# Semantic Web Rules for Business Information

Anna Maclachlan
Idilia, Inc.
Montreal, Quebec, Canada

Harold Boley
IIT – e-Business, NRC
Fredericton, New Brunswick, Canada

– Joint Work with Marcel Ball, NRC & UNB Fredericton –

The IASTED International Conference on
WEB TECHNOLOGIES, APPLICATIONS, AND SERVICES (WTAS 2005)
July 4-6, 2005, Calgary, Alberta, Canada

# NBBizKB: New Brunswick Business Knowledge Base

- Static source: “Biznet Directory of Manufacturers and Selected Services to Industry” (Biznet)

# NBBizKB: New Brunswick Business Knowledge Base

- Static source: “Biznet Directory of Manufacturers and Selected Services to Industry” (Biznet)
- Dynamic source: Yahoo! Canada Business Finder (Yahoo!)

# NBBizKB: New Brunswick Business Knowledge Base

- Static source: “Biznet Directory of Manufacturers and Selected Services to Industry” (Biznet)
- Dynamic source: Yahoo! Canada Business Finder (Yahoo!)
- Extract factual data from these sources and integrate them via rules

# NBBizKB: New Brunswick Business Knowledge Base

- Static source: “Biznet Directory of Manufacturers and Selected Services to Industry” (Biznet)
- Dynamic source: Yahoo! Canada Business Finder (Yahoo!)
- Extract factual data from these sources and integrate them via rules
- Use case for data interchange with POSL/RuleML rules

# POSL and RuleML: Knowledge on the Web

- Semantic knowledge representation requires taxonomies and rules

# POSL and RuleML: Knowledge on the Web

- Semantic knowledge representation requires taxonomies and rules
- POSL integrates **po**sitional and **sl**otted knowledge for humans
  (e.g.: Prolog's positional and F-logic's slotted knowledge)

# POSL and RuleML: Knowledge on the Web

- Semantic knowledge representation requires taxonomies and rules
- POSL integrates **po**sitional and **sl**otted knowledge for humans
  (e.g.: Prolog's positional and F-logic's slotted knowledge)
- RuleML marks up such knowledge for machines

# POSL and RuleML: Knowledge on the Web

- Semantic knowledge representation requires taxonomies and rules
- POSL integrates **po**sitional and **sl**otted knowledge for humans
  (e.g.: Prolog's positional and F-logic's slotted knowledge)
- RuleML marks up such knowledge for machines
- POSL $\leftrightarrow$ RuleML translators in OO jDREW and via Java Web Start
  (`http://www.jdrew.org/oojdrew`)

# Yahoo! Data Record as POSL Fact

```
YAHOO_ENT(
  OPER_NAME->"Nackawic Mechanical Ltd";
  PHONE->"(506) 575-2218";
  ...
  CATEGORY->"Mechanical Contractors";
  WEB_SITE->).
```

# Yahoo! Data Record as RuleML Fact

```
<Atom>
  <Rel>YAHOO_ENT</Rel>
  <slot><Ind>OPER_NAME</Ind><Ind>Nackawic Mechanical Ltd</Ind></slot>
  <slot><Ind>PHONE</Ind><Ind>(506) 575-2218</Ind></slot>
  ...
  <slot><Ind>CATEGORY</Ind><Ind>Mechanical Contractors</Ind></slot>
  <slot><Ind>WEB_SITE</Ind><Ind/></slot>
</Atom>
```

# Mapping Rules

Instance-based taxonomy alignment, where instances are RuleML facts

# Mapping Rules

Instance-based taxonomy alignment, where instances are RuleML facts

1. Remove duplicates from each fact base

# Mapping Rules

Instance-based taxonomy alignment, where instances are RuleML facts

1. Remove duplicates from each fact base
2. Find set of enterprises in both fact bases by some identity criterion

# Mapping Rules

Instance-based taxonomy alignment, where instances are RuleML facts

1. Remove duplicates from each fact base
2. Find set of enterprises in both fact bases by some identity criterion
3. For each overlapping pair of enterprise facts derive a candidate sector-category mapping

# Mapping Rules

Instance-based taxonomy alignment, where instances are RuleML facts

1. Remove duplicates from each fact base
2. Find set of enterprises in both fact bases by some identity criterion
3. For each overlapping pair of enterprise facts derive a candidate sector-category mapping
4. A mapping is considered valid if at least *n* enterprise pairs support it

# Enterprise Identity Across Fact Bases

- Phone number generally provided: `BIZNET_ENT`'s `CONTACT/PHONE` slot and `YAHOO_ENT`'s `PHONE` slot

# Enterprise Identity Across Fact Bases

- Phone number generally provided: `BIZNET_ENT`'s `CONTACT/PHONE` slot and `YAHOO_ENT`'s `PHONE` slot
- Requires minimal normalization: Add missing area codes and handle variations in use of, e.g., spaces, hyphens, and parentheses

# Enterprise Identity Across Fact Bases

- Phone number generally provided: `BIZNET_ENT`'s `CONTACT`/`PHONE` slot and `YAHOO_ENT`'s `PHONE` slot
- Requires minimal normalization: Add missing area codes and handle variations in use of, e.g., spaces, hyphens, and parentheses
- Multiple-slot identity over more contact details or enterprise name

# Enterprise Identity Across Fact Bases

- Phone number generally provided: `BIZNET_ENT`'s `CONTACT/PHONE` slot and `YAHOO_ENT`'s `PHONE` slot
- Requires minimal normalization: Add missing area codes and handle variations in use of, e.g., spaces, hyphens, and parentheses
- Multiple-slot identity over more contact details or enterprise name
- Basic overlap between 10,738 Yahoo! and 2,108 Biznet facts: 706 enterprises

# Phone-Identitified Sector-Category Pairs

*In* `IdSectorCategory`*, I, S and C constitute a Phone-identitified Sector-Category pair if there is some* `BIZNET_ENT` *with* `CONTACT/PHONE` *P having* `NAICS_INDUSTRY_SECTOR` *S and some* `YAHOO_ENT` *with* `PHONE` *Q having* `CATEGORY` *C and there is an I such that* `normalphone` *holds for P and I as well as Q and I.*

```
IdSectorCategory(?I, ?S, ?C) :-
  BIZNET_ENT(CONTACT[PHONE->?P ! ?]; NAICS_INDUSTRY_SECTOR->?S ! ?),
  YAHOO_ENT(PHONE->?Q; CATEGORY->?C ! ?),
  normalphone(?P, ?I),
  normalphone(?Q, ?I).
```

# Coping with Many-to-Many Mappings

Mapping analysis showed: **sector** $many-to-many$ **category**

# Coping with Many-to-Many Mappings

Mapping analysis showed: **sector** $many-to-many$ **category**

- Sector 336612 (Boat Building) maps to categories “Boat Builders" and “Canoes Kayaks"

# Coping with Many-to-Many Mappings

Mapping analysis showed: **sector** $many-to-many$ **category**

- Sector 336612 (Boat Building) maps to categories “Boat Builders" and “Canoes Kayaks"
- Category “Draperies & Curtains Retail & Custom Made" maps to sectors 314120 (Curtain and Linen Mills) and 337920 (Blind and Shade Manufacturing)

# Coping with Many-to-Many Mappings

Mapping analysis showed: **sector** $many-to-many$ **category**

- ▷ Sector 336612 (Boat Building) maps to categories “Boat Builders" and “Canoes Kayaks"

- ▷ Category “Draperies & Curtains Retail & Custom Made" maps to sectors 314120 (Curtain and Linen Mills) and 337920 (Blind and Shade Manufacturing)

$\rightsquigarrow$ Sector-category pair found via any identity criterion only accepted if also in 2nd enterprise, with non-identity based again on phones

# 2-Enterprise-Supported Pairs

*Sector S and Category C form a 2-enterprise-supported* `SectorCategory` *pair if* `IdSectorCategory` *holds for a first normalized* `PHONE` *I1 with S and C, and for a second normalized* `PHONE` *I2 with S and C, where I1, I2 are notEqual.*

```
SectorCategory(?S, ?C) :-
  IdSectorCategory(?I1, ?S, ?C),
  IdSectorCategory(?I2, ?S, ?C),
  notEqual(?I1,?I2).
```

Generalizing such 2-enterprise-supported pairs, *n*-enterprise-supported pairs can be introduced

# Mapping Results and Use

- From the 706 overlapping facts, mapping rules discovered 84 2-enterprise-supported sector-category pairs

# Mapping Results and Use

- From the 706 overlapping facts, mapping rules discovered 84 2-enterprise-supported sector-category pairs
- If, for every category, only maximally-supported sector chosen, 27 of the 84 pairs are eliminated, leaving 57 reliable sector-category pairs

# Mapping Results and Use

- From the 706 overlapping facts, mapping rules discovered 84 2-enterprise-supported sector-category pairs
- If, for every category, only maximally-supported sector chosen, 27 of the 84 pairs are eliminated, leaving 57 reliable sector-category pairs
- Some rules on top of `SectorCategory` relation:
  - `uniqueCategory` tests whether `YAHOO_ENT` category from global category list is unique for `BIZNET_ENT` sector

# Mapping Results and Use

- From the 706 overlapping facts, mapping rules discovered 84 2-enterprise-supported sector-category pairs
- If, for every category, only maximally-supported sector chosen, 27 of the 84 pairs are eliminated, leaving 57 reliable sector-category pairs
- Some rules on top of `SectorCategory` relation:
  - `uniqueCategory` tests whether `YAHOO_ENT` category from global category list is unique for `BIZNET_ENT` sector
  - `CSCPath` chains Category-Sector-Category paths connecting categories via intermediate sectors

# Integration Rules

New `NBEnterprise` relation defined through view-like rules integrating the `BIZNET_ENT` and `YAHOO_ENT` facts

- ▷ *Total view* of `NBEnterprise` as `BIZNET_ENT`:

  no information loss

  `NBEnterprise` *assumes all slots unchanged from* `BIZNET_ENT` *facts.*

```
NBEnterprise(! ?All) :- BIZNET_ENT(! ?All).
```

# Partial Views

- *Partial view* of `NBEnterprise` as `YAHOO_ENT`:
  information loss

# Partial Views

- *Partial view* of `NBEnterprise` as `YAHOO_ENT`:
  information loss
- `SectorCategory` rule used to derive sectors from categories
  - Unify slots of `YAHOO_ENT` with variables and employ `SectorCategory` right-to-left in premises
  - Conclusion is `NBEnterprise` relation containing renamed versions of slots – filled with variable values and derived `NAICS_INDUSTRY_SECTOR`

NBEnterprise *assumes all slots from* YAHOO_ENT *facts, using a* SectorCategory *mapping, where the* CONTACT *slots are renamed and restructured as defined in* BIZNET_ENT.

```
NBEnterprise(                                    YAHOO_ENT(
  NAME->?Ne;                                       OPER_NAME->?Ne;
  NAICS_INDUSTRY_SECTOR->?S;                       PHONE->?P;
  CONTACT->                                        FAX->?Fax;
    [PHONE->?P;                                    MAIL_ENG->?MLe;
     FAX->?Fax;                                    MAIL_PLACE->?MCity;
     WEB_SITE->?URL;                               MAIL_PROV->?MProv;
     MAILING_ADDRESS->                             MAIL_POST->?MCode;
       [ADDRESS_LINE1_ENG->?MLe;        :-         CATEGORY->?C;
        CITY->?MCity;                              WEB_SITE->?URL),
        PROVINCE->?MProv;
        POSTAL_CODE->?MCode]])                   SectorCategory(?S, ?C).
```

# Integration Results and Benefit

- From 10,738 `YAHOO_ENT` facts above `NBEnterprise` rule creates `NBEnterprise` facts:
  - 2,700 if categories are mapped to multiple sectors
  - 1,470 if categories are mapped to maximally-supported sector

# Integration Results and Benefit

- From 10,738 `YAHOO_ENT` facts above `NBEnterprise` rule creates `NBEnterprise` facts:
  - 2,700 if categories are mapped to multiple sectors
  - 1,470 if categories are mapped to maximally-supported sector
- Other rules provide further `NBEnterprise` slots such as `COUNTY`

# Integration Results and Benefit

- From 10,738 `YAHOO_ENT` facts above `NBEnterprise` rule creates `NBEnterprise` facts:
  - 2,700 if categories are mapped to multiple sectors
  - 1,470 if categories are mapped to maximally-supported sector
- Other rules provide further `NBEnterprise` slots such as `COUNTY`

$\rightsquigarrow$ Secondary dynamic source plus rules helps to maintain primary static source

# Conclusions

- Presented data interchange use case with RuleML POSL syntax

# Conclusions

- Presented data interchange use case with RuleML POSL syntax
- New Brunswick Business Knowledge Base being maintained at `http://www.ruleml.org/usecases/nbbizkb`

# Conclusions

- Presented data interchange use case with RuleML POSL syntax
- New Brunswick Business Knowledge Base being maintained at `http://www.ruleml.org/usecases/nbbizkb`
- View-based exchange of objected-centred data via OO RuleML rules

# Conclusions

- Presented data interchange use case with RuleML POSL syntax
- New Brunswick Business Knowledge Base being maintained at `http://www.ruleml.org/usecases/nbbizkb`
- View-based exchange of objected-centred data via OO RuleML rules
- Mapping rules essential part of integration rules

# Conclusions

- Presented data interchange use case with RuleML POSL syntax
- New Brunswick Business Knowledge Base being maintained at `http://www.ruleml.org/usecases/nbbizkb`
- View-based exchange of objected-centred data via OO RuleML rules
- Mapping rules essential part of integration rules
- Tradeoff: Transformation (preprocessing) vs. view (querying) rules

# Conclusions

- Presented data interchange use case with RuleML POSL syntax
- New Brunswick Business Knowledge Base being maintained at `http://www.ruleml.org/usecases/nbbizkb`
- View-based exchange of objected-centred data via OO RuleML rules
- Mapping rules essential part of integration rules
- Tradeoff: Transformation (preprocessing) vs. view (querying) rules
- RuleML rules themselves constitute (XML) data for interchange