Enhancing Rules with Imperfection
Integration with Soft Computing

Davide Sottara [dsotty@gmail.com]

Department of Computer Science, Electronics and Systems
Faculty of Engineering, University of Bologna
Viale Risorgimento 2, 40100 Bologna (BO) Italy

Bologna/San Diego - April 19-23th, 2010
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
What is Imperfection?

Imperfection, be it Imprecision or Uncertainty, pervades ... systems that attempt to provide an accurate model of the real world

P. Smets, 1999
What is Imperfection?

Imperfection

Imperfection, be it Imprecision or Uncertainty, pervades . . . systems that attempt to provide an accurate model of the real world

P.Smets, 1999

Uncertainty

Uncertainty is a condition where Boolean truth values are unknown, unknowable, or inapplicable . . .

W3C Incubator Group on Uncertainty Reasoning for the Web, 2005
What is Imperfection?

Imperfection - a negative definition

Uncertainty/Imperfection is the opposite of preciseness and certainty, i.e. of what Boolean logic models
An Ontology for Imperfection

- Uncertainty
  - Nature
    - Aleatory
    - Episthemic
  - Derivation
    - Subjective
    - Objective
  - Type
    - Ambiguity
    - Randomness
    - Inconsistency
    - Vagueness
    - Incompleteness
  - Model
    - Probability
    - Belief
    - RandomSets
    - RoughSets
    - FuzzySets

more...
An Ontology for Imperfection

What is Imperfection?

Uncertainty / Confidence Factors

- Nature
  - Aleatory
  - Episthemic
- Derivation
  - Subjective
  - Objective
- Type
  - Ambiguity
  - Randomness
  - Inconsistency
  - Vagueness
  - Incompleteness
- Model
  - Probability
  - Belief
  - RandomSets
  - RoughSets
  - FuzzySets

more...
An Ontology for Imperfection

- Nature
  - Aleatory
  - Episthemic

- Derivation
  - Subjective
  - Objective

- Type
  - Ambiguity
  - Randomness
  - Inconsistency
  - Vagueness
  - Incompleteness

- Model
  - Probability
  - Belief
  - RandomSets
  - RoughSets
  - FuzzySets

Uncertainty / Frequentist Probability

more...
An Ontology for Imperfection

Uncertainty / Bayesian Probability

- Nature
  - Aleatory
  - Episthemic

- Derivation
  - Subjective
  - Objective

- Type
  - Ambiguity
  - Randomness
  - Inconsistency
  - Vagueness
  - Incompleteness

- Model
  - Probability
  - Belief
  - RandomSets
  - RoughSets
  - FuzzySets

more...
An Ontology for Imperfection

What is Imperfection?

- Nature
  - Aleatory
  - Episthemic
- Derivation
  - Subjective
  - Objective
- Type
  - Ambiguity
- Model
  - Probability
  - Belief
  - RandomSets
  - RoughSets
  - FuzzySets

Vagueness / Fuzzy Logic

- Uncertainty
  - Incompleteness
  - Vagueness
  - Inconsistency
  - Randomness
  - Ambiguity

Belief

Probability

more...
Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
- Classification
  - Symptom Matching
  - Fraud Detection
Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
- Classification
  - Symptom Matching
  - Fraud Detection
- Prediction
  - Prognosis
  - Stock Market
What is Imperfection?

Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
- Classification
  - Symptom Matching
  - Fraud Detection
- Prediction
  - Prognosis
  - Stock Market
- Diagnosis
  - Health Care
  - Machine Failure
What is Imperfection?

Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
- Classification
  - Symptom Matching
  - Fraud Detection
- Prediction
  - Prognosis
  - Stock Market
- Diagnosis
  - Health Care
  - Machine Failure
- Monitoring
  - Vital Sign Monitoring
  - Video-Surveillance
Possible Applications

- Information Processing
  - Clinical Procedures
  - Semantic Web
- Classification
  - Symptom Matching
  - Fraud Detection
- Prediction
  - Prognosis
  - Stock Market
- Diagnosis
  - Health Care
  - Machine Failure
- Monitoring
  - Vital Sign Monitoring
  - Video-Surveillance
- ...
Introduction

What is Imperfection?

Why Imperfection?

A generalized inference schema

Modus Ponens

Language and Engine Enhancements

Behind the Scenes

Applications

Logic-Based Approaches

Soft Computing and Hybrid Systems

Integration Patterns

Examples

Conclusions
Why Imperfection?

Using Imperfection

Rules should handle uncertainty, not ignore it

Benefits
- Conciseness
- Robustness

Drawbacks
- Complexity
- Correctness and Coherence
Using Imperfection

Rules should handle uncertainty, not ignore it

**Benefits**
- Conciseness
- Robustness

**Drawbacks**
- Complexity
- Correctness and Coherence

"If you place your bet on an improbable number, and it gets extracted on next round, then expect an increase in your capital"
"If you place your bet on an improbable number, and it gets extracted on next round, then expect an increase in your capital"
Why Imperfection?

Some Issues

If bet(Sum, Number, T) ∧ improbable(Number) ∧ extracted(Number, T+1)
Why Imperfection?

Some Issues

\[
\text{If } \text{bet}(\text{Sum, Number, } T) \\
\land \text{improbable}(\text{Number}) \\
\land \text{extracted}(\text{Number, } T+1)
\]

\[
\text{Then } \\
\text{eval}(\text{bet}(\text{Sum, Number, } T), B) \\
\text{eval}_F(\text{improbable}(\text{Number}), D) \\
\text{eval}_P(\text{extracted}(\text{Number}), P) \\
\text{eval}_\land(B, D, P, \text{Number, } T, X) \\
\text{eval}_\rightarrow(X, \text{Number, } T) \\
\text{print('Gain is', gain(\text{Number, } X))}
\]
Some Issues

If \[ \text{bet}(\text{Sum}, \text{Number}, T) \land \text{improbable}(\text{Number}) \land \text{extracted}(\text{Number}, T+1) \]

Then \[
\begin{align*}
\text{eval}(\text{bet}(\text{Sum}, \text{Number}, T), B) \\
\text{eval}_F(\text{improbable}(\text{Number}), D) \\
\text{eval}_P(\text{extracted}(\text{Number}), P) \\
\text{eval}_\land(B, D, P, X) \\
\text{eval}_\rightarrow(X) \\
\text{print(}'Gain is', \text{gain}((\text{Number}, X)))
\end{align*}
\]

- Truth-functionality
  - Simplifies computation
  - Not always possible (e.g. probability)
Some Issues

If \( \text{bet}(\text{Sum,Number,T}) \land \text{improbable}(\text{Number}) \land \text{extracted}(\text{Number,T+1}) \) Then

\[
\begin{align*}
\text{eval}(\text{bet}(\text{Sum,Number,T}), B) \\
\text{eval}(\text{improbable}(\text{Number}), D) \\
\text{eval}(\text{extracted}(\text{Number}), P) \\
\text{eval}(B, D, P, \text{Number,T,X}) \\
\text{eval}(\rightarrow(X, \text{Number,T})) \\
\text{print}(\text{'Gain is'}, \text{gain}(\text{Number,X}))
\end{align*}
\]

- Truth-functionality
  - Simplifies computation
  - Not always possible (e.g. probability)
- Transparency
  - Automatic computation
  - User should not be aware
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Generalized Inference

\[
\langle P(x), P(X) \rightarrow C(Y) \rangle \\
\hline
C(y)
\]

- Classic Modus Ponens
- **Premise and Implication** entail Consequence

Example

\[ Rich(X) \land Healthy(X) \rightarrow Happy(X) \]
## Generalized Inference

\[
\langle \Phi(\ldots, A_j(x)/\varepsilon_j, \ldots), P(X) \rightarrow C(Y) \rangle \\
\downarrow C(y)
\]

- **Premise**
  - Atomic constraints are **evaluated**
  - General, **pluggable** Evaluators
  - A **Degree** is returned

### Example

\[ Rich(x)_{0.6} \land Healthy(x)_{0.8} \rightarrow Happy(X) \]
Generalized Inference

\[ \langle \Phi(\ldots, A_j(x)/\varepsilon_j, \ldots)/\varepsilon P, P(X) \rightarrow C(Y) \rangle \]
\[ C(y) \]

- **Premise**
  - Atomic constraints are evaluated
  - General, pluggable Evaluators
  - A Degree is returned

- **Premise**
  - Atoms are aggregated in formulas
  - using generalized logic Connectives
  - evaluated by Operators

**Example**

(Rich(x) \land_{0.6} Healthy(x) \rightarrow Happy(X))
Generalized Inference

\[
\langle P(x)/\varepsilon P, \rightarrow (X,Y)/\varepsilon \rightarrow \rangle
\]

\[
\frac{C(y)}{}
\]

- Implication
  - Implication has a Degree
  - often given \textit{a priori}

Example

\[\text{Rich}(x) \land \text{Healthy}(x) \rightarrow_{0.4} \text{Happy}(X)\]
Generalized Inference

\[ \langle P(x)/\varepsilon, \rightarrow (x, y)/\varepsilon \rightarrow \rangle \]

\[ C(y)/\varepsilon C \]

- **Implication**
  - Implication has a Degree
  - often given *a priori*

- **Modus Ponens**
  - MP computes the Degree of the Consequence

**Example**

\( \text{Rich}(x) \land_{0.6} \text{Healthy}(x) \rightarrow_{0.4} \text{Happy}(x)_{0.4} \)
Generalized Inference

\[
\frac{\langle P_1, \rightarrow_1 \rangle}{C_1/\varepsilon C_1}, \ldots, \frac{\langle P_n, \rightarrow_n \rangle}{C_n/\varepsilon C_n}
\]

\[\overbrace{\frac{C(y)}{\varepsilon C}}^\text{C(y)/\varepsilon C}\]

- Merging multiple sources
- Multiple premises for the same conclusion
- Solve conflicts
- Handle missing values

Example

\[\text{Rich}(x) \land \text{Healthy}(x) \rightarrow \text{Happy}(x)^{0.4 \cap 0.2 \cap 0.7}\]
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Language extensions : Example

```java
rule "Rule"
    // custom: implications and MP
    implication @[degree ="0.75"]
    deduction @[kind="min"]
    when
        $o1 : Type( $f1 : field1
            /* custom: external evaluator */
            = = @[id="i1", kind="external", params="..."]
            "val")
    or @[kind="max"] // custom: operators
        $o2 : AnotherType(
            field3 = = 0
            ^^ // custom: operators
            field3 = = @[crisp] $f1 ) //custom: behaviour
    then
        /* consequence degree */
        ... = drools.getConsequenceDegree();
```
Generalized Degrees

Degrees generalize the boolean true/false

- **truth**: compatibility with a prototype
- **probability**: ratio of relevant events over total
- **belief**: opinion in assuming a property to be true.
- **possibility**: disposition towards accepting a situation to be true.
- **confidence**: strength of an agent’s belief in a statement.

Different models, including:

```
  ε  τ  Φ  ≈ ε
```

- Simple
- Interval
- Type-II degrees
Custom Evaluators

\[ \text{Object} \times \text{Object} \rightarrow \text{Degree} \]

\textbf{when} \\
Patient(\text{fever} \sim \text{seems} \text{‘‘high’’})

\textbf{then} \\
\ldots

- Wrap an external function
- Define (and evaluate) a property \( p(L, R) \)
- Return a \textit{Degree}
Custom Operators

\{(Tuple), Degree\}^n \rightarrow Degree

**rule** "Ops"
  **implication**
  **deduction**
when
  $p : Patient( temperature \geq 38 \land \leq 41 )$
  **and**
  exists Medicine( this not \sim allergenic $p$ )
then
  ...

- Aggregate evaluations
- Better if truth-functional
- Return a Degree
- Noteworthy: implication and modus-ponens
Language and Engine Enhancements

Configuration Attributes

Control the behaviour of the engine

- **id**: assign id to constraint/operator
- **kind**: choose evaluator/operator implementation
- **degree**: set “prior” degree
- **params**: additional initialization info
- **crisp**: cast to boolean
- **filter**: configure propagation strategy
- more...
Injection

\[
\text{rule } \text{"Inject"} \\
\text{when } \text{\$p : Patient( temperature \simgeq 38 )} \\
\text{then } \text{inject(\text{\textquotesingle\textquotesingle idFever\textquotesingle\textquotesingle},\$p)} \\
\text{end}
\]

\[
\text{rule } \text{"Injected"} \\
\text{when } \text{\$p Patient( fever \textquotesingle\textquotesingle seems @[id=\text{\textquotesingle\textquotesingle idFever\textquotesingle\textquotesingle] \text{\textquotesingle\textquotesingle high\textquotesingle\textquotesingle}))))} \\
\ldots
\]

Chaining by evaluation: source consequence degree sets the target's

\footnote{Soon to be deprecated in form, but not in concept}
Outline

1 Introduction
   - What is Imperfection?
   - Why Imperfection?

2 A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3 Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4 Conclusions
Refactored Rule Structure

when
  $p : Patient(\text{ age } \sim > 18 )$
  \textbf{implies}
  \texttt{Person( this = $p$, weight } \sim > 50 )$
\textbf{then}$
  ...

Refactored Rule Structure

$H \land \frac{p : P}{\text{age} > 18} \land \frac{H \land P}{\frac{\text{this} = p}{\text{wgt} > 50}}$
Extended RETE

- \( \text{age} > 18 \)
- \( \text{weight} > 50 \)

\( \text{this} \rightarrow 2 \rightarrow 0 \rightarrow 2 \rightarrow \)
Factory controls the coherence

- Builds Degrees
- Builds Operators
- Attributes become params for the factory
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Applications

Pure Logic-Based Approaches
- Symbolic reasoning
- Rules are annotated with degrees
- Computation of facts and degrees according to inference rules

Hybrid Approaches
- Mixed Symbolic/Sub-Symbolic reasoning
- Rule delegate, embed or emulate SC techniques
Outline

1 Introduction
   - What is Imperfection?
   - Why Imperfection?

2 A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3 Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4 Conclusions
Certainty Factors

rule "Mycin"
    implication @ [ degree='0.7' ]
    when
        $s : Site( this ~ sterile )
        Infection( cause ~== 'bacteremia',
                    site == @[crisp] $s )
    ... then
        // Infection is bacteroid
    end

- Simple rule structure
- Evaluators return CF
- Rules have CF themselves
Bayesian Logic Programs

```
rule "BLP"
  //CPT here : p(m|S1,S2)
implication @[ degree='‘...' ' ]
  when
    $s1 : Symptom1( ... )
    $s2 : Symptom2( ... )
  then
    // Illness is ...
end
```

- Conditional probabilities over state of premises
Many-Valued (Fuzzy) Logic Programs

```plaintext
rule "MVL"
  implication @[ kind="Lukas" ]
    when
      Patient( pressure ~seems 'high'
        || @[ kind="max" ]
          temperature ~seems 'high' )
    then
      // ...
  end
```

- Variety of operators (families)
- Full fuzzy set chaining not complete (yet)
Possibilistic Logic Programs

- Degrees given by Necessity/Possibility intervals
- Similar in form to a specific MVL
  - Specific operators
  - Specific semantics
  - Not gradual truth, nor probability!!
- ... but generalizes to fuzzy possibility easily
Hybrid Logic Programs

```plaintext
rule "Hybrid Imperfect"
  // probability
  implication @[ degree="0.99"]
when
  // truth
  true @[ degree="0.5,0.7"] ( Patient( temperature ~seems 'high' ) )
then
  // ...
end
```

- Uncertain/Vague Mix
- Consequence is given a specific probability...
- ... if and only if premise is true to a certain degree
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
     - Soft Computing and Hybrid Systems
       - Integration Patterns
       - Examples

4. Conclusions
Soft Computing

Alternative (?) to Rule-Based Systems

A vast family

Basically, everything that is not (purely) symbolic

- Fuzzy Logic
- Neural Networks
- Genetic Algorithms
- Bayesian Network
- Clustering
Outline

1 Introduction
   - What is Imperfection?
   - Why Imperfection?

2 A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3 Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4 Conclusions
No Integration - External Call

```
rule "No integration"
when
  $s : SCModule(...)
then
  $s.invoke(...);
end
```

- Rules, at best, select the SC module
- SC module is invoked in RHS
- Compatible with boolean logic
Loose Integration - Wrapper

```
rule "Cytofluorimetry"
when
  // Using a neural classifier
  $c : Cell( $f : features ~isA 'red globule')
then
  ...
end
```

- SC module is embedded in a custom evaluator
- SC module must evaluate a predicate
- i.e. the return value must be a *Degree*
- Boolean return value would be a limitation
Strong Integration - Emulation

- SC module is implemented using (imperfect) rules
- Based on Degree manipulation - using operators
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
     - Soft Computing and Hybrid Systems
       - Integration Patterns
       - Examples

4. Conclusions
Induction

rule "Induction"
when
  for any ( $p : Patient( heart ~risk 'high')
    subject_to
      Patient( this == $p,
        weight ~seems 'heavy')
  )
then
  ...

- Generalized quantifier
- Accumulates quantitative degrees
Self-Organizing Map

```
rule "Map Query"
when
  $x : Sample()
  exists Neuron(position ~ close $x )
then
  // (Gradual) Recall...
```

- Rule-based Training algorithm
- Rule-based querying
Self-Organizing Map

Example: 10 neurons in a 2D space:
Feed-Forward Neural Network

- Function-Approximating Networks → Invoke
- Classification Networks → Wrap
- Emulation feasible, under development
Bayesian Network

- Wrappable for use in probabilistic logic
- Emulation is possible (still too verbose)
Outline

1. Introduction
   - What is Imperfection?
   - Why Imperfection?

2. A generalized inference schema
   - Modus Ponens
   - Language and Engine Enhancements
     - Behind the Scenes

3. Applications
   - Logic-Based Approaches
   - Soft Computing and Hybrid Systems
     - Integration Patterns
     - Examples

4. Conclusions
Conclusions

- Uncertainty exists in many forms
Conclusions

- Uncertainty exists in many forms
  → Uncertainty should be embedded in rules
Conclusions

- Uncertainty exists in many forms
  → Uncertainty should be embedded in rules

- Several Imperfect Logics do exist
Conclusions

- Uncertainty exists in many forms
  → Uncertainty should be embedded in rules

- Several Imperfect Logics do exist
- Uncertainty can be handled using other approaches:
  - Bayesian Networks, Neural Networks, Fuzzy Systems, ...
Conclusions

- Uncertainty exists in many forms
  → Uncertainty should be embedded in rules

- Several Imperfect Logics do exist
- Uncertainty can be handled using other approaches:
  - Bayesian Networks, Neural Networks, Fuzzy Systems, ...
- Current Goal: Provide a unified and integrated framework