Domain knowledge as Ontology-based Event-B Theories [1]

Ismail Mendil [1], Neeraj Kumar Singh[1], Yamine Aït Ameur[1], Dominique Méry[2] and Philippe Palanque[3]

[1]INPT-ENSEEIHT / IRIT Université de Toulouse, France
[2]LORIA, Telecom Nancy Université de Lorraine, France
[3]IRIT, Université de Toulouse, Toulouse, France

June 8, 2021

ABZ'21: 9th Rodin User and Developer Workshop

[1]This work is supported by Formedicis and EBRP French ANR projects

Introduction

Objective

Use of Event-B theories for formalising domain knowledge as an ontology for designing interactive systems.

Challenges

- Formal methods lack domain-knowledge integration.
- Examples of properties: *critical aircraft shall be visible, the selection of widgets in a radio box shall be exclusive*

Motivation

- Domain-knowledge integration $\Rightarrow$ explicitation of application context.
- Separating general common knowledge from system-specific requirements.
- Greater attention on mathematical theories: sequences, differential equations, etc.
- Lack of domain-specific theories.

There is a need for a general integrated framework for expressing and transferring domain knowledge into formal design models.

Ontology as a notation for formalising domain knowledge

Ontology is a formal specification of a shared conceptualisation[2]*.*

Many ontology modelling languages (OML)

- *Classes*
- *Properties*
- *Types*
- *Instances*
- *Constraints*

[2] A translation approach to portable ontology specifications, Thomas R.Gruber 1993

OML - DATATYPE

- **Ontology(C, P, I)** is a generic data type
- Providing: **classes**, **properties**, **instances**.
- Specifying **classProperties**, **classInstances**.
- Constrained instantiation: `instancePropertyValues` & `isWDInstancePropertyValues`.

DATATYPES **Ontology(C, P, I)**
CONSTRUCTORS
consOntology(*classes*: $\mathbb{P}(C)$,
properties: $\mathbb{P}(P)$,
instances: $\mathbb{P}(I)$,
classesProperties: $\mathbb{P}(C \times P)$,
classesInstances: $\mathbb{P}(C \times I)$,
classesAssociations: $\mathbb{P}(C \times P \times C)$,)
instancePropertyValues: $\mathbb{P}(I \times P \times I)$)

OML - a glimpse of OPERATORS

- Accessors, updating operators, predicate operators for well-definedness conditions
- **instancePropertyValues** shall be conform to **classAssociations**
- isWDOntology formalises all the conditions for a well-defined ontology.
- isA defines the subsumption relationship.

```
OPERATROS
  isWDGetInstancePropertyValues <predicate> (o: Ontology(C, P, I))
  well-definedness isWDClassProperites(o) ∧ isWDClassInstances(o) ∧ isWDClassAssociations(o)
  direct definition
   instanceAssociations(o) ⊆ { i1 ↦ p ↦ i2 | i1 ∈ I ∧ p ∈ P ∧ i2 ∈ I ∧ i1 ↦ p ↦ i2 ∈ instances(o) × properties(o) × instances(o) ∧
    ( ∃c1, c2 · {c1, c2} ⊆ getClasses(o) ⇒ ( c1 ↦ p ↦ c2 ∈ getClassAssociations(o) ∧ p ∈ getClassProperties(o)[c1] ∧
                                       i1 ∈ getClassInstances(o)[{c1}] ∧ i2 ∈ getClassInstances(o)[{c2}])) }
  getInstancePropertyValues <predicate> (o: Ontology(C, P, I))
  well-definedness isWDGetInstancePropertyValues
                                    ...
  isWDOntology <predicate> (o: Ontology(C, P, I))
                                    ...
  isA <predicate> (o: Ontology(C, P, I),c1: C,c2: C)
  well-definedness isWDClassProperites(o) ∧ ontologyContainsClasses(o, {c1, c2})
  direct definition
   getInstancesOfaClass(o, c1) ⊆ getInstancesOfaClass(o, c2)
  ...
```

OML - an extract of THEOREMS

- Generic theorems for reusability in proof process.
- Theorems are instrumental for *once and for all* proving paradigm.

THEOREMS

isATransitivity: ∀ o, c1, c2, c3·o∈Ontology(C,P,I) ∧ isWDOntology(o) ∧ c1 ∈ C ∧ c2 ∈ C ∧ c3 ∈ C ∧
ontologyContainsClasses(o, {c1, c2, c3}) ⇒ (isA(o, c1, c2) ∧ isA(o, c2, c3) ⇒ isA(o, c1, c3))

containsClassCompatibleWithUnion:
∀o, cs1, cs2 · o ∈ Ontology(C, P, I) ∧ isWDOntology(o) ∧ cs1 ⊆ C ∧ cs2 ⊆ C ∧ cs1 ≠ ∅ ∧ cs2 ≠ ∅ ∧
ontologyContainsClasses(o, cs1) ∧ ontologyContainsClasses(o, cs2) ⇒ (ontologyContainsClasses(o, cs1 ∪ cs2))

ARINC 661 specification standard

- ARINC 661 [3] is standard for Cockpit Display System (CDS) of aircraft.
- **The document is structured but written in natural language (759p + 265p).**
- ARINC 661 defines a widget library ($\sim$ 65 widgets).
- Some widgets: PicturePushButtons, RadioButtons and EditBoxNumeric

[3] ARINC 661 661 specification: Cockpit display system interfaces to user systems (June 2019)

ARINC 661 Event-B theory description

- Ontology-based conceptualisation of ARINC 661 standard as an Event-B theory.
- `Ontology(C, P, I)` data type instantiation.
  - C $\longrightarrow$ `ARINC661Classes`
  - P $\longrightarrow$ `ARINC661Properties`
  - I $\longrightarrow$ `ARINC661Instances`
- Additional operators built upon `OntologiesTheory` operators.
- ARINC 661 requirements embedded in the operators and proved as theorems.

ARINC661Theory - ontology-based definition

AXIOMATIC DEFINITIONS
ARINC661Axiomatisation
TYPES ARINC661Classes, ARINC66Properties, ARINC661Instances
OPERATORS

- **Label** <expression> ARINC661Classes
- **RadioBox** <expression> ARINC661Classes
- **CheckButton** <expression> ARINC661Classes
- **PushButton** <expression> ARINC661Classes
- **ToggleButton** <expression> ARINC661Classes
- **EditBoxNumeric** <expression> ARINC661Classes

consARINC661Ontology <expression> *(ii: $\mathbb{P}(I)$, cii: $\mathbb{P}(C \times I)$, ipvs: $\mathbb{P}(I \times P \times I)$)*

well-definedness isWDOntology(o)

AXIOMS

ARINC661ClassesDef

partition(ARINC661Classes, . . . , {ARINC661_BOOL}, {Label}, {RadioBox}, {CheckButton}, {PushButton}, {ToggleButton}, {EditBoxNumeric}, . . .)

consARINC661OntologyDef

. . . ⇒ consARINC661Ontology(ii, cii, ipvs) = consOntology(ARINC661Classes, ARINC66Properties, ii, wellBuiltClassProperties, wellbuiltTypesElements ∪ cii, wellBuiltClassAssociations, ipvs), . . .

- Ontology-based definition of ARINC661Theory.
- Axiomatic definitions for all the definitions.
- Theorems for theory validation.

Theory	# Operators	# Axioms
OntologiesTheories	37	0
ARINC661Theory	55	15

Table: Theories statistics

Conclusion & perspectives

- Formalising domain knowledge as ontology in Event-B as theories.
- Exploiting generic typing and well-definedness conditions.
- Annotation of Event-B events and expressing properties like:

Every user **input**
must be followed by
a user **confirmation** *interaction*.

- Bug identification and correction available in Theory Plug-in release 4.0.2

Event-B component	# PO	Proof process
OntologiesTheories	21	Interactive
ARINC661Theory	09	Interactive

Table: Proof-obligation statistics table

Thank you for your attention.
Any questions?