NRC Publications Archive Archives des publications du CNRC

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

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version. /
La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version
acceptée du manuscrit ou la version de l'éditeur.

Publisher's version / Version de l'éditeur:

*Proceedings of the 3rd International Symposium on Rules, Applications and
Interoperability, 2009, 2009-11-07*

NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=b66d1ce8-c812-42ad-9306-91c0aa7ce91b>
<https://publications-cnrc.canada.ca/fra/voir/objet/?id=b66d1ce8-c812-42ad-9306-91c0aa7ce91b>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

Questions? Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the
first page of the publication for their contact information.

Vous avez des questions? Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la
première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez
pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.

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

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 then defined between the Dlex subset of the RIF Presentation Syntax and RIF/XML, RIF/XML and RuleML/XML, as well as RuleML/XML and the Prolog-like RuleML/POSL. These mappings 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.

1 Introduction

The RuleML family of languages [Bol07] has recently been enhanced by RIF dialects starting with the Basic Logic Dialect [BK09a]. As RuleML moves to version 1.0 and RIF to Recommendation, the overlap area between RIF and RuleML is of broad interest for Web rule interchange. In this paper, we focus on its kernel, Dlex, defined as the common sublanguage of Datalog RuleML [Bol06] and RIF-Core [BHK⁺09] restricted to positional arguments and non-conjunctive rule conclusions, and allowing equality plus externals – built-ins as defined in RIF-DTB [PBK09] – in rule premises (only).

To advance RIF RuleML development and interoperation, we define semantics-preserving mappings within this overlap, in particular within Dlex. These include central mappings bridging RIF/XML and RuleML/XML, mappings between the RIF Presentation Syntax and RIF/XML, as well as mappings between RuleML/XML and the Prolog-like RuleML/POSL. The slightly augmented mappings can be composed into a ('Rosetta') ring for round-tripping between all pairs of Dlex representations.

This study on the Dlex kernel and its mappings includes a RIF RuleML feature comparison, acts as a specification of concrete translators between RIF and RuleML, and helps in language convergence; definitions of an 'even' predicate exemplify.¹ Moreover, it prepares identifying the largest common subset of Datalog RuleML and RIF-Core, also to include "slotted" (RuleML) or "named" (RIF) arguments as well as objects/frames.

First, we formalize the syntax and semantics of Dlex by specializing relevant definitions of RIF-BLD [BK09a], since the specialization of RIF-BLD to RIF-Core is currently only done 'by exclusion' [BHK⁺09]. This formalization can be seen both as a new RIF dialect, RIF-Dlex, and a new RuleML sublanguage, Dlex RuleML. We focus on

¹ <http://ruleml.org/rif/dlex/RIFRuleMLRosettaEven>

Dlex since (1) it contains Datalog [CGT89], which is close to relational databases with recursive views, (2) with equality plus externals added, it is relevant to various commercial rule systems, (3) it is a kernel of RIF and RuleML, also helping to understand and converge their larger subsets. Because RIF-Core (and even RIF-BLD) does not support negation, its Dlex sublanguage does not either (cf. section 6). Following RuleML's SWSL subfamily² and RIF's overall design, Dlex uses Hilog/Relfun-like higher-order syntactic sugar with individual (**Ind**) and predicate (**Rel**) symbols merged into a single set of constant (**Const**) symbols. The Dlex formalization provides a foundation for RIF RuleML mapping as needed for our round-tripping.

Second, instead of defining pairwise Dlex mappings or mappings to and fro a distinguished Dlex representative, we define a neutral 'mapping ring', enabling round-tripping between all of these languages using compositions that chain one to four mappings. The mappings that have XML as their sources can be refined to XSLT translators. The other ones are handled by text parsers and generators.

We focus on a mapping χ_{dlex} from RIF Presentation Syntax to RIF/XML [BK09a], a mapping ξ_{dlex} from RIF/XML to Fully Striped RuleML/XML, and a mapping π_{dlex} from Stripe-Skipped RuleML/XML to RuleML/POSL. A further mapping σ_{dlex} from Fully Striped to Stripe-Skipped RuleML/XML essentially omits all (lower-case) role tags.³ For a *closed ring*, a final (text-to-text) mapping ω_{dlex} from RuleML/POSL to RIF Presentation Syntax can omit/change separators and insert RIF **Documents**, **Groups**, etc. For an *open ring*, the 4-mapping composition $\pi_{\text{dlex}} \circ \sigma_{\text{dlex}} \circ \xi_{\text{dlex}} \circ \chi_{\text{dlex}}$ can instead be complemented by the inverse composition $\chi_{\text{dlex}}^{-1} \circ \xi_{\text{dlex}}^{-1} \circ \sigma_{\text{dlex}}^{-1} \circ \pi_{\text{dlex}}^{-1}$.

The rest of the paper is organized as follows. Dlex will be defined next. The syntax, in section 2; the semantics, in section 3. The mappings of the RIF Presentation Syntax to RIF/XML (χ_{dlex}) to RuleML/XML (ξ_{dlex}) to RuleML/POSL (π_{dlex}) will then be specified for Dlex via two chains of tables, and their inverses discussed (for semantics preservation and open-ring round-tripping). The chain in section 4 defines those mappings for the Condition Language (conjunctive queries, as in rule premises); the other, in section 5, for the Rule Language (entire rules and rulebases); we will omit annotations and their mappings. Conclusions will be given in section 6.

2 Syntax of Dlex

We begin with defining the syntax of Dlex, starting from the alphabet, proceeding to terms, and then to formulas.

2.1 Alphabet of Dlex

Definition 1 (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 from **Const**)
- connective symbols **And** and **:-**
- the quantifier **Forall**
- the symbols **=** and **External**
- the symbols **Group** and **Document**
- the auxiliary symbols **(, , <, >, and ^**.

² <http://www.w3.org/Submission/SWSF-SWSL>

³ The inverse $\sigma_{\text{dlex}}^{-1}$ of σ_{dlex} is more difficult as it infers and inserts role tags; see <http://ruleml.org/0.89/#XSLT-BasedNormalizer>.

The set comprising the connective symbols, *Forall*, = and *External*, etc., is disjoint from *Const* and *Var*. Variables are written as Unicode strings preceded with the symbol “?”.²

Constants are written as “*literal*”³ *symspace*, where *literal* is a sequence of Unicode characters and *symspace* is an identifier for a symbol space. $\square$

2.2 Terms

Dlex defines several kinds of terms: *constants* and *variables*, *positional* terms, plus *equality* and *external* terms.

In the next definition, the phrase *base term* refers to simple or positional terms, or to terms of the form *External*(*t*), where *t* is a positional term.

Definition 2 (Term).

1. *Constants and variables.* If $t \in \text{Const}$ or $t \in \text{Var}$ then t is a **simple term**.
2. *Positional terms.* If $t \in \text{Const}$ and $t_1, \dots, t_n, n \geq 0$, are base terms then $t(t_1 \dots t_n)$ is a **positional term**.
Positional terms correspond to the 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. $\square$

2.3 Formulas

Dlex distinguishes certain subsets of the set *Const* of symbols, including subsets of *predicate symbols* and *function symbols*.

Any term of the form *p*(...), where *p* is a predicate symbol, is also an **atomic formula**. Equality terms are also atomic formulas. An externally defined term of the form *External*(φ), where φ is an atomic formula, is also an atomic formula, called an **externally defined** atomic formula. Note that simple terms (constants and variables) are *not* formulas.

More general formulas are constructed out of atomic formulas with the help of logical connectives.

Definition 3 (Formula).

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

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 the form of a conjunction: If $\varphi_1, \dots, \varphi_n, n \geq 0$, are condition formulas then so is *And*($\varphi_1 \dots \varphi_n$), called a **conjunctive formula**.
3. *Rule implication:* $\varphi :- \psi$ 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 $?V_1, \dots, ?V_n, n > 0$, are distinct variables then **Forall** $?V_1 \dots ?V_n(\varphi)$ is a universal rule formula. It is required that all the free variables in φ occur among the variables $?V_1 \dots ?V_n$ in the quantification part. An occurrence of a variable $?v$ is free in φ if it is not inside a substring of the form $Q ?v(\psi)$ of φ , where Q is a quantifier (**Forall**) and ψ is a formula. Universal rules will also be referred to as **Dlex rules**.
5. *Universal fact*: If φ is an atomic formula that is not an equality term and $?V_1, \dots, ?V_n, n > 0$, are distinct variables then **Forall** $?V_1 \dots ?V_n(\varphi)$ is a universal fact formula, provided that all the free variables in φ occur among the variables $?V_1 \dots ?V_n$.
Universal facts are treated as rules without premises.
6. *Group*: If $\varphi_1, \dots, \varphi_n$ are Dlex rules, universal facts, variable-free rule implications, variable-free atomic formulas, or group formulas then **Group**($\varphi_1 \dots \varphi_n$) 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.
7. *Document*: An expression of the form **Document**(directive₁ ... directive_n Γ) is a Dlex document formula (or simply a document formula), if
 - Γ is an optional group formula; it is called the group formula associated with the document.
 - directive₁, ..., directive_n is an optional sequence of directives. We will assume this sequence [BK09a] to be empty for simplicity. $\square$

The Dlex *Condition Language* comprises formulas according to definition 3, items 1 and 2; its *Rule Language* is the superset defined via formulas 4-7.

3 Semantics of Dlex

Given the syntax, we now define the semantics of Dlex via semantic structures and the interpretation of formulas, finishing with logical entailment.

3.1 Semantic Structures

We will use $\mathbf{TV}$ to denote $\{\mathbf{t}, \mathbf{f}\}$, the set of truth values used in the semantics. The key concept in a model-theoretic semantics of a logic language is the notion of *semantic structures*, which is defined below.

Definition 4 (Semantic structure). A *semantic structure*, $\mathcal{I}$, is a tuple of the form $\langle \mathbf{TV}, \mathbf{D}, \mathbf{D}_{ind}, \mathbf{D}_{func}, \mathbf{I}_C, \mathbf{I}_V, \mathbf{I}_F, \mathbf{I}_=, \mathbf{I}_{external}, \mathbf{I}_{truth} \rangle$. Here $\mathbf{D}$ is a non-empty set of elements called the **domain** of $\mathcal{I}$, and $\mathbf{D}_{ind}, \mathbf{D}_{func}$ are nonempty subsets of $\mathbf{D}$. $\mathbf{D}_{ind}$ is used to interpret the elements of **Const** that occur as individuals and $\mathbf{D}_{func}$ is used to interpret the constants that occur as function symbols. As before, **Const** denotes the set of all constant symbols and **Var** the set of all variable symbols.

The remaining components of $\mathcal{I}$ are total mappings defined as follows:

1. $\mathbf{I}_C$ maps **Const** to $\mathbf{D}$.
This mapping interprets constant symbols. In addition, it is required that:
 - If a constant, $c \in \mathbf{Const}$, is an individual then it must be that $\mathbf{I}_C(c) \in \mathbf{D}_{ind}$.
 - If $c \in \mathbf{Const}$, is a function symbol then it must be that $\mathbf{I}_C(c) \in \mathbf{D}_{func}$.
2. $\mathbf{I}_V$ maps **Var** to $\mathbf{D}_{ind}$.
This mapping interprets variable symbols.

3. I_F maps D to functions $D^*_{ind} \rightarrow D$ (here D^*_{ind} is a set of all finite sequences over the domain D_{ind}).
This mapping interprets positional terms. In addition:
 - If $d \in D_{func}$ then $I_F(d)$ must be a function $D^*_{ind} \rightarrow D_{ind}$.
 - This implies that when a function symbol is applied to arguments that are individual objects then the result is also an individual object.
4. $I_=$ is a mapping of the form $D_{ind} \times D_{ind} \rightarrow D$.
It gives meaning to the equality operator.
5. I_{truth} is a mapping of the form $D \rightarrow TV$.
It is used to define truth valuation for formulas.
6. $I_{external}$ is a 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 the specification of those built-ins and external definitions.

We also define the following generic mapping from terms to D , which we denote by I (this is the same symbol as the one used to denote semantic structures, but note the different font).

- $I(k) = I_C(k)$, if k is a symbol in **Const**
- $I(?v) = I_V(?v)$, if $?v$ is a variable in **Var**
- $I(f(t_1 \dots t_n)) = I_F(I(f))(I(t_1), \dots, I(t_n))$
- $I(x=y) = I_=(I(x), I(y))$
- $I(\text{External}(p(s_1 \dots s_n))) = I_{external}(p)(I(s_1), \dots, I(s_n))$. □

3.2 Interpretation of Formulas

It can now be established how semantic structures determine the truth value of Dlex formulas, excluding group and document formulas [BK09a] for simplicity. Here we define a mapping, $TVal_{\mathcal{I}}$, from the set of all formulas to TV .

Definition 5 (Truth valuation). *Truth valuation* for well-formed formulas in Dlex is determined using the following function, denoted $TVal_{\mathcal{I}}$:

1. Positional atomic formulas: $TVal_{\mathcal{I}}(r(t_1 \dots t_n)) = I_{truth}(I(r(t_1 \dots t_n)))$
2. Equality: $TVal_{\mathcal{I}}(x=y) = I_{truth}(I(x=y))$.
– To ensure that equality has precisely the expected properties, it is required that $TVal_{\mathcal{I}}(x=y) = t$ if and only if $I(x) = I(y)$.
3. Externally defined atomic formula: $TVal_{\mathcal{I}}(\text{External}(t)) = I_{truth}(I_{external}(t))$.
4. Conjunction: $TVal_{\mathcal{I}}(\text{And}(c_1 \dots c_n)) = t$ if and only if $TVal_{\mathcal{I}}(c_1) = \dots = TVal_{\mathcal{I}}(c_n) = t$. Otherwise, $TVal_{\mathcal{I}}(\text{And}(c_1 \dots c_n)) = f$.
The empty conjunction is treated as a tautology, so $TVal_{\mathcal{I}}(\text{And}()) = t$.
5. Quantification:
 - $TVal_{\mathcal{I}}(\text{Forall } ?v_1 \dots ?v_n (\varphi)) = t$ if and only if for every $\mathcal{I}^*$, described below, $TVal_{\mathcal{I}^*}(\varphi) = t$.
 Here $\mathcal{I}^*$ is a semantic structure of the form $\langle TV, D, D_{ind}, D_{func}, I_C, I^*_V, I_F, I_=(, I_{external}, I_{truth} \rangle$, which is exactly like I , except that the mapping I^*_V , is used instead of I_V . I^*_V is defined to coincide with I_V on all variables except, possibly, on $?v_1, \dots, ?v_n$.
6. Rule implication:
 - $TVal_{\mathcal{I}}(\text{conclusion} :- \text{condition}) = t$, if either $TVal_{\mathcal{I}}(\text{conclusion})=t$ or $TVal_{\mathcal{I}}(\text{condition})=f$.
 - $TVal_{\mathcal{I}}(\text{conclusion} :- \text{condition}) = f$ otherwise. □

3.3 Logical Entailment

We now define what it means for a set of Dlex rules (in a group or a document formula) to entail another Dlex formula. In Dlex we are mostly interested in entailment of condition formulas, which can be viewed as queries to Dlex groups or documents. In the definitions, the symbol $\models$ will generically stand for *models* or for *entails*.

Definition 6 (Models). *A semantic structure $\mathcal{I}$ is a **model** of a formula, φ , written as $\mathcal{I} \models \varphi$, iff $TVal_{\mathcal{I}}(\varphi) = t$. $\square$*

Definition 7 (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 $TVal_{\mathcal{I}}(\varphi)$ and $TVal_{\mathcal{I}}(\psi)$ are defined, $\mathcal{I} \models \varphi$ implies $\mathcal{I} \models \psi$. $\square$*

4 Mapping of the Dlex Condition Language

We proceed to formalizing the chain of semantics-preserving Dlex mappings χ_{dlex} , ξ_{dlex} , and π_{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, the source language K and target language L of μ .

In each table below defining a mapping μ , each row specifies a translation $\mu(\text{column}_1) = \text{column}_2$. The **bold-italic** symbols represent metavariables. (A sequence of terms containing metavariables with subscripts is indicated by an ellipsis.) column_2 often contains applications of the mapping μ to these metavariables. When an expression $\mu(\text{metavar})$ occurs in column_2 , it should be understood as a recursive application of μ to metavar . The result of such an application is substituted for the expression $\mu(\text{metavar})$. The μ inverse, μ^{-1} , is obtained by reading the table right-to-left while replacing right-hand-side recursive μ applications with corresponding left-hand-side recursive μ^{-1} applications.

4.1 Conditions from RIF Presentation Syntax to RIF/XML

The text-to-XML mapping χ_{dlex} from the RIF Presentation Syntax of Dlex to RIF/XML specializes a mapping χ_{core} for RIF-Core, which itself is a specialization of χ_{bld} from [BK09a]. We will thus start the mapping chain with the subset of the RIF Presentation Syntax that was defined as the Dlex Presentation Syntax in section 2 and used to specify the Dlex semantics in section 3.

Since the RIF Presentation Syntax is context sensitive, the mapping χ_{dlex} must differentiate between terms that occur in the position of individuals and terms that occur as atomic formulas. To this end, the terms that occur in the context of atomic formulas are denoted by expressions of the form **pred**(...) and those that occur as individuals are denoted by expressions of the form **func**(...). In RIF-Core, however, **func**(...) is only needed for built-in calls.

RIF Presentation Syntax	RIF/XML
$\text{And } ($ $\quad conjunct_1$ $\quad \dots$ $\quad conjunct_n$ $\quad)$	$\langle \text{And} \rangle$ $\quad \langle \text{formula} \rangle \chi_{\text{dlex}}(conjunct_1) \langle / \text{formula} \rangle$ $\quad \dots$ $\quad \langle \text{formula} \rangle \chi_{\text{dlex}}(conjunct_n) \langle / \text{formula} \rangle$ $\langle / \text{And} \rangle$
$\text{External } ($ $\quad atomezpr$ $\quad)$	$\langle \text{External} \rangle$ $\quad \langle \text{content} \rangle \chi_{\text{dlex}}(atomezpr) \langle / \text{content} \rangle$ $\langle / \text{External} \rangle$
$\text{pred } ($ $\quad argument_1$ $\quad \dots$ $\quad argument_n$ $\quad)$	$\langle \text{Atom} \rangle$ $\quad \langle \text{op} \rangle \chi_{\text{dlex}}(\text{pred}) \langle / \text{op} \rangle$ $\quad \langle \text{args ordered} = \text{"yes"} \rangle$ $\quad \quad \chi_{\text{dlex}}(argument_1)$ $\quad \quad \dots$ $\quad \quad \chi_{\text{dlex}}(argument_n)$ $\quad \langle / \text{args} \rangle$ $\langle / \text{Atom} \rangle$
$\text{func } ($ $\quad argument_1$ $\quad \dots$ $\quad argument_n$ $\quad)$	$\langle \text{Expr} \rangle$ $\quad \langle \text{op} \rangle \chi_{\text{dlex}}(\text{func}) \langle / \text{op} \rangle$ $\quad \langle \text{args ordered} = \text{"yes"} \rangle$ $\quad \quad \chi_{\text{dlex}}(argument_1)$ $\quad \quad \dots$ $\quad \quad \chi_{\text{dlex}}(argument_n)$ $\quad \langle / \text{args} \rangle$ $\langle / \text{Expr} \rangle$
$\text{left} = \text{right}$	$\langle \text{Equal} \rangle$ $\quad \langle \text{left} \rangle \chi_{\text{dlex}}(\text{left}) \langle / \text{left} \rangle$ $\quad \langle \text{right} \rangle \chi_{\text{dlex}}(\text{right}) \langle / \text{right} \rangle$ $\langle / \text{Equal} \rangle$
$\text{"unicodestring"}^{\wedge \text{sym space}}$	$\langle \text{Const}$ $\quad \text{type} = \text{"sym space"} \rangle \text{unicodestring} \langle / \text{Const} \rangle$
$? \text{unicodestring}$	$\langle \text{Var} \rangle \text{unicodestring} \langle / \text{Var} \rangle$

The table shows that the functional-style terms of the RIF Presentation Syntax are mapped to RIF/XML elements by inserting role tags such as `formula` in `And` terms. The `Atom` and `Expr` elements serialize n-ary *pred*(...) and *func*(...) applications, respectively, using two role tags and moving the n-ary branching underneath the second role tag, `args`. Since the RIF/XML serialization provides explicit markup, the unique inverse χ_{dlex}^{-1} is easily obtained. The round-tripping composition $\chi_{\text{dlex}}^{-1} \circ \chi_{\text{dlex}}$ is the identity, so χ_{dlex} is semantics-preserving.

4.2 Conditions from RIF/XML to RuleML/XML

The XML-to-XML mapping ξ_{dlex} from RIF/XML to RuleML/XML is central to our chain of mappings, as it allows to compare, and bridge between, the kernels of RIF and RuleML. To reduce the mapping ‘distance’, Fully Striped RuleML/XML is used as the target (i.e., second column) of ξ_{dlex} . Stripe-skipping via a mapping σ_{dlex} (cf. section 1) can then be employed to obtain the more compact Stripe-Skipped RuleML/XML, chaining to the source (i.e., first column) of the mapping π_{dlex} in section 4.3.

RIF/XML	RuleML/XML (Fully Striped)
<pre> <And> <formula>conjunct₁</formula> . . . <formula>conjunct_n</formula> </And> </pre>	<pre> <And> <formula>$\xi_{dlex}(conjunct_1)$</formula> . . . <formula>$\xi_{dlex}(conjunct_n)$</formula> </And> </pre>
<pre> <External> <content>atome_{expr}</content> </External> </pre>	<pre> $\xi_{dlex}(atome_{expr})$ </pre>
<pre> <Atom> <op>$\xi_{dlex}(pred)$</op> <args ordered="yes"> $\xi_{dlex}(argument_1)$. . . $\xi_{dlex}(argument_n)$ </args> </Atom> </pre>	<pre> <Atom> <op>$\xi_{dlex}(pred)$</op> <arg index="1"> $\xi_{dlex}(argument_1)$ </arg> . . . <arg index="n"> $\xi_{dlex}(argument_n)$ </arg> </Atom> </pre>
<pre> <Expr> <op>$\xi_{dlex}(func)$</op> <args ordered="yes"> $\xi_{dlex}(argument_1)$. . . $\xi_{dlex}(argument_n)$ </args> </Expr> </pre>	<pre> <Expr> <op>$\xi_{dlex}(func)$</op> <arg index="1"> $\xi_{dlex}(argument_1)$ </arg> . . . <arg index="n"> $\xi_{dlex}(argument_n)$ </arg> </Expr> </pre>
<pre> <Equal> <left>left</left> <right>right</right> </Equal> </pre>	<pre> <Equal> <left>$\xi_{dlex}(left)$</left> <right>$\xi_{dlex}(right)$</right> </Equal> </pre>
<pre> <Const type="st">unicodestring</Const> </pre>	<pre> <Const type="st">unicodestring</Const> </pre>
<pre> <Var>unicodestring</Var> </pre>	<pre> <Var>unicodestring</Var> </pre>

As can be seen in the table, the commonalities between RIF and RuleML prevail, but the following differences should be noted:

While RIF employs **External** wrappers around every built-in call, RuleML makes the distinction between user-defined and built-in calls via an optional attribute on the operators (not shown here). A RIF advantage here may be explicitness. The RuleML advantage is, as in Lisp, that there is no need to (structurally) change the syntax at all call occurrences when a user-defined operator becomes ‘compiled’ into a built-in. Note that the RIF built-ins themselves, currently in version RIF-DTB 1.0 [PBK09], are adopted by RuleML: Unlike the earlier SWRL built-ins, which write n -ary functions as $(1+n)$ -ary relations [HPSB⁺04], functional RIF built-ins remain functions. While RIF uses an **args** role tag around the sequence of all arguments to predicates and functions, RuleML uses **arg** role tags around each argument separately, which can be omitted thanks to stripe-skipping. We propose that both the RIF and RuleML methods should be permitted in RIF RuleML. While RIF **Constants** use symbol spaces as values of their mandatory **type** attribute, RuleML **Individuals** and neutralized **Constants** use optional **types**, which can be symbol spaces.

Taking care of these differences, the unique inverse ξ_{dlex}^{-1} can be obtained, making ξ_{dlex} semantics-preserving.

4.3 Conditions from RuleML/XML to RuleML/POSL

The XML-to-text mapping π_{dlex} from Stripe-Skipped RuleML/XML to RuleML/POSL generates Prolog-like conditions (where, however, variables are indicated by a “?”-prefix rather than by capitalization).

RuleML/XML (Stripe-Skipped)	RuleML/POSL
<code><And></code> <code> <i>conjunct</i>₁</code> <code> . . .</code> <code> <i>conjunct</i>_{n-1}</code> <code> <i>conjunct</i>_n</code> <code></And></code>	$\pi_{\text{dlex}}(\textit{conjunct}_1),$ $\dots$ $\pi_{\text{dlex}}(\textit{conjunct}_{n-1}),$ $\pi_{\text{dlex}}(\textit{conjunct}_n)$
<code><Atom></code> <code> <i>pred</i></code> <code> <i>argument</i>₁</code> <code> . . .</code> <code> <i>argument</i>_{n-1}</code> <code> <i>argument</i>_n</code> <code></Atom></code>	$\pi_{\text{dlex}}(\textit{pred}) ($ $\quad \pi_{\text{dlex}}(\textit{argument}_1),$ $\quad \dots$ $\quad \pi_{\text{dlex}}(\textit{argument}_{n-1}),$ $\quad \pi_{\text{dlex}}(\textit{argument}_n)$ $)$
<code><Expr></code> <code> <i>func</i></code> <code> <i>argument</i>₁</code> <code> . . .</code> <code> <i>argument</i>_{n-1}</code> <code> <i>argument</i>_n</code> <code></Expr></code>	$\pi_{\text{dlex}}(\textit{func}) ($ $\quad \pi_{\text{dlex}}(\textit{argument}_1),$ $\quad \dots$ $\quad \pi_{\text{dlex}}(\textit{argument}_{n-1}),$ $\quad \pi_{\text{dlex}}(\textit{argument}_n)$ $)$
<code><Equal></code> <code> <i>left</i></code> <code> <i>right</i></code> <code></Equal></code>	<code>equal (</code> <code> $\pi_{\text{dlex}}(\textit{left}),$</code> <code> $\pi_{\text{dlex}}(\textit{right})$)</code>
<code><Const</code> <code> type="st"><i>unicodestring</i></Const></code>	<code><i>unicodestring:st</i></code>
<code><Var><i>unicodestring</i></Var></code>	<code>?<i>unicodestring</i></code>

Note that the `type` attribute of RIF RuleML XML serializations becomes a “:”-infix in POSL, which – closing the mapping ring with mapping ω_{dlex} – in RIF Presentation Syntax is written as a “^^”-infix. As a term-type separator, the POSL use of “:” is modeled on “:” in sorted logics, while the Presentation Syntax use of “^^” is modeled on N3/Turtle (where “:” is needed for namespaces).

The generator π_{dlex} is semantics-preserving since it can be uniquely inverted to the parser π_{dlex}^{-1} , although parsing is more difficult, as always. Both π_{dlex} and the inverse text-to-XML mapping π_{dlex}^{-1} are supported by online translators under a single GUI (<http://www.ruleml.org/posl/converter.jsp>).

5 Mapping of the Dlex Rule Language

We now extend the chain of Dlex Condition Language mappings χ_{dlex} , ξ_{dlex} , and π_{dlex} of section 4 to a chain of semantics-preserving Dlex Rule Language mappings for Datalog RuleML and RIF-Core.

5.1 Rules from RIF Presentation Syntax to RIF/XML

The mapping χ_{dlex} of section 4.1 from the RIF Presentation Syntax to RIF/XML is extended here for rules.

RIF Presentation Syntax	RIF/XML
Document(<i>group</i>)	<Document> <payload> $\chi_{\text{dlex}}(\textit{group})$ </payload> </Document>
Group(<i>clause₁</i> . . . <i>clause_n</i>)	<Group> <sentence> $\chi_{\text{dlex}}(\textit{clause}_1)$ </sentence> . . . <sentence> $\chi_{\text{dlex}}(\textit{clause}_n)$ </sentence> </Group>
Forall <i>variable₁</i> . . . <i>variable_n</i> (<i>rule</i>)	<Forall> <declare> $\chi_{\text{dlex}}(\textit{variable}_1)$ </declare> . . . <declare> $\chi_{\text{dlex}}(\textit{variable}_n)$ </declare> <formula> $\chi_{\text{dlex}}(\textit{rule})$ </formula> </Forall>
<i>conclusion</i> :- <i>condition</i>	<Implies> <if> $\chi_{\text{dlex}}(\textit{condition})$ </if> <then> $\chi_{\text{dlex}}(\textit{conclusion})$ </then> </Implies>

The table rows map through a RIF Document, via a Group and Foralls, down to the level of “:-”-rules. The extended RIF/XML serialization again provides explicit markup, so that the unique inverse χ_{dlex}^{-1} and the semantics-preservation property of χ_{dlex} are easily obtained.

5.2 Rules from RIF/XML to RuleML/XML

The central mapping ξ_{dlex} of section 4.2 from RIF/XML to RuleML/XML is similarly extended for rules.⁴

RIF/XML	RuleML/XML (Fully Striped)
<Document> <payload> <i>group</i> </payload> </Document>	<RuleML> <performative index="1"> <Assert> <formula> $\xi_{\text{dlex}}(\textit{group})$ </formula> </Assert> </performative> </RuleML>
<Group> <sentence> <i>clause₁</i> </sentence> . . . <sentence> <i>clause_n</i> </sentence> </Group>	<Rulebase> <formula> $\xi_{\text{dlex}}(\textit{clause}_1)$ </formula> . . . <formula> $\xi_{\text{dlex}}(\textit{clause}_n)$ </formula> </Rulebase>

continued on next page

⁴ Regarding the last table row, the RuleML 0.91 upgrade to 0.95 adopts the RIF role tags <if> ... <then>, also long contemplated in RuleML, instead of <body> ... <head>.

<i>continued from previous page</i>	
<pre> <Forall> <declare>variable_i</declare> . . . <declare>variable_n</declare> <formula>rule</formula> </Forall> </pre>	<pre> <Forall> <declare>$\xi_{\text{dlex}}(\text{variable}_i)$</declare> . . . <declare>$\xi_{\text{dlex}}(\text{variable}_n)$</declare> <formula>$\xi_{\text{dlex}}(\text{rule})$</formula> </Forall> </pre>
<pre> <Implies> <if>condition</if> <then>conclusion</then> </Implies> </pre>	<pre> <Implies> <if>$\xi_{\text{dlex}}(\text{condition})$</if> <then>$\xi_{\text{dlex}}(\text{conclusion})$</then> </Implies> </pre>

Again, the commonalities between RIF and RuleML prevail, but there are the following differences: The first row maps the RIF root element **Document** to the RuleML root element **RuleML**, which permits (ordered) transactions of performatives including **Asserts**. 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 ξ_{dlex}^{-1} can be obtained, keeping ξ_{dlex} semantics-preserving.

5.3 Rules from RuleML/XML to RuleML/POSL

The mapping π_{dlex} of section 4.3 and the online translators from Stripe-Skipped RuleML/XML to RuleML/POSL are likewise extended for rules.

RuleML/XML (Stripe-Skipped)	RuleML/POSL
<pre> <RuleML> <Assert> group </Assert> </RuleML> </pre>	$\pi_{\text{dlex}}(\text{group})$
<pre> <Rulebase> clause_i . . . clause_n </Rulebase> </pre>	$\pi_{\text{dlex}}(\text{clause}_i)$. . . $\pi_{\text{dlex}}(\text{clause}_n)$
<pre> <Forall> <declare>variable_i</declare> . . . <declare>variable_n</declare> <formula>rule</formula> </Forall> </pre>	$\pi_{\text{dlex}}(\text{rule})$
<pre> <Implies> condition conclusion </Implies> </pre>	$\pi_{\text{dlex}}(\text{conclusion}) \text{ :- } \pi_{\text{dlex}}(\text{condition}).$

Note that the **RuleML**, **Assert**, and **Rulebase** levels are understood in POSL. Similarly, RuleML/XML (optionally) and RuleML/POSL (always) make the same universal closure assumption as Prolog, so the row for **Forall** is rarely needed. Finally, RuleML/POSL “:-”-rules are terminated by Prolog-style periods, unlike those in RIF Presentation Syntax (cf. section 5.1). The generator π_{dlex} has a parser inverse π_{dlex}^{-1} preserving the semantics (<http://www.ruleml.org/posl/converter.jnlp>).

6 Conclusion

The Dlex kernel of the RIF RuleML overlap studied here revealed commonalities and differences between the two languages, enabling round-tripping and convergence. This

positional Dlex can now be expanded for the remaining syntactic features of Datalog RuleML / RIF-Core, including slotted/named arguments, objects/frames, and IRIs as attributes/elements. Since our Dlex mappings already include (interpreted) function symbols for external calls, the study should also be extended to the overlap of Hornlog RuleML and RIF-BLD using (uninterpreted) function symbols for Herbrand terms.

In the RuleML family, Datalog RuleML is complemented by Naf Datalog RuleML, a sublanguage with the often needed Negation as failure, which is also carried through to Naf Hornlog RuleML. Corresponding RIF Non-Monotonic Dialects are called for as well, specializing default negation in the RIF Framework for Logic Dialects [BK09b]. Since Naf constructs are wide-spread in practice, most rule engines have implemented them (including Scoped Naf in cwm [BLCK⁺08]), so their interoperation is already a focus of RIF RuleML efforts (cf. use case: <http://ruleml.org/WellnessRules>).

On the next level, the development of the RIF Production Rule Dialect towards a RIF Reaction Rule Dialect aligned with Reaction RuleML will provide further opportunities for joint work between W3C, OMG, and RuleML, as needed by the industry.

7 Acknowledgements

Many thanks go to Michael Kifer and all colleagues in the W3C RIF Working Group and in the RuleML Technical Groups for continued collaboration on Web rule interchange. NSERC is thanked for its support through a Discovery Grant.

References

- [BHK⁺09] H. Boley, G. Hallmark, M. Kifer, A. Paschke, A. Polleres, and D. Reynolds. RIF Core Dialect. W3C Working Draft. <http://www.w3.org/2005/rules/wiki/Core>, July 2009.
- [BK09a] H. Boley and M. Kifer. RIF Basic Logic Dialect. W3C Working Draft. <http://www.w3.org/2005/rules/wiki/BLD>, July 2009.
- [BK09b] H. Boley and M. Kifer. RIF Framework for Logic Dialects. W3C Working Draft. <http://www.w3.org/2005/rules/wiki/FLD>, July 2009.
- [BLCK⁺08] Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3Logic: A Logical Framework For the World Wide Web. *Theory and Practice of Logic Programming (TPLP)*, 8(3), May 2008.
- [Bol06] Harold Boley. The RuleML Family of Web Rule Languages. In J. J. Alferes, J. Bailey, W. May, and U. Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2006.
- [Bol07] Harold Boley. Are Your Rules Online? Four Web Rule Essentials. In A. Paschke and Y. Biletskiy, editors, *Advances in Rule Interchange and Applications, International Symposium (RuleML-2007), Orlando, Florida*, volume 4824 of *LNCS*, pages 7–24. Springer, 2007.
- [CGT89] S. Ceri, G. Gottlob, and L. Tanca. What You Always Wanted to Know About Datalog (And Never Dared to Ask). *IEEE Trans. on Knowledge and Data Eng.*, 1(1), March 1989.
- [HPSB⁺04] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosz, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission, <http://www.w3.org/Submission/SWRL/>, May 2004.
- [PBK09] A. Polleres, H. Boley, and M. Kifer. RIF Datatypes and Built-ins 1.0. W3C Working Draft. <http://www.w3.org/2005/rules/wiki/DTB>, July 2009.