OUTLINE

➤ Introduction
➤ Examples
➤ Techniques
➤ Summary
INTRODUCTION
EXAMPLE 1 - MULTIPLE STAKEHOLDERS


Guy W. Wallace, CPT  pursuingperformanaceblog.blogspot.com  www.eppic.biz
EXAMPLE 2 – CONFLICTING FEATURES

R20 Reduce user actions

F26 Videos must run automatically

R14 Avoid blocking user interface

F16 Videos must run on request
EXAMPLE 3 – CONFLICTING DEPENDENCIES

R20 Reduce user actions

R14 Avoid blocking user interface

R34 Avoid data leakage

S3 User Profile

F14 Request sign-in
EXAMPLE 4 - CONFLICTING DEPENDENCIES FOR KR

R12  Data Availability

S9   Catalog

F39  Offline navigation

R11  Data Consistency

S98  Item availab.

F37  Update before view
EXAMPLE 5 – CONFLICTING KR

O2 Trusted Domains

Google
LinkedIn
Twitter

OWL: EquivalentClass

O3 Trusted Domains

IEEE
ORCID
CORDIS
EXAMPLE 6 - APPARENT CONFLICTING KR

O2 Trusted Service

O3 Critical Service

O2 Trusted Domains

O2 Trusted Service

O3 Critical Service

O2 Trusted Domains

O3 Trusted Domains

F68 Login

F47 Advertisement

F37 Single sign-on

F45 Banner

OWL: ComplementOf

OWL: allValuesFrom

OWL: EquivalentClass

Google

LinkedIn

Twitter

IEEE

ORCID

CORDIS

OWL: EquivalentClass
EXAMPLE 6 - APPARENT CONFLICT

R1 App is used outdoor

R11 Set best screen visibility

R12 Reduce battery consumption

F17 Increase screen brightness
EXAMPLE 6 – APPARENT CONFLICT: NEW SOLUTION

R1: App is used outdoor

R11: Set best screen visibility

R12: Reduce battery consumption

F19: Use dark mode on the GUI
EXAMPLE 7 – APPARENT CONFLICT

R11 Data Consistency → F17 ACID transactions

R12 Data Availability

R11 Data Consistency → F17 ACID transactions

R12 Data Availability
EXAMPLE 7 – APPARENT CONFLICT: TRADE-OFF

R11 Data Consistency

R12 Data Availability

F19 Buffer transactions
EXAMPLE 8 - APPARENT CONFLICT: TRADE-OFF

R11  Maximise Accuracy

F17  Batch Processing

R12  Minimise Memory Consumption

F19  Stream Processing

F21  Window with frequent items
If requirements and specifications are associated with a **formal semantics** we can use automated reasoning to detect inconsistencies.

Following the **classical principle of non-contradiction**, KR languages consider invalid any set of assertions bringing to a contradictory proposition.

To solve inconsistencies the goal is **isolating the minimal inconsistent subset**.

This operation **grows exponentially** with the number of assertions in the knowledge base.

Greedy methods can be used but human intervention cannot be avoided.
In Hussain et al. an example following the structure below generated 98132 possible resolve candidates

```prolog
# Ontology Org A
o1:UserAccount
    rdfs:subClassOf
        [ a owl:Restriction ;
            owl:maxCardinality 1;
            owl:onProperty o1:recoveryPhone ;
        ]

# Data Org B
o2:UserA o2:recovery "+393287738".
o2:UserA o2:recovery "+393287654".

# Mapping ontologies A and B
"+393287738" owl:differentFrom "+393287654".
o1:recoveryPhone owl:equivalentProperty o2:recovery.
```

Defeasible reasoning is an alternative
- Reasoning does not produce a full or final demonstration
- Its conclusions go beyond the premises
- Integrating common sense and expert knowledge

\[
T_{8.4} = \left\{ \begin{array}{l}
 r_1 : \text{Bird}^* \sqsubseteq \text{Flies}^* \sqcap \text{Animal} \\
 r_2 : \text{Eagle} \sqsubseteq \text{Bird} \\
 r_3 : \text{Penguin}^* \sqsubseteq \text{Bird}^* \sqcap (\neg \text{Fly})^* 
\end{array} \right\}.
\]

\[
\text{Lam}_{\text{Penguin}}(T_{8.4}) = \text{Bird}, \text{Flies}, \text{Penguin}, (\neg \text{Flies}).
\]

Defeasible reasoning is an alternative

Knowledge unification bring to intractable knowledge

Explicit representation of conflicts in knowledge simplify both detection and resolution
The explicit representation of dependencies between features and requirements is often adopted in Software Engineering.

In terms of propositional calculus a conflicting requirement can be interpreted as an alternative denial, the negation of the conjunction

\[ P \rightarrow \neg Q \] equivalent to \( \neg P \lor \neg Q \)

When propositions are correctly gathered detection is easy

To resolve conflicts priority and specificity are typical criteria
**Model of potential conflict and cooperation**

<table>
<thead>
<tr>
<th>Requirement attribute</th>
<th>Functionality</th>
<th>Efficiency</th>
<th>Usability</th>
<th>Reliability</th>
<th>Security</th>
<th>Recoverability</th>
<th>Accuracy</th>
<th>Maintainability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Efficiency</td>
<td>0</td>
<td>+/-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Usability</td>
<td>+</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Reliability</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Security</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Recoverability</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Maintainability</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

EXAMPLE 7 – APPARENT CONFLICT

R11  Data Consistency

R12  Data Availability

F17  ACID transactions

R11  Data Consistency

R12  Data Availability

F17  ACID transactions
EXAMPLE 7 – APPARENT CONFLICT: TRADE-OFF

R11  Data Consistency

R12  Data Availability

F19  Buffer transactions
EXAMPLE 8 - APPARENT CONFLICT: TRADE-OFF

R11  Maximise Accuracy

R12  Minimise Memory Consumption

F17  Batch Processing

F19  Stream Processing

F21  Window with frequent items
D - LESSON LEARNED

➤ Explicit representation of conflicts in knowledge simplify both detection and resolution

➤ The definition of conflicts using a priori knowledge is not always feasible

➤ Partial conflicts requires a detailed understanding of the correlation between feature values
DOMIAN DEPENDENCIES

- Representing the dependency between two domains of values permits to constraints their values
- Different scales can apply
- Different distributions shapes can apply
- Statistical generalisation may not apply
- Non stationary behaviour may apply
Fuzzy membership functions can be used to map a domain to a normalised degree of satisfaction and to put it in relationship with other domains.

Fuzzy membership functions can be used to map a domain to a normalised degree of satisfaction and to put it in relationship with other domains.

Using fuzzy logic the same truth function used with predicate logic can be used with domain dependency:

- \( P \rightarrow \neg Q \) equivalent to \( \neg P \lor \neg Q \)
- \( \neg P \) equivalent to \( 1 - P \)
- \( P \lor Q \) equivalent to \( \max(P, Q) \)
- \( P \rightarrow \neg Q \) equivalent to \( \max(1 - P, 1 - Q) \)

---

Given two variables A and B we can define the dependency between their domains using the following steps:

1. Map domain $D_A$ to a $[0, 1]$ interval using a fuzzy MF
2. Map domain $D_B$ to a $[0, 1]$ interval using a fuzzy MF
3. The truth function $\text{max}(1-P, 1-Q)$ tells us if two values of the valuables are conflicting or not

Example: is $\mu_B$ conflicting with $\mu_A$ given that $\mu_A$ is 0.5 and $\mu_B$ is 0.6?

$$\text{max}(0.5, 0.4) = 0.5 \leq \mu_B$$

A domain is viewed as the interval of values (either discrete or continuous) contained in a lower limit and an upper limit

\[ D: [a, b] \]

A domain is viewed as the interval of values (either discrete or continuous) contained in a lower limit and an upper limit $D: [a, b]$

a) linear and monotonic
$$\mu_A(v \in D) = \frac{a - v}{b - a}.$$  

b) quadratic monotonic
$$\mu_A(v \in D) = \frac{(a - v)^2}{(b - a)^2}.$$  

d) triangular non-monotonic
$$\mu_A(v \in D) = \begin{cases} 0, & v \leq a \\ \frac{v-a}{m-a}, & a \leq v \leq m \\ \frac{b-v}{b-m}, & m \leq v \leq b \\ 1, & v \geq b \end{cases}.$$  

f) gaussian non-monotonic
$$\mu_A(v \in D) = e^{-\frac{v-m}{2k^2}}.$$  

A key point to represent dependencies between two variables’ domains is studying their correlation

- Regression analysis
- Nonparametric regression
- Multiple regression analysis
- Random forest
- Neural networks

EXAMPLE 4 – CONFLICTING DEPENDENCIES FOR KR

R12 Data Availability → S9 Catalog

R11 Data Consistency → S98 Item availab.

S9 Catalog → F39 Offline navigation

S98 Item availab. → F37 Update before view
 Explicit representation of conflicts in knowledge simplify both detection and resolution

The definition of conflicts using a priori knowledge is not alway feasible

Partial conflicts requires a detailed understanding of the correlation between feature values

In dynamic systems conflicts may arise from concurrency
If a resource can be used by mutually exclusive access only, conflicts can be addressed by resource sharing.

Concurrency must be represented and handled.

If a resource can be used by mutually exclusive access only, conflicts can be addressed by resource sharing

Concurrency must be represented and handled

Explicit representation of conflicts in knowledge simplify both detection and resolution

The definition of conflicts using a priori knowledge is not always feasible

Partial conflicts requires a detailed understanding of the correlation between feature values

In dynamic systems conflicts may arise from concurrency

Resources may be consumed after a number of accesses
## IN SUMMARY

<table>
<thead>
<tr>
<th></th>
<th>Knowledge Representation</th>
<th>Dependencies</th>
<th>Dynamic Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detect</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alignent</td>
<td>*</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>Manual Tagging</td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detect Correlation</td>
<td>**</td>
<td></td>
<td>**</td>
</tr>
<tr>
<td>Discovery</td>
<td>**</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Resolve</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partitioning</td>
<td>**</td>
<td></td>
<td>**</td>
</tr>
<tr>
<td>Priorities</td>
<td>**</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Specificity</td>
<td>**</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Domain Mapping</td>
<td>**</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Reachability</td>
<td>**</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td><strong>Predicates</strong></td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>**Dis &amp; Cont **</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td>**</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td><strong>Concurrency</strong></td>
<td></td>
<td></td>
<td>**</td>
</tr>
<tr>
<td><strong>Factual Impossible</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## MOTIVATIONS

<table>
<thead>
<tr>
<th>Knowledge Representation</th>
<th>Dependencies</th>
<th>Dynamic Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alignement</td>
<td>Manual Tagging</td>
<td>Detect Correlation</td>
</tr>
<tr>
<td>More inconsistencies</td>
<td>May implies incompleteness</td>
<td>Data may be incom. Concept Drift applies</td>
</tr>
<tr>
<td>that needed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resolve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Partitioning</td>
<td>Priorities Specificity</td>
<td>Domain Mapping</td>
</tr>
<tr>
<td>Predicates</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Well with priorities</td>
<td>Well with priorities and specificity</td>
<td>Overcomplicated</td>
</tr>
<tr>
<td>Dis &amp; Cont Variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>You have to accept</td>
<td>You have to accept</td>
<td>Well if the mapping is appropriate</td>
</tr>
<tr>
<td>possibility</td>
<td>possibility</td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factual Impossible</td>
<td>Nope</td>
<td>Nope</td>
</tr>
</tbody>
</table>

**MOTIVATIONS**

- **Knowledge Representation**
  - Detect
    - Alignement
    - More inconsistencies that needed
  - Resolve
    - Partitioning
    - Predicates
      - Well with priorities
  - Dis & Cont Variables
    - You have to accept possibility
  - Concurrency
    - Factual Impossible
      - Nope

**Dependencies**

- Detect Correlation
  - May implies incompleteness
  - Data may be incom. Concept Drift applies

**Dynamic Systems**

- Discovery
  - Multiple models
- Reachability
  - Limited to what can be expressed by tokens
- Well
- Nope
Thank You

HANDLING CONFLICTING REQUIREMENTS

Paolo Ceravolo - SESAR Lab - Dipartimento di Informatica
Università degli Studi di Milano