Overview of Knowledge Formalization with XTT2 Rules

Grzegorz J. Nalepa, Antoni Ligęza, Krzysztof Kaczor

AGH University of Science and Technology, Poland

RuleML2011
19-21 July 2010, Barcelona

`http://geist.agh.edu.pl`

AGH

Outline

Plan: Introduction – the Big Picture

Motivation – some limitations of the existing RBS design approaches

1. *Informal description* – in existing systems rules are informally described and often have no formal semantics.
2. *Sparse knowledge representation* – single rules constitute items of low knowledge processing capabilities.
3. *Blind inference algorithms* – common inference algorithms assume a flat rule base structure, where a brute force (blind) search for a solution is used, whereas number of practical applications are goal-oriented.
4. *Unstructured design approach* – no practical methodology for consecutive top-down design and development of RBS.
5. *Limited quality analysis* – existing solutions to rule design consider system quality analysis only as a step in the system life-cycle not as an integral part of the design.
6. *Lack of integration on the semantic level* – today software engineering practice uses methods and tools that are not conceptually compatible with the rule-based approach.

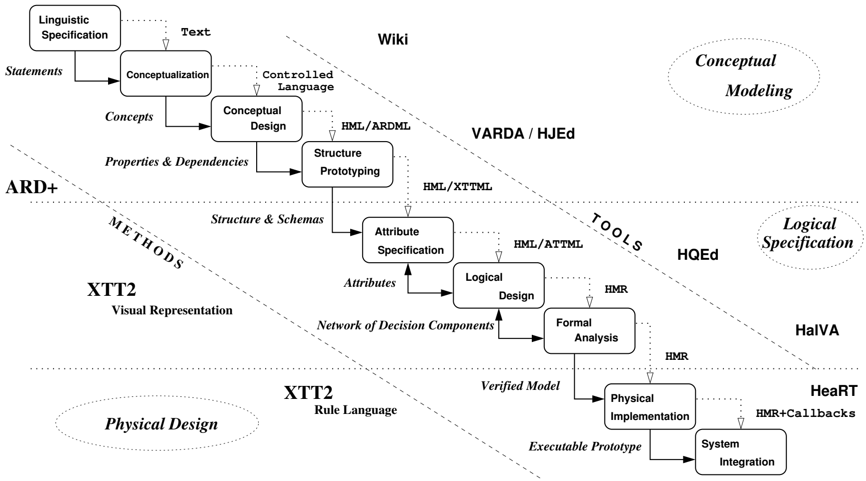
Main solutions

1. *strong formalization of the rule language*: crucial for determining the expressive power, defining the inference mechanism and solving verification issues; formalization uses the ALSV(FD) logic.
2. *custom rule representation formalism*: XTT2, (eXtended Tabular Trees) there are two levels of abstraction in the model.
3. *new inference methods for modularized rule bases*: It is ensured that only the rules necessary for achieving the goal are fired.
4. *systematic design procedure*: a complete, well-founded process that covers all of the main phases of the system life-cycle.
5. *quality analysis solutions* during the design: Formal description in ALSV(FD) allows for the use of formal quality properties tests of the RBS.
6. *practical integration of rule-based approach* with important existing methods to software design. The approach proposes integration solutions for both object-oriented as well as Semantic Web design methods.

Systematic design process

The process includes three main *phases* and several important *stages*.

1. **Conceptual modeling and rule prototyping**: Linguistic Specification, Conceptualization, Conceptual Design and Structure Prototyping,
2. **Logical specification and analysis**: Attribute Specification, Logical Design, and Formal Analysis,
3. **Physical design with prototype integration**: Physical Implementation, and System Integration.


Linguistic Specification
Text
Statements
Conceptualization
Controlled Language
Concepts
Conceptual Design
HML/ARDML
Properties & Dependencies
Structure Prototyping
HML/XTTML
Wiki
VARDA / HJEd
Conceptual Modeling
ARD+
METHODS
TOOLS
Structure & Schemas
Attribute Specification
HML/ATTML
Attributes
Logical Design
HMR
HQEd
Logical Specification
XTT2
Visual Representation
Network of Decision Components
Formal Analysis
HMR
HaIVA
Verified Model
Physical Implementation
HeaRT
HMR+Callbacks
XTT2
Rule Language
Physical Design
Executable Prototype
System Integration

Benefits of rule formalization – XTT2

- *reliable design process* – formalized rule language opens the possibility to partially formalize the design process $\rightarrow$ a better design error detection at early stages, and to simplify the transitions between design stages,
- *clear definition of expressive power* – a strict definition of not only syntax but also semantics of the rule language allows to define the expressiveness of the formulas it is based on,
- *superior control of knowledge base quality* – formal methods can be used to identify logical errors in rule formulation, and
- *knowledge interoperability* – semi-formalized translation to other knowledge representation formats is possible.

Papers' overview

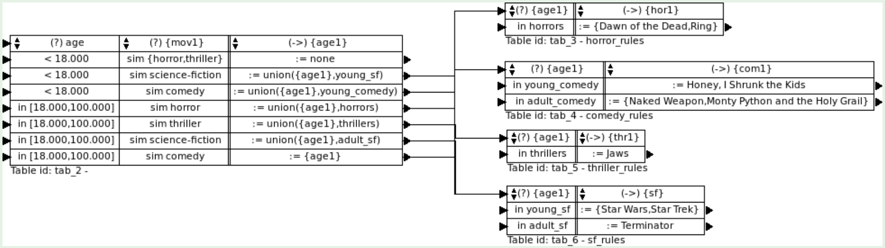
1. This paper:
   - Formalization of single rules with the ALSV(FD) logic.
   - Formalization of modularized rule bases with the XTT2 method.
2. Second Tue paper: Description of custom inference algorithms for XTT2.
3. Wed paper: Discussion of the formalized verification of XTT2.

General Ideas

Rule Formalization in the ALSV(FD) Logic

$$\text{age} < 18 \land \text{movie\_types} \cap \{\text{horror, thriller}\} \neq \emptyset \rightarrow \text{age\_filter} := \emptyset$$

Rule Grouping into Contexts during the Design



(?) age	(?) {mov1}	(->) {age1}
< 18.000	sim {horror,thriller}	:= none
< 18.000	sim science-fiction	:= union({age1},young_sf)
< 18.000	sim comedy	:= union({age1},young_comedy)
in [18.000,100.000]	sim horror	:= union({age1},horrors)
in [18.000,100.000]	sim thriller	:= union({age1},thrillers)
in [18.000,100.000]	sim science-fiction	:= union({age1},adult_sf)
in [18.000,100.000]	sim comedy	:= {age1}

Table id: tab_2 -

(?) {age1}	(->) {hor1}
in horrors	:= {Dawn of the Dead,Ring}

Table id: tab_3 - horror_rules

(?) {age1}	(->) {com1}
in young_comedy	:= Honey, I Shrunk the Kids
in adult_comedy	:= {Naked Weapon,Monty Python and the Holy Grail}

Table id: tab_4 - comedy_rules

(?) {age1}	(->) {thr1}
in thrillers	:= Jaws

Table id: tab_5 - thriller_rules

(?) {age1}	(->) {sf}
in young_sf	:= {Star Wars,Star Trek}
in adult_sf	:= Terminator

Table id: tab_6 - sf_rules

Rule Execution and Verification in HMR

```
[age lt 18, movie_types sim [horror, thriller]] ==> [age_filter set [none]
```

Plan: Formalization of Rules and Rule Bases

ALSV(FD)

- ALSV(FD) (Attributive Logic with Set Values over Finite Domains)
- The formalism is oriented towards Finite Domains (FD) and its expressive power is increased through the introduction of new relational symbols enabling definitions of atomic formulae.
- ALSV(FD) introduces a formal specification of the partitioning of the attribute set needed for practical implementation, and a coherent notation.

Example: attributes

Recommending system

A system recommends books to different groups of persons depending on their age and reading preferences. The age of a reader and his/her preference could be represented by the following attributes:
$\mathbb{A} = \{fav\_genres, age, age\_filter, rec\_book\}$.
In this case we assume that the second attribute is a simple one whereas the others are generalized ones. The fourth one contains book titles that can be recommended to a reader

The attributes have the following domains:
$\mathbb{D} = \mathbb{D}_{fav\_genres} \cup \mathbb{D}_{age} \cup \mathbb{D}_{age\_filter} \cup \mathbb{D}_{rec\_book}$, where:

- $\mathbb{D}_{fav_genres} = \{$horror, handbook, fantasy, science, historical, poetry$\}$,
- $\mathbb{D}_{age} = \{1 \ldots 99\}$,
- $\mathbb{D}_{age_filter} = \{$young_horrors, young_poetry, adult_horrors, adult_poetry$\}$,
- $\mathbb{D}_{rec\_book} = \{$ 'It', 'Logical Foundations for RBS', 'The Call of Cthulhu'$\}$.

Attribute domains

- Let $\mathbb{A}$ denote the set of all attributes used to describe the system.
- Each attribute has a set of admissible values that it takes (a domain).
- Any domain $\mathbb{D}_i$ is assumed to be a *finite*.
- In a general case, a domain can be ordered, partially ordered or unordered (this depends on the specification of an attribute.

Domain

$\mathbb{D}$ is the *set of all possible attributes values*:

$$\mathbb{D} = \mathbb{D}_1 \cup \mathbb{D}_2 \cup \cdots \cup \mathbb{D}_n \tag{1}$$

where $\mathbb{D}_i$ is the domain of attribute $A_i \in \mathbb{A}$, $i = 1 \ldots n$.

Partitioning of the set of attributes

Two types of attributes are identified:

- *simple* ones taking only one value at a time, and
- *generalized* ones taking multiple values at a time.

We introduce the following partitioning of the set of all attributes:

Partitioning

$$\mathbb{A} = \mathbb{A}^s \cup \mathbb{A}^g, \ \mathbb{A}^s \cap \mathbb{A}^g = \emptyset \tag{2}$$

where:

- $\mathbb{A}^s$ is the set of simple attributes, and
- $\mathbb{A}^g$ is the set of generalized attributes.

Simple attribute

A *simple attribute* A_i is a function (or a partial function) of the form:

Simple attribute

$$A_i \colon \mathbb{O} \to \mathbb{D}_i \tag{3}$$

where:

- $\mathbb{O}$ is a set of objects,
- $\mathbb{D}_i$ is the domain of attribute A_i.

Generalized attribute

The definition of *generalized attribute* is as follows:

Generalized attribute

$$A_i \colon \mathbb{O} \to 2^{\mathbb{D}_i} \tag{4}$$

where:

- $\mathbb{O}$ is a set of objects,
- $2^{\mathbb{D}_i}$ is the set of all possible subsets of the domain $\mathbb{D}_i$.

State representation

- The current values of all attributes are specified within the contents of the knowledge base.
- From logical point of view the *state* of the system is represented as a logical formula of the form:

State

$$s\colon (A_1 = S_1) \wedge (A_2 = S_2) \wedge \ldots \wedge (A_n = S_n) \tag{5}$$

where A_i are the attributes and S_i are their current values.

- Note that $S_i \in \mathbb{D}_i$ for simple attributes and $S_i \subseteq \mathbb{D}_i$ for generalized ones.
- An explicit notation for covering unspecified, unknown values is proposed: $A_i = \mathit{null}$ means that the value of A_i is unspecified.

Example: state

Following the example, an exemplary state can be defined as:

$$(age = 16) \wedge (fav\_genres = \{\text{horror, fantasy}\})$$

This means that a given person is 16 years old and she or he likes reading horror and fantasy books.
It is a partial state where only the values of the input attributes are defined. In this example it will be sufficient to start the inference process. To specify the full state, the values of the remaining attributes should be defined as *null*.

Attribute Classes

Attribute Classes

$$\mathbb{A}^s = \mathbb{A}^s_{in} \cup \mathbb{A}^s_{int} \cup \mathbb{A}^s_{out} \cup \mathbb{A}^s_{io} \quad (6)$$

$$\mathbb{A}^g = \mathbb{A}^g_{in} \cup \mathbb{A}^g_{int} \cup \mathbb{A}^g_{out} \cup \mathbb{A}^g_{io} \quad (7)$$

where all these sets are pairwise disjoint:

- $\mathbb{A}^s_{in}, \mathbb{A}^g_{in}$ are the sets of *input* attributes,
- $\mathbb{A}^s_{int}, \mathbb{A}^g_{int}$ are the sets of *internal* attributes,
- $\mathbb{A}^s_{out}, \mathbb{A}^g_{out}$ are the sets of *output* attributes, and
- $\mathbb{A}^s_{io}, \mathbb{A}^g_{io}$ are the sets of attributes that can be simultaneously input and output (*communication* attributes).

These *attribute classes* (i.e. input, internal, output, and communication) are used in the rule specification to support the interaction of the system with its environment.
They are handled by dedicated *callbacks*. Callbacks are procedures providing means for reading and writing attribute values.

Example: atomic formulae

- In the example the following atomic formulae could be present: $age \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}$
- which could also be denoted as $age < 18$ and
- $fav\_genres \subseteq \{\text{science, fantasy, horror}\}$.
- The interpretation of the second one is: the person likes a subset of *science*, *fantasy*, *horror* books.

Simple attribute formulae syntax

Syntax	Meaning	Relation
$A_i = d_i$	the value of A_i is precisely defined as d_i	`eq`
$A_i \in V_i$	the current value of A_i belongs to V_i	`in`
$A_i \neq d_i$	shorthand for $A_i \in (\mathbb{D}_i \setminus \{d_i\})$	`neq`
$A_i \notin V_i$	shorthand for $A_i \in (\mathbb{D}_i \setminus V_i)$	`notin`

Generalized attribute formulae syntax

Syntax	Meaning	Relation
$A_i = V_i$	A_i equals to V_i (and nothing more)	`eq`
$A_i \neq V_i$	A_i is different from V_i (at least one element)	`neq`
$A_i \subseteq V_i$	A_i is a subset of V_i	`subset`
$A_i \supseteq V_i$	A_i is a superset of V_i	`supset`
$A_i \sim V_i$	A_i has a non-empty intersection with V_i	`sim`
$A_i \not\sim V_i$	A_i has an empty intersection with V_i	`notsim`

Non-trivial conditions

- ALSV(FD) allows to build non-trivial conditions of rules.
- Formulae *may not be logically independent*.
- For example imagine two formulae: the first: $age \in \{1, 2, 3, 4\}$ and the second one: $age \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.
- If these formula are use in conditions of two different rules, the first rule is not needed, and such can be removed from the knowledge base.
- This is because the second condition is more general $\rightarrow$*rule subsumption*.
- In order to be able to analyze such a case of logical relation between rules, as well as other cases, the inference rules for the atomic formulae are considered next.

Inference rules for atomic formulae, simple attributes

$\models$	$A_i = d_j$	$A_i \neq d_j$	$A_i \in V_j$	$A_i \notin V_j$
$A_i = d_i$	$d_i = d_j$	$d_i \neq d_j$	$d_i \in V_j$	$d_i \notin V_j$
$A_i \neq d_i$	—	$d_i = d_j$	$V_j = \mathbb{D}_i \setminus \{d_i\}$	$V_j = \{d_i\}$
$A_i \in V_i$	$V_i = \{d_j\}$	$d_j \notin V_i$	$V_i \subseteq V_j$	$V_i \not\sim V_j$
$A_i \notin V_i$	$\mathbb{D}_i \setminus V_i = \{d_j\}$	$V_i = \{d_j\}$	$V_j = \mathbb{D}_i \setminus V_i$	$V_j \subseteq V_i$

See related works for inference rules for atomic formulae generalized attributes...

Relational operators in rules I

The set of all relational *operators* that can be used in rules is defined as follows:

$$\mathbb{F} = \mathbb{F}^{cond} \cup \mathbb{F}^{dec} \tag{8}$$

- $\mathbb{F}^{cond}$ – set of operators that can be used in the conditional part of a rule.

$$\mathbb{F}^{cond} = \mathbb{F}_a^{cond} \cup \mathbb{F}_s^{cond} \cup \mathbb{F}_g^{cond} \tag{9}$$

  where:

  - $\mathbb{F}_a^{cond}$ contains operators, that can be used in the rule conditional part with all attributes. The set is defined as:

$$\mathbb{F}_a^{cond} = \{=, \neq\} \tag{10}$$

Relational operators in rules II

- $\mathbb{F}_s^{cond}$ is the set that contains the operators, which can be used in a rule conditional part with simple attributes. The set is defined as:

$$\mathbb{F}_s^{cond} = \{\in, \notin\} \tag{11}$$

  Operators $<, >, \leq, \geq$ provide only a variation for $\in$ operator, to be used only with ordered domains.

- $\mathbb{F}_g^{cond}$ contains operators that can also be used in the rule conditional part with generalized attributes. The set is defined as:

$$\mathbb{F}_g^{cond} = \{\subseteq, \supseteq, \sim, \nsim\} \tag{12}$$

- $\mathbb{F}^{dec}$ is a set of all operators that can be used in a rule decision part.

$$\mathbb{F}^{dec} = \{:=\} \tag{13}$$

The operator $:=$ allows for assigning a new value to an attribute

ALSV(FD) Triples

Let us consider the set $\mathbb{E}$ that contains all the triples that are legal atomic formulae in ALSV(FD). The triples are built using the operators:

Triples

$$\begin{aligned}\mathbb{E} = &\{(A_i, \propto, d_i), A_i \in \mathbb{A}^s, \propto \in (\mathbb{F} \setminus \mathbb{F}_g^{cond}), d_i \in \mathbb{D}_i\} \cup \\ &\{(A_i, \propto, V_i), A_i \in \mathbb{A}^g, \propto \in (\mathbb{F} \setminus \mathbb{F}_s^{cond}), V_i \in 2^{\mathbb{D}_i}\}\end{aligned} \tag{14}$$

The ALSV(FD) triples are the basic components of rules.

XTT2 Rule

Rule

$$r = (\mathbb{COND}, \mathbb{DEC}, \mathbb{ACT}) \tag{15}$$

where:

- $\mathbb{COND} \subseteq \mathbb{E}$,
- $\mathbb{DEC} \subseteq \mathbb{E}$, and
- $\mathbb{ACT}$ is a set of actions to be executed when a rule is fired.

Operational notation

The rule defined by Formula 15 can also be presented in the following form:

$$r\colon [\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_k] \to [\theta_1 \wedge \theta_2 \wedge \cdots \wedge \theta_l], DO(\mathbb{ACT}) \tag{16}$$

where:

- $\{1, \ldots, k\}$ and $\{1, \ldots, l\}$ are the sets of identifiers, $k \in \mathbb{N}$, $l \in \mathbb{N}$
- $\phi_1, \ldots, \phi_k \in \mathbb{COND}$, and
- $\theta_1, \ldots, \theta_l \in \mathbb{DEC}$.

Example: rules

Consider example of the following rules:

$r_1 : [age < 18 \wedge fav\_genres \supseteq \{\text{horror}\}] \longrightarrow [age\_filter := \text{young\_horrors}]$

$r_2 : [age = \_ \wedge fav\_genres \subseteq \{\text{science}\}] \longrightarrow [age\_filter := \text{all\_science}]$

$r_3 : [age\_filter \in \{\text{young\_horrors, adult\_horrors}\}] \longrightarrow [rec\_book := \text{'It'}]$

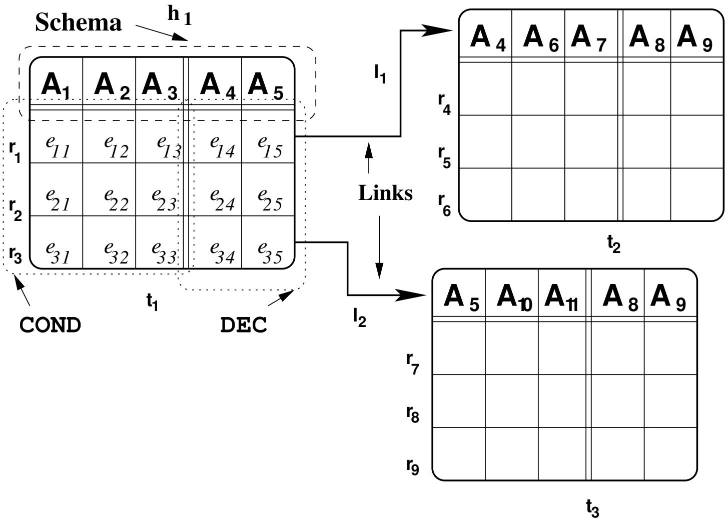
Now, having the previously defined state:
$(age = 16) \wedge (fav\_genres = \{horror, fantasy\})$,
it can be observed, that rules r_1 and r_3 could be *fired*.

Rule firing

Firing a single XTT2 rule r involves the following basic steps:

1. Checking if *all* the ALSV(FD) triples in the $\mathbb{COND}$ part are satisfied.
2. If so, changing the system state by evaluating triples (assigning new values to attributes) in the $\mathbb{DEC}$ part.
3. Executing actions defined by $\mathbb{ACT}$; actions do not change attribute values.

An example of an XTT2 knowledge base



Rule schema definition

Rule schema

$$\begin{aligned} \forall r &= (\mathbb{COND}, \mathbb{DEC}, \mathbb{ACT}) \\ h(r) &= (\mathrm{trunc}(\mathbb{COND}), \mathrm{trunc}(\mathbb{DEC})) \end{aligned} \tag{17}$$

Therefore, each rule has a schema that is a pair of attribute sets:

$$h = (H^{cond}, H^{dec}) \tag{18}$$

H^{cond} and H^{dec} sets define the attributes occuring in the conditional and decision part of the rule; $H^{cond} = \mathrm{trunc}(\mathbb{COND})$ and $H^{dec} = \mathrm{trunc}(\mathbb{DEC})$. The *trunc* function which transforms the set of atomic formulae into a set of attributes that are used in these triples.

A schema is used to identify rules working in the same operational context.

Decision component (Table)

It is an ordered set (sequence) of rules having the same rule schema:

Table

$$t = (r_1, r_2, \ldots, r_n) \text{ such that } \forall_{i,j}\colon r_i, r_j \in t \rightarrow h(r_i) = h(r_j) \tag{19}$$

where $h(r_i)$ is the schema of the rule r_i.

Inference link

Inference link

An *inference link* l is an ordered pair:

$$l = (r, t),\ l \in \mathbb{R} \times \mathbb{T} \tag{20}$$

where:

- $\mathbb{R}$ is the set of rules in the knowledge base, and
- $\mathbb{T}$ is the set of tables in the knowledge base.

Components (tables) are connected (linked) in order to provide inference control.
A single link connects a single rule (a row in a table) with another table.
A structure composed of linked components is called a XTT2 knowledge base.

XTT2 Knowledge Base

The XTT2 *knowledge base* is the set of components connected with links.

XTT2 Knowledge Base

$$X = (\mathbb{T}, \mathbb{L}) \tag{21}$$

where:

- $\mathbb{T}$ is a set of components (tables),
- $\mathbb{L}$ is a set of links, and
- all the links from $\mathbb{L}$ connect rules from $\mathbb{R}$ with tables from $\mathbb{T}$.

Links are introduced during the design process according to the specification provided by the designer.
The knowledge base can be perceived as an *inference network*.

XTT2: observations

- A number of specific structures of knowledge bases could be considered including decision trees.
- In such a decision tree-like structure nodes would consist of single decision components.
- So, the XTT2 knowledge base can be seen as a generalization of classic decision trees and tables
- In fact, it can also be a generalization of structures describing other inference processes such as business processes described using BPMN (Business Process Model and Notation), or selected UML diagrams.

Benefits

1. well-defined design and rule translation
2. different inference options (RuleML2011 Tue paper)
3. rule base verification on early design stages (RuleML2011 Wed paper)

Plan: Summary

XTT2: summary

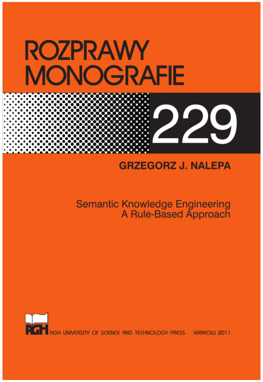
- Here a logic-based formalism for decision rules was considered.
- The formalism allows for building modularized knowledge bases.
- The XTT2 knowledge base is highly modularized and hence XTT2 tables are not only a mechanism for managing large knowledge bases, but they also allow for context-aware reasoning.
- In practice, building such a modularized knowledge base is not a straightforward taskldots
- Practical inference in such a system cannot use the classic inference algorithms. This is why custom inference modes are introduced.

Applications

- knowledge management
- Business Rules
- soft-real time control of mobile robots
- Software Engineering
- Semantic Web

If you want to know more. . .

- See our two other presentations on RuleML2011
- Take a copy of the book (outside the room):



ROZPRAWY MONOGRAFIE 229

GRZEGORZ J. NALEPA

Semantic Knowledge Engineering
A Rule-Based Approach

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY PRESS KRAKOW 2011

The End

¡Thank you for your attention!

¿Any questions?

Web Page: `http://geist.agh.edu.pl`