AAI-ANALYSIS. USED FORMALISMS
Version V2 (still not complete)

eJournal: uffmm.org, ISSN 2567-6458
2.May 2018
Email: info@uffmm.org

AUTHOR: Gerd Doeben-Henisch
EMAIL: gerd@doeben-henisch.de

Contents

Abstract

This is a short introduction into those formalisms, which will be used during the AAI-analysis as described in the AAI-Theory-Micro-Edition. This version differs strongly from preceding versions.

1 Introduction

During the AAI-analysis an 'actor story (AS)' has to be composed by the AAI-experts. An actor story assumes that a task can be understood as a series of 'situations' also called 'states. A state is assumed to represent a finite 'collection of facts'. The 'transition' from one state to another is characterized by 'facts which change'. There can be different kinds of changes which represent different options which lead to 'different successor states'.

For such an actor story at least the following 'modes of representations' are assumed:

1. The base version is represented as a 'text' D_0 written in an ordinary language named L_0.

2. One of the extended versions is a 'formalized version' D_{st} of the base version written in a 'formal language' named L_{st}. It is assumed that there exists a 'translation function' $\tau_{0,st} : D_0 \longmapsto D_{st}$.

3. A second extended version is a 'pictorial version' D_{pict} of the base version written in a 'pictorial language' named L_{pict}. It is assumed that there exists two more 'translation functions' $\tau_{0,pict} : D_0 \longmapsto D_{pict}$ as well as $\tau_{st,pict} : D_{st} \longleftrightarrow D_{pict}$

4. A third extended version is a '(mathematical) graph version' D_γ of the formalized version written in a 'graph language' named L_γ. It is assumed that there exists a 'translation function' $\tau_{st,\gamma} : D_{st} \longleftrightarrow D_\gamma$

5. A fourth extended version is a 'simulated version' D_{sim} of the base version written in a formal language L_ϵ which describes an 'automaton' named α. This automaton can 'simulate' the actor story in each of the preceding modes as a sequence of states (connected in the graph), as a sequence of pictures (following the order given by the graph), and each simulation mode is always commented by the text from the textual mode.

2 Domain of Reference D^R

If an AAI expert starts the construction of a representation for an actor story it has to be assumed that there exist a 'domain of reference' D^R either as a 'real situation' or as a 'mental model', which represents all the facts which shall be analyzed.

The basic elements of this domain of reference are as follows:

1. There are 'objects' associated with 'attributes', at least one. There can be attributes which are associated with more than one object.
2. An attribute associated with only one object is also called a 'property'.
3. An attribute associated with more than one object is called a 'relation'.
4. Attributes associated with objects constitute 'facts'.
5. A whole situation or state is called a 'collection of facts'.
6. A 'transition' from one state s to another state s' is characterized by a 'change' of at least one fact, i.e. an existing 'fact' can either be 'deleted' in the successor state or a not yet existing fact can newly be 'created' in the successor state.
7. With regard to a state 'more than one change' can occur alternatively which is manifested by different possible successor states.

3 Formal Representation

To be able to represent the content of the domain of reference D^R in formal expressions one needs a formal language L_{st} which fits to D^R. The introduction of these formal expressions happens stepwise following the 'ingredients' of the domain of reference. Implicitly this gives an 'interpretation' ι' too from elements of D^R to formal expressions from L_{st}.

1. The 'objects' of D^R will be represented by 'Names' in L_{st}, realized by lower-case letters.
2. The 'attributes' of D^R will be represented by 'Predicates' in L_{st}, realized by upper-case letters.
3. A predicate expression in L_{st} associated with only one name in L_{st} is also called a 'property'.
4. A predicate expression in L_{st} associated with more than one name in L_{st} is also called a 'relation'.
5. Predicates associated with names constitute 'atomic expressions' in L_{st} and atomic expressions can represent 'facts' in D^R. An atomic expression can occur in a special 'is-equal-relation' written as '=' equating a fact-name with a fact, written as 'Fname = fact'. The elements on the left or on the right of an 'equation-relation' can be substituted for each other.
6. A 'collection of facts' in D^R can be represented in L_{st} as a set of atomic expressions. A set of atomic expression can also occur in a special 'is-equal-relation' written as '=' equating a Set-name with a set of facts, written as '$Sname = \{f_1, ..., f_n\}$'. The elements on the left or on the right of an 'equation-relation' can be substituted for each other.
7. A 'transition' from one state s to another state s' can be represented by a 'change expression' ϵ in L_{st} defined by $\epsilon \subseteq Sname \times Sname \times Fname \times 2^{Fname} \times 2^{Fname}$. A change expression indicates first the given state, then an successor state, then a fact name which represents an action causing a change, then a set of fact-names indicating which facts have to be deleted, and finally a set of fact-names indicating which facts have to be created.
8. Because in the domain 'more than one change' can occur alternatively it is possible to represent more than one change expression, but given as a fixed sequence $\epsilon = \epsilon_1 \otimes ... \otimes \epsilon_n$ every ϵ_i represents then a change from a state s to a state s'.

4 Example

Thus if one has in D^R an object which is an 'actor' with name 'a' behaving as a 'user' one could represent this attribute with the expression 'USER(a)' of L_{st}. If there exists another object with name 'i' in D^R serving as an 'interface' for the user one can represent this property with the expression 'INTERFACE(i)' of L_{st}. And if one wants to describe in D^R the relation, that the user 'a' can 'see' the interface object 'i', one can represent this with the expression 'SEES(a,i)' in L_{st}. A state 's' in D^R can then be written with the expression '$s = \{USER(a), INTERFACE(i), SEES(a, i)\}$' of L_{st}

5 Self-Incrementing Name-Spaces

If someone starts to formalize a domain of reference D^R with expressions from the language L_{st} one can think of the set of 'Names' as well as of the set of 'Predicates' as an initially 'empty set'. Every name which during the process of formalization will be introduced will be stored in the set of 'Names' or 'Predicates'. Thereby the set of 'Names' and 'Predicates' is incrementing. If the actor story has been finished the sets of 'Names' and 'Predicates' will be kept fixed.

6 Makros

... how one can replace collections or parts of collections by names and thereby compress graphs for a better understanding ...

7 A Graph

... describing what a graph is ...

8 A Fact-Graph

... and how one can extend the definition with facts and changes.

9 The Simulation of a Graph

...

10 A Pictorial Story

...