*** `Var` (disjoint)
- connective symbols `And` and `:-`
- the quantifier `Forall`
- the symbols `=` and `External`
- the symbols `Group` and `Document`
- the auxiliary symbols `(`, `)`, `<`, `>`, and `^^`

Set of connective symbols, `Forall`, `=` and `External`, etc., disjoint from `Const` and `Var`. Variables written as Unicode strings preceded with symbol "`?`"
Constants written as `"literal"^^symspace`, where `literal` is Unicode string and `symspace` identifies symbol space □

Syntax of Dlex (Cont'd)

In this definition, *base terms* are simple or positional terms, or terms of the form `External(t)`, where `t` is a positional term

Definition (Term)

1. *Constants and variables*. If $t \in Const$ or $t \in Var$ then t is a ***simple term***
2. *Positional terms*. If $t \in Const$ and t_1, ..., t_n, $n \geq 0$, are base terms then $t(t_1 \ldots t_n)$ is a ***positional term.*** Positional terms correspond to usual terms and atomic formulas of classical first-order logic
3. *Equality terms*. $t = s$ is an ***equality term***, if t and s are base terms
4. *Externally defined terms.* If t is a positional term then `External(t)` is an ***externally defined term.*** External terms are used for representing built-in functions and predicates □

Syntax of Dlex (Cont'd)

Dlex distinguishes subsets of set `Const` of symbols, including *predicate symbols* and *function symbols*

Any term of form `p(...)`, where `p` is a predicate symbol, is also an ***atomic formula***

Equality terms are also atomic formulas. An externally defined term of form `External(`φ`)`, where φ is an atomic formula, is also an atomic formula: an ***externally defined*** atomic formula

Definition (Formula, Condition Language)

A ***formula*** can have several different forms and is defined thus:

1. *Atomic*: If φ is an atomic formula then it is also a formula
2. *Condition formula*: A ***condition formula*** is either an atomic formula or a formula that has form of a *conjunction*: If φ_1, ..., φ_n, $n \geq 0$, are condition formulas then so is `And(`φ_1 `...` φ_n`)`, called a *conjunctive* formula

Syntax of Dlex (Cont'd)

Definition (Formula, Rule Language)

3. *Rule implication*: φ `:-` ψ is a formula, called *rule implication*, if:
   - φ is an atomic formula that is not an equality term,
   - ψ is a condition formula, and
   - the atomic formula in φ is not an externally defined term (i.e., a term of the form `External(...)`) and does not have such a term as an argument
4. *Universal rule*: If φ is a rule implication and `?V`$_1$, ..., `?V`$_n$, `n>0`, are distinct variables then `Forall ?V`$_1$ `...` `?V`$_n$ `(`φ`)` is a *universal rule* formula. It is required that all the *free* variables in φ occur among the variables `?V`$_1$ `...` `?V`$_n$ in the quantification part. An occurrence of a variable `?v` is *free* in φ if it is not inside a substring of the form `Q ?v (`ψ`)` of φ, where `Q` is a quantifier (`Forall`) and ψ is a formula. Universal rules will also be called ***Dlex rules***

Syntax of Dlex (Cont'd)

Definition (Formula, Rule Language, Cont'd)

5. *Universal fact*: If φ is an atomic formula that is not an equality term and `?V`$_1$, ..., `?V`$_n$, `n>0`, are distinct variables then `Forall ?V`$_1$ `...` `?V`$_n$`(`φ`)` is a *universal fact* formula, provided that all the free variables in φ occur among the variables `?V`$_1$ `...` `?V`$_n$.
   Universal facts are treated as rules without premises
6. *Group*: If φ_1, ..., φ_n are Dlex rules, universal facts, variable-free rule implications, variable-free atomic formulas, *or* group formulas then `Group(`φ_1 `...` φ_n`)` is a *group formula*.
   Group formulas are used to represent sets of rules and facts. Note that some of the φ_i's can be group formulas themselves, which means that groups can be nested

Syntax of Dlex (Cont'd)

Definition (Formula, Rule Language, Cont'd)

7. *Document*: An expression of form
   `Document(`$directive_1$ ... $directive_n$ Γ`)`
   is a *document formula*, if
   - Γ is an optional group formula; it is called the group formula *associated* with the document
   - $directive_1$, ..., $directive_n$ is an optional sequence of *directives*. We will assume this sequence to be empty for simplicity □

Semantics of Dlex: Semantic Structures

We use ***TV*** as set of semantic truth values {**t**,**f**}

Definition (Semantic structure)

A ***semantic structure***, $\mathcal{I}$, is a tuple of the form
$<\boldsymbol{TV}, \boldsymbol{D}, \boldsymbol{D}_{\text{ind}}, \boldsymbol{D}_{\text{func}}, \boldsymbol{I}_{\text{C}}, \boldsymbol{I}_{\text{V}}, \boldsymbol{I}_{\text{F}}, \boldsymbol{I}_{=}, \boldsymbol{I}_{\text{external}}, \boldsymbol{I}_{\text{truth}}>$

Here ***D*** is a non-empty set of elements called the ***domain*** of $\mathcal{I}$, and $\boldsymbol{D}_{\text{ind}}$, $\boldsymbol{D}_{\text{func}}$ are nonempty subsets of ***D***

$\boldsymbol{D}_{\text{ind}}$ is used to interpret elements of `Const` that occur as individuals and $\boldsymbol{D}_{\text{func}}$ is used to interpret constants that occur as function symbols

As before, `Const` denotes set of all constant symbols and `Var` set of all variable symbols

Semantics of Dlex: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

The other components of $\mathcal{I}$ are *total* mappings defined thus:

1. $\boldsymbol{I}_{\mathrm{C}}$ maps `Const` to $\boldsymbol{D}$.
   This mapping interprets constant symbols.
   In addition, it is required that:
   - If a constant, $\mathtt{c} \in \mathtt{Const}$, is an *individual* then it must be that $\boldsymbol{I}_{\mathrm{C}}(\mathtt{c}) \in \boldsymbol{D}_{\mathrm{ind}}$
   - If $\mathtt{c} \in \mathtt{Const}$, is a *function symbol* then it must be that $\boldsymbol{I}_{\mathrm{C}}(\mathtt{c}) \in \boldsymbol{D}_{\mathrm{func}}$
2. $\boldsymbol{I}_{\mathrm{V}}$ maps `Var` to $\boldsymbol{D}_{\mathrm{ind}}$. Mapping interprets variable symbols
3. $\boldsymbol{I}_{\mathrm{F}}$ maps $\boldsymbol{D}$ to functions $\boldsymbol{D}^*_{\mathrm{ind}} \rightarrow \boldsymbol{D}$ (here $\boldsymbol{D}^*_{\mathrm{ind}}$ is set of all finite sequences over $\boldsymbol{D}_{\mathrm{ind}}$). Interprets positional terms
   - If $\mathtt{d} \in \boldsymbol{D}_{\mathrm{func}}$ then $\boldsymbol{I}_{\mathrm{F}}(\mathtt{d})$ must be a function $\boldsymbol{D}^*_{\mathrm{ind}} \rightarrow \boldsymbol{D}_{\mathrm{ind}}$
   - This implies that when function symbol is applied to arguments that are individual objects then result is also an individual object

Semantics of Dlex: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

4. $\boldsymbol{I}_{=}$ is mapping of form $\boldsymbol{D}_{\text{ind}} \times \boldsymbol{D}_{\text{ind}} \rightarrow \boldsymbol{D}$.
   It gives meaning to equality operator
5. $\boldsymbol{I}_{\text{external}}$ is mapping that is used to give meaning to `External` terms. It maps symbols in `Const` designated as external to *fixed functions* of appropriate arity.
   Typically, external terms are invocations of built-in functions or calls to external definitions, and their fixed interpretations are determined by specification of those built-ins and external definitions
6. $\boldsymbol{I}_{\text{truth}}$ is mapping of form $\boldsymbol{D} \rightarrow \boldsymbol{TV}$.
   It is used to define truth valuation for formulas

Semantics of Dlex: Semantic Structures (Cont'd)

Definition (Semantic structure, Cont'd)

We also define generic mapping from terms to $\boldsymbol{D}$, denoted by $\boldsymbol{I}$ (the same symbol as used to denote semantic structures, but note different font)

- $\boldsymbol{I}(\texttt{k}) = \boldsymbol{I}_{\mathsf{C}}(\texttt{k})$, if $\texttt{k}$ is a symbol in $\texttt{Const}$
- $\boldsymbol{I}(\texttt{?v}) = \boldsymbol{I}_{\mathsf{V}}(\texttt{?v})$, if $\texttt{?v}$ is a variable in $\texttt{Var}$
- $\boldsymbol{I}(\texttt{f}(\texttt{t}_1 \ \ldots \ \texttt{t}_n)) = \boldsymbol{I}_{\mathsf{F}}(\boldsymbol{I}(\texttt{f}))(\boldsymbol{I}(\texttt{t}_1),\ldots,\boldsymbol{I}(\texttt{t}_n))$
- $\boldsymbol{I}(\texttt{x=y}) = \boldsymbol{I}_{=}(\boldsymbol{I}(\texttt{x}), \boldsymbol{I}(\texttt{y}))$
- $\boldsymbol{I}(\texttt{External}(\texttt{p}(\texttt{s}_1 \ \ldots \ \texttt{s}_n))) = \boldsymbol{I}_{\text{external}}(\texttt{p})(\boldsymbol{I}(\texttt{s}_1), \ldots, \boldsymbol{I}(\texttt{s}_n))$

□

Semantics of Dlex: Interpretation of Formulas

Definition (Truth valuation)

Truth valuation for well-formed formulas in Dlex is determined using function $TVal_{\mathcal{I}}$:

1. *Positional atomic formulas*:
   $TVal_{\mathcal{I}}(\texttt{r(t}_1\ \ldots\ \texttt{t}_n\texttt{)}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\texttt{r(t}_1\ \ldots\ \texttt{t}_n\texttt{)}))$
2. *Equality*: $TVal_{\mathcal{I}}(\texttt{x} = \texttt{y}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}(\texttt{x} = \texttt{y}))$
   - To ensure that equality has precisely expected properties, it is required that $TVal_{\mathcal{I}}(\texttt{x} = \texttt{y}) = \mathbf{t}$ if and only if $\boldsymbol{I}(\text{x}) = \boldsymbol{I}(\text{y})$
3. *Externally defined atomic formula*:
   $TVal_{\mathcal{I}}(\texttt{External(t)}) = \boldsymbol{I}_{\text{truth}}(\boldsymbol{I}_{\text{external}}(\texttt{t}))$
4. *Conjunction*: $TVal_{\mathcal{I}}(\texttt{And(}c_1\ \ldots\ c_n\texttt{)}) = \mathbf{t}$ if and only if $TVal_{\mathcal{I}}(c_1) = \ldots = TVal_{\mathcal{I}}(c_n) = \mathbf{t}$.
   Otherwise, $TVal_{\mathcal{I}}(\texttt{And(}c_1\ \ldots\ c_n\texttt{)}) = \mathbf{f}$.
   Empty conjunction is a tautology: $TVal_{\mathcal{I}}(\texttt{And()}) = \mathbf{t}$

Semantics of Dlex: Interpretation of Formulas (Cont'd)

Definition (Truth valuation, Cont'd)

5. *Quantification*:
   - $TVal_{\mathcal{I}}$(`Forall ?v1 ... ?vn (`φ`)`) = **t** if and only if for every $\mathcal{I}^*$, described below, $TVal_{\mathcal{I}*}(\varphi)$ = **t**

   Here $\mathcal{I}^*$ is a semantic structure of form
   $<\boldsymbol{TV}, \boldsymbol{D}, \boldsymbol{D}_{\text{ind}}, \boldsymbol{D}_{\text{func}}, \boldsymbol{I}_{\text{C}}, \boldsymbol{I}^*_{\text{V}}, \boldsymbol{I}_{\text{F}}, \boldsymbol{I}_{=}, \boldsymbol{I}_{\text{external}}, \boldsymbol{I}_{\text{truth}}>$
   Exactly like $\boldsymbol{I}$, except that mapping $\boldsymbol{I}^*_{\text{V}}$, used instead of $\boldsymbol{I}_{\text{V}}$. $\boldsymbol{I}^*_{\text{V}}$ defined to coincide with $\boldsymbol{I}_{\text{V}}$ on all variables except, possibly, on `?v1`,...,`?vn`
6. *Rule implication*:
   - $TVal_{\mathcal{I}}$(*conclusion* :- *condition*) = **t**, if $TVal_{\mathcal{I}}$(*conclusion*) = **t** or $TVal_{\mathcal{I}}$(*condition*) = **f**
   - $TVal_{\mathcal{I}}$(*conclusion* :- *condition*) = **f** otherwise □

Semantics of Dlex: Logical Entailment

Entailment of condition formulas can be viewed as queries to Dlex groups or documents. In definitions, symbol $\models$ generically stands for ***models*** or for ***entails***

Definition (Models)

A semantic structure $\mathcal{I}$ is a ***model*** of a formula, φ, written as $\mathcal{I} \models \varphi$, iff $\mathit{TVal}_{\mathcal{I}}(\varphi) = \mathbf{t}$ □

Definition (Logical entailment)

Let φ and ψ be formulas.
We say that φ ***entails*** ψ, written as $\varphi \models \psi$, if and only if for every semantic structure $\mathcal{I}$ for which both $\mathit{TVal}_{\mathcal{I}}(\varphi)$ and $\mathit{TVal}_{\mathcal{I}}(\psi)$ are defined, $\mathcal{I} \models \varphi$ implies $\mathcal{I} \models \psi$ □

Mapping of the Dlex Condition Language

Paper formalizes chain of semantics-preserving Dlex mappings $\chi_{\texttt{dlex}}$, $\xi_{\texttt{dlex}}$, and $\pi_{\texttt{dlex}}$ for the Condition Languages of Datalog RuleML and RIF-Core

A Dlex mapping μ is ***semantics-preserving*** if for any pair φ, ψ of *Dlex* formulas for which $\varphi \models_K \psi$ is defined, $\varphi \models_K \psi$ if and only if $\mu(\varphi) \models_L \mu(\psi)$

Here $\models_K$ and $\models_L$ denote logical entailment in, respectively, source language *K* and target language *L* of μ

Mapping of the Dlex Condition Language (Cont'd)

Tables below define mappings μ, each row specifying translation $\mu(\texttt{column}_1) = \texttt{column}_2$

$\texttt{column}_2$ often contains applications of mapping μ to these ***metavariables***

When expression $\mu\,(\texttt{***metavar***})$ occurs in $\texttt{column}_2$,
it is understood as recursive application of μ to ***metavar***.
Result of application is substituted for expression $\mu\,(\texttt{***metavar***})$

The μ inverse, μ^{-1}, obtained by reading table right-to-left while replacing right-hand-side recursive μ applications with left-hand-side recursive μ^{-1} applications

Mapping of the Dlex Condition Language: From RIF/XML to RuleML/XML

XML-to-XML mapping $\xi_{\texttt{dlex}}$ from RIF/XML to RuleML/XML is central to chain of mappings, comparing, and bridging between, kernels of RIF and RuleML

To reduce mapping 'distance', *Fully Striped* RuleML/XML used as target (i.e., second column) of $\xi_{\texttt{dlex}}$

Stripe-skipping via mapping $\sigma_{\texttt{dlex}}$ can be employed to obtain more compact *Stripe-Skipped* RuleML/XML, chaining to source (i.e., first column) of mapping $\pi_{\texttt{dlex}}$

RIF/XML	RuleML/XML (Fully Striped)
<And> <formula>***conjunct*$_1$**</formula> . . . <formula>***conjunct*$_n$**</formula> </And>	<And> <formula>ξ_{dlex}(***conjunct*$_1$**)</formula> . . . <formula>ξ_{dlex}(***conjunct*$_n$**)</formula> </And>
<External> <content>***atomexpr***</content> </External>	ξ_{dlex}(***atomexpr***)
<Atom> <op>***pred***</op> <args ordered="yes"> ***argument*$_1$** . . . ***argument*$_n$** </args> </Atom>	<Atom> <op>ξ_{dlex}(***pred***)</op> <arg index="1"> ξ_{dlex}(***argument*$_1$**) </arg> . . . <arg index="***n***"> ξ_{dlex}(***argument*$_n$**) </arg> </Atom>

continued on next page

continued from previous page

<Expr>
  <op>**func**</op>
  <args ordered="yes">
  argument$_1$
  . . .
  argument$_n$
  </args>
</Expr>
	<Expr>
	  <op>ξ_{dlex}(**func**)</op>
	  <arg index="1">
	  ξ_{dlex}(**argument**$_1$)
	  </arg>
	  . . .
	  <arg index="**n**">
	  ξ_{dlex}(**argument**$_n$)
	  </arg>
	</Expr>

<Equal> <left>**left**</left> <right>**right**</right> </Equal>	<Equal> <left>ξ_{dlex}(**left**)</left> <right>ξ_{dlex}(**right**)</right> </Equal>

<Const
  type="**st**">**unicodestring**</Const>
	<Const
	  type="**st**">**unicodestring**</Const>

<Var>**unicodestring**</Var>	<Var>**unicodestring**</Var>

Mapping of the Dlex Condition Language: From RIF/XML to RuleML/XML (Cont'd)

RIF employs `External` wrappers around every built-in call. RuleML distinguishes user-defined/built-in calls via attribute. A RIF advantage: explicitness. The RuleML advantage: no need to (structurally) change syntax at all call occurrences when user-defined operator becomes 'compiled' into built-in

RIF built-ins themselves are adopted by RuleML: Unlike earlier SWRL built-ins, which write n-ary functions as (1+n)-ary relations, functional RIF built-ins remain functions

RIF: `args` role tag for argument sequence. RuleML: `arg` role tags for arguments separately – omitted via stripe-skipping. RIF **and** RuleML methods should be permitted in RIF RuleML

RIF `Const`ants use symbol spaces as values of their mandatory `type` attribute. RuleML `Ind`ividuals and `Const`ants use optional types, which can be symbol spaces

Unique inverse $\xi_{\texttt{dlex}}{}^{-1}$ obtainable: $\xi_{\texttt{dlex}}$ semantics-preserving

Mapping of the Dlex Rule Language: From RIF/XML to RuleML/XML

Paper extends chain of Dlex Condition Language mappings $\chi_{\texttt{dlex}}$, $\xi_{\texttt{dlex}}$, and $\pi_{\texttt{dlex}}$

to a chain of semantics-preserving Dlex Rule Language mappings for Datalog RuleML and RIF-Core

Central Condition Language mapping $\xi_{\texttt{dlex}}$ from RIF/XML to RuleML/XML is extended here for rules

RIF/XML	RuleML/XML (Fully Striped)
<Document> <payload>***group***</payload> </Document>	<RuleML> <performative index="1"> <Assert> <formula> ξ_{dlex}(***group***) </formula> </Assert> </performative> </RuleML>

<Group>
  <sentence>$\boldsymbol{clause_1}$</sentence>
  . . .
  <sentence>$\boldsymbol{clause_n}$</sentence>
</Group>
	<Rulebase>
	  <formula>ξ_{dlex}(***clause***$_1$)</formula>
	  . . .
	  <formula>ξ_{dlex}(***clause***$_n$)</formula>
	</Rulebase>

<Forall>
  <declare>$\boldsymbol{variable_1}$</declare>
  . . .
  <declare>$\boldsymbol{variable_n}$</declare>
  <formula>***rule***</formula>
</Forall>
	<Forall>
	  <declare>ξ_{dlex}(***variable***$_1$)</declare>
	  . . .
	  <declare>ξ_{dlex}(***variable***$_n$)</declare>
	  <formula>ξ_{dlex}(***rule***)</formula>
	</Forall>

<Implies> <if>***condition***</if> <then>***conclusion***</then> </Implies>	<Implies> <if>ξ_{dlex}(***condition***)</if> <then>ξ_{dlex}(***conclusion***)</then> </Implies>

Mapping of the Dlex Rule Language: From RIF/XML to RuleML/XML (Cont'd)

The first row maps the RIF root element `Document` to the RuleML root element `RuleML`, which permits (ordered) transactions of performatives including `Assert`s

The second row maps a RIF `Group` to a RuleML `Rulebase`

While a general RIF `Document` can also contain directives for a base, prefixes, and imports, a general `RuleML` root can also contain `Query` and `Retract` transactions

But for our special case the unique inverse $\xi_{\texttt{dlex}}{}^{-1}$ can be obtained, keeping $\xi_{\texttt{dlex}}$ semantics-preserving

(Regarding last table row, the RuleML 0.91 upgrade to 0.95 adopts RIF role tags `<if>` ... `<then>`, also long contemplated in RuleML, instead of `<body>` ... `<head>`)

Conclusion

- **Dlex kernel** of **RIF RuleML overlap** study enables round-tripping and convergence
- Positional Dlex to be expanded for remaining features of Datalog RuleML / RIF-Core, e.g. slotted/named arguments, objects/frames, and IRIs as attributes/elements
- Since our Dlex mappings already include (interpreted) functions for external calls, study should also be extended to overlap of Hornlog RuleML and RIF-BLD using (uninterpreted) functions for Herbrand terms

Conclusion (Cont'd)

- Datalog RuleML is complemented by Naf Datalog RuleML, a sublanguage with often needed Negation as failure, which is also carried through to Naf Hornlog RuleML
- Since Naf constructs are wide-spread in practice, most rule engines have implemented them (including Scoped Naf in cwm/AIR and Euler)
- So their interoperation is already a focus of RIF RuleML efforts (cf. our use case: `http://ruleml.org/WellnessRules`)

Conclusion (Cont'd)

On the next level, the development of the RIF Production Rule Dialect & Reaction RuleML towards a RIF RuleML Reaction Rule Dialect

will provide further opportunities for joint work between W3C, OMG, and RuleML, as needed by the industry