Describing Knowledge Transformations

AKT Working Document

Stephen Potter 5 April 2002

1 Introduction

This note discusses some initial thoughts about how knowledge transformations may be described at some general level of abstraction. The immediate purpose of generating such descriptions would be to provide a brokering language for expressing, on one hand, the competences of knowledge-transforming agents and, on the other, for formulating queries requesting particular transformation services of those agents. (These initial thoughts draw quite heavily upon the descriptions introduced by Compatangelo and Sleeman (2001) in their draft paper *K-ShaRe: a vision of sharing and reuse in AKT*.)

2 Knowledge Transformation

A knowledge transformation might be considered to be any process that alters the content or form of some *body of knowledge* so as to alter the potential inferences that result from the application of that knowledge, the contexts in which it may be utilised, or else the effectiveness or efficiency of its application. (As is conventional in AI, knowledge is here considered to be actionable information.)

Attempting to formalise this definition, knowledge transformation is here characterised as a mapping from one or more input *bodies of knowledge*, K_i to a single output body of knowledge, K_O:

$$K_1 \times K_2 \times \ldots \times K_n \rightarrow K_O$$

Hence, to describe this transformation, we need to describe the input and output bodies of knowledge, and the nature of the mapping between them.

2.1 Describing Bodies of Knowledge

A body of knowledge (BoK) can be described, in abstract terms, using the attributes shown in Table I. In order to specify fully a particular BoK in terms of this description, then, two additional classes are needed, one describing the representation format, the other the abstraction level.

attribute	*type*	*cardinality*	*description*
name	string	1*	the reference name of the BoK.
description	string	1	a natural language description of the content and form of the BoK.
pragmatics	string	1	the pragmatics, in the UPML sense, of the BoK – author, date of creation, etc.
lifecycle-history	lifecycle_term	1*	the history of transformations of the BoK.
representation-format	class	1*	the format in which the BoK is represented; see section 2.1.1 for details of how this might be described.[†]
abstraction-level	class	1*	the level of abstraction of the BoK; section 2.1.2 discusses one approach to describing this.
underlying-ontology	string	1	the ontology through which the BoK is expressed.[‡]

Table I. The attribute descriptors of a Body of Knowledge. An asterisk next to its cardinality indicates that at least one value is mandatory for an attribute. [§]

2.1.1 Representation Format

The representation-format attribute of a BoK is described using one of the concept classes shown in Figure 1; hopefully, the terms in this diagram will be self-explanatory. Whereas we might expect most of the knowledge that we are dealing with to be represented in some formal manner, we might encounter knowledge represented semi-formally or informally (from an AI perspective) in the guise of, say, natural language text documents or sketch diagrams. (Note that neither this diagram, nor the following, is intended to be a complete

[†] There would seem to be no reason why a BoK shouldn't be described using multiple types of representation, or, for that matter, at multiple levels of abstraction, at the same time. However, for the time being I will assume a single representation type and a single level of abstraction.

[‡] I'm not sure what, if anything, we need say about the domain of the BoK. Some transformations (e.g., a neural network training process) are essentially independent of the domain of the knowledge they manipulate, but this might not be the case for more specialised operations.

[§] Note: the attribute *knowledge-type* (i.e., 'procedural domain knowledge', 'ontology', 'UPML refiner' etc.) has been removed from the description of a BoK: it now seems to me that the information that I was trying to express using this attribute is effectively contained within the *abstraction-level* attribute. So, while it may return in some form at some point in the future, I have removed this attribute in the interests of simplicity.

conceptual description; in particular, not all the lower concepts are expanded as completely as might be necessary. However, it *is* intended that the upper levels are reasonably complete.)

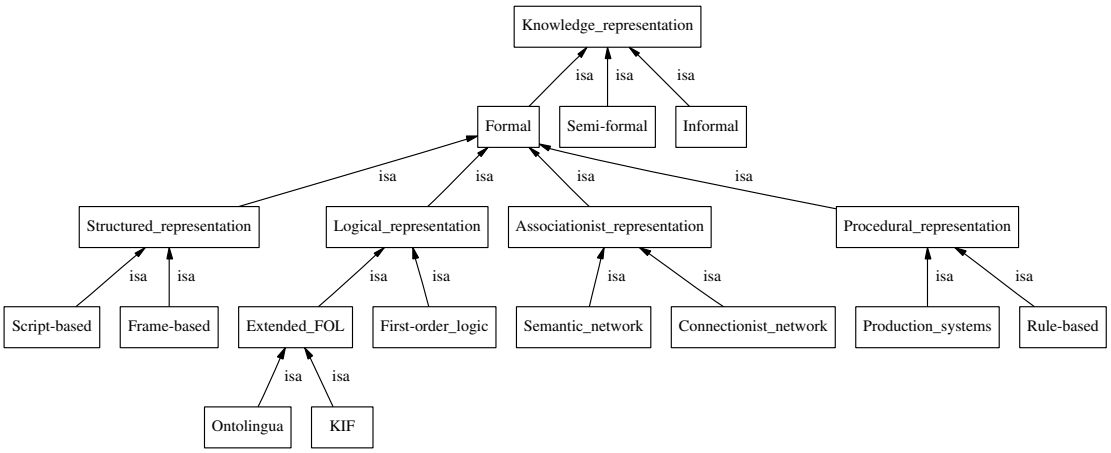


Figure 1. Knowledge Representation conceptual hierarchy.

2.1.2 Level of Abstraction

Figure 2 shows the concepts that can be used to describe the abstraction-level attribute of a particular BoK.

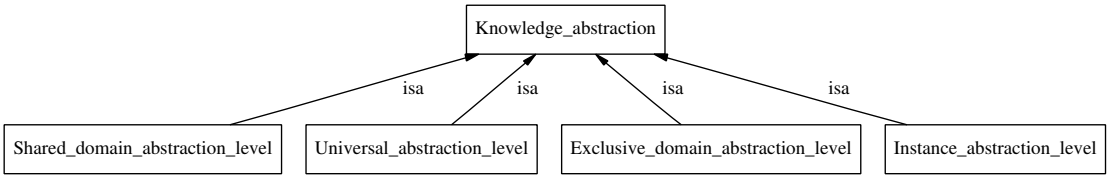


Figure 2. Knowledge abstraction conceptual hierarchy.

Some definitions are needed here:

- *instance_abstraction_level* – knowledge at this level concerns extensional knowledge, containing specific properties (facts, attribute values, etc).
- *exclusive_domain_abstraction_level*[**] – at this level, knowledge focuses on intensional knowledge about some domain which is particular to some application or system. Compare with:
- *shared_domain_abstraction_level* – focuses on ontological knowledge which is common to a particular area, discipline or environment.
- *universal_abstraction_level* – focuses on ontological knowledge which is common to *every* area, discipline or environment.

[**] This is not the ideal name for this level, but I couldn't think of a better one.

Note that there is an implied order in the relative abstraction of these concepts; if '*x* > *y*' is used to indicate that *x* is more abstract than *y*, then this ordering is as follows:

Universal > Shared_domain > Exclusive_domain > Instance abstractions

2.2 Describing Knowledge Transformations

A Knowledge Transformation (KT) can be described using the attributes shown in Table II.

attribute	*type*	*cardinality*	*description*
name	string	1*	the reference name of the KT.
description	string	1	a natural language description of the nature of the KT.
pragmatics	string	1	the pragmatics, in the UPML sense, of the KT – author, date of creation, etc.
input knowledge	instance	*n**	the input BoK(s) described in abstract terms as an instantiation of the description given above in section 2.1.
output knowledge	instance	1*	the output BoK described in abstract terms as an instantiation of the description given above in section 2.1.
transformation type	class	*n**	the general type of the transformation; see section 2.2.1 for details of how this might be described.

Table II. The attribute descriptors of a Knowledge Transformation. An asterisk next to its cardinality indicates that at least one value is mandatory for an attribute.

It should be noted that both the input and the output knowledge is described using *instances* of the BoK concept defined in terms of its attributes above. In addition to these, the transformation type class(es) that the transformation falls into must be specified.

2.2.1 Knowledge Transformation Types

The conceptual classes representing transformation types are shown in Figure 3. This is based on the ideas introduced by Compatangelo and Sleeman. They consider knowledge reuse to be one key to effective knowledge management, since knowledge engineering for knowledge reuse implies the application of good practices throughout the knowledge lifecycle.

With this goal in mind, they identify four groups of knowledge management functionalities; functionalities in successive groups use and build upon the functionalities provided by previous groups. These groups are as follows:

- *knowledge structuring* – functionalities at this level support the creation or modification of BoKs, along with basic syntactic analyses of the knowledge.

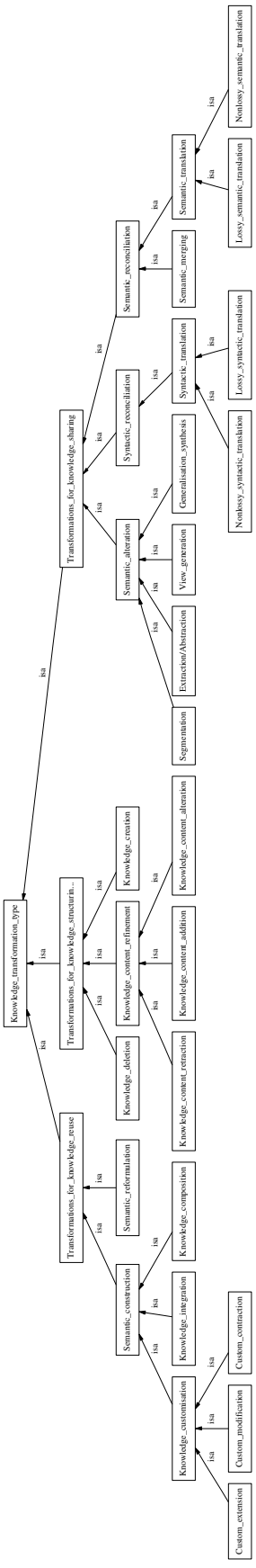


Figure 3. Knowledge transformation type conceptual hierarchy.

- *knowledge analysis* – functionalities at this level perform more detailed semantic analyses of the BoKs.
- *knowledge sharing* – at this level, functionalities effect or support the common representation, interpretation and integration of BoKs.
- *knowledge reuse* – reuse functionalities assist in the adaptation of BoKs to new domains and contexts.

These groups are decomposed further into more specific function-areas, which form the basis of the conceptual hierarchy shown here. Since we are interested in describing knowledge *transformations*, analytical functions are either ignored entirely or else assumed to be incorporated within the depicted transformation concepts, since they do not in themselves bring about a change in BoKs (that said, we will probably want to talk about analytical knowledge services at some stage). Again, no definitions of the lower concepts in the hierarchy are provided; it is hoped that their meanings will be intuited.

3 Examples

To illustrate these ideas in a more concrete fashion, examples of knowledge transformations, described using the above concepts, is shown in Figures 4 and 5. The first example describes some particular neural network training process, which, given structured (e.g., as attribute-value pairs) examples as input, produces as output some generalised description of the concept they embody (in the form of a trained neural network).

The second example is of a "cleanup and prune" operation on an Ontolingua ontology. The transformation involves knowledge content retraction and alteration, but note that the level of abstraction (and representation) of the input and output ontologies remains constant – at the level of generality

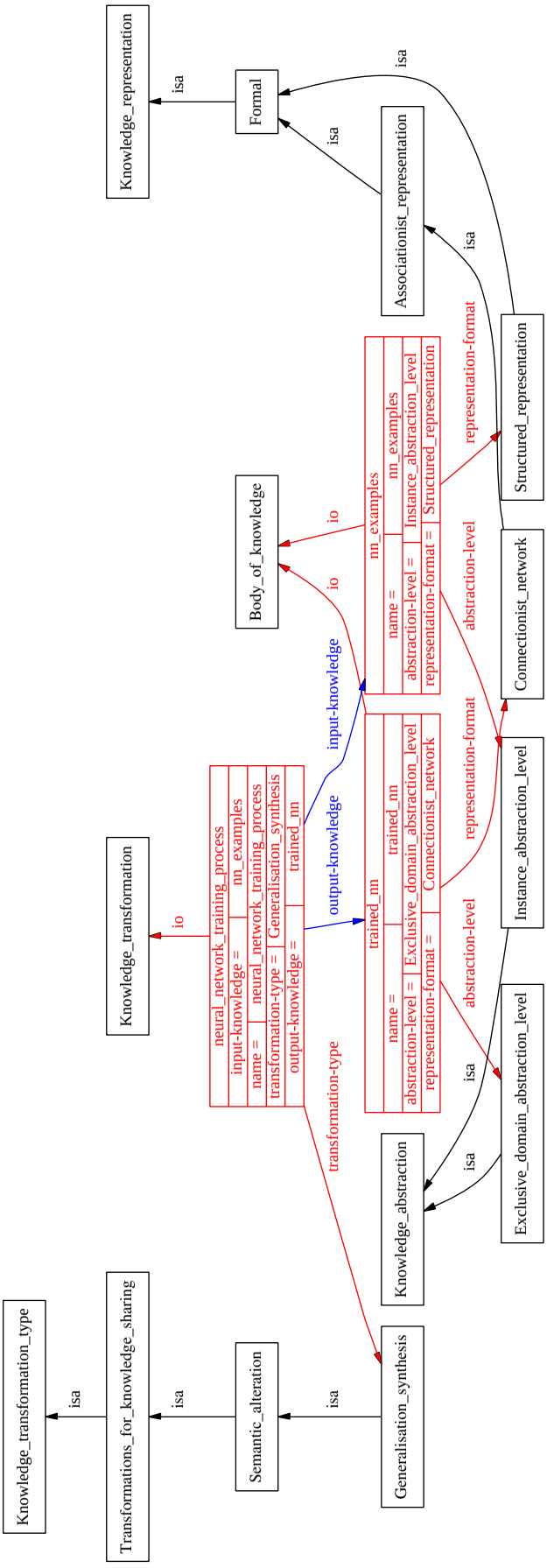


Figure 4. Example representation of a neural network training process.

used here, except for a change in name, the descriptions of the input and the output ontology are identical.

Reference

Compatangelo, E. & Sleeman, D. (2001) *K-ShaRe: a vision of sharing and reuse in AKT*, draft v. 3.2, available from: http://www.csd.abdn.ac.uk/research/akt/reuse/K-ShaRe_vision.pdf

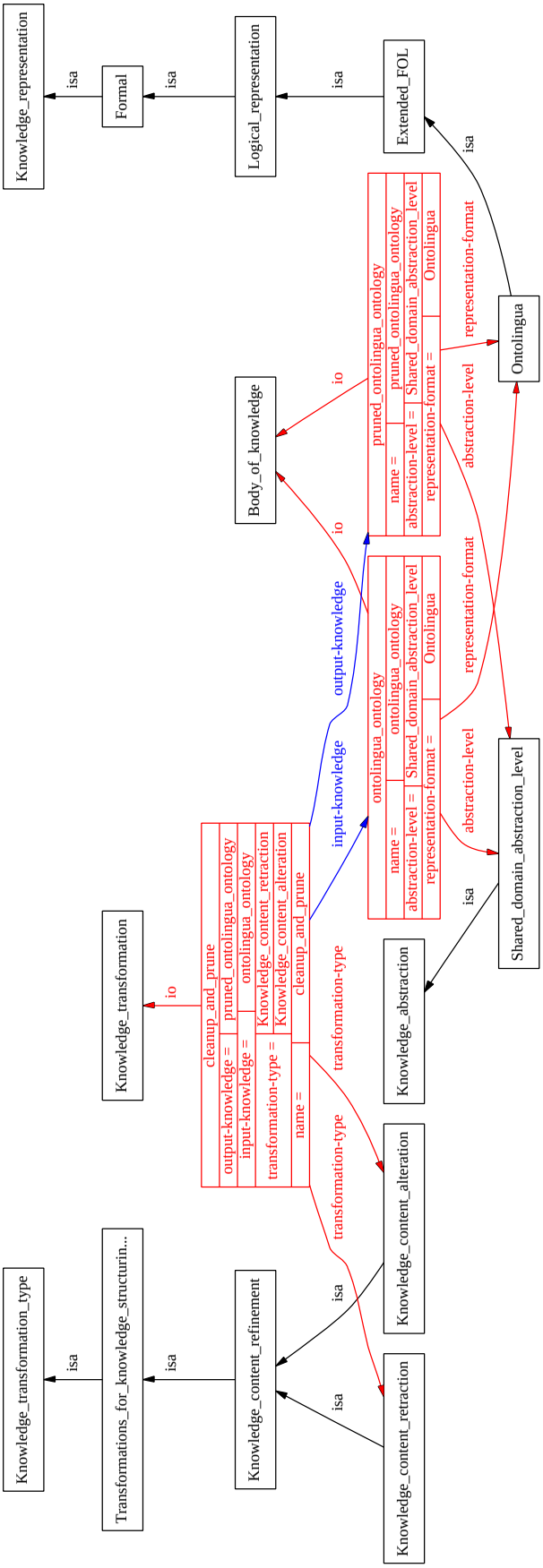
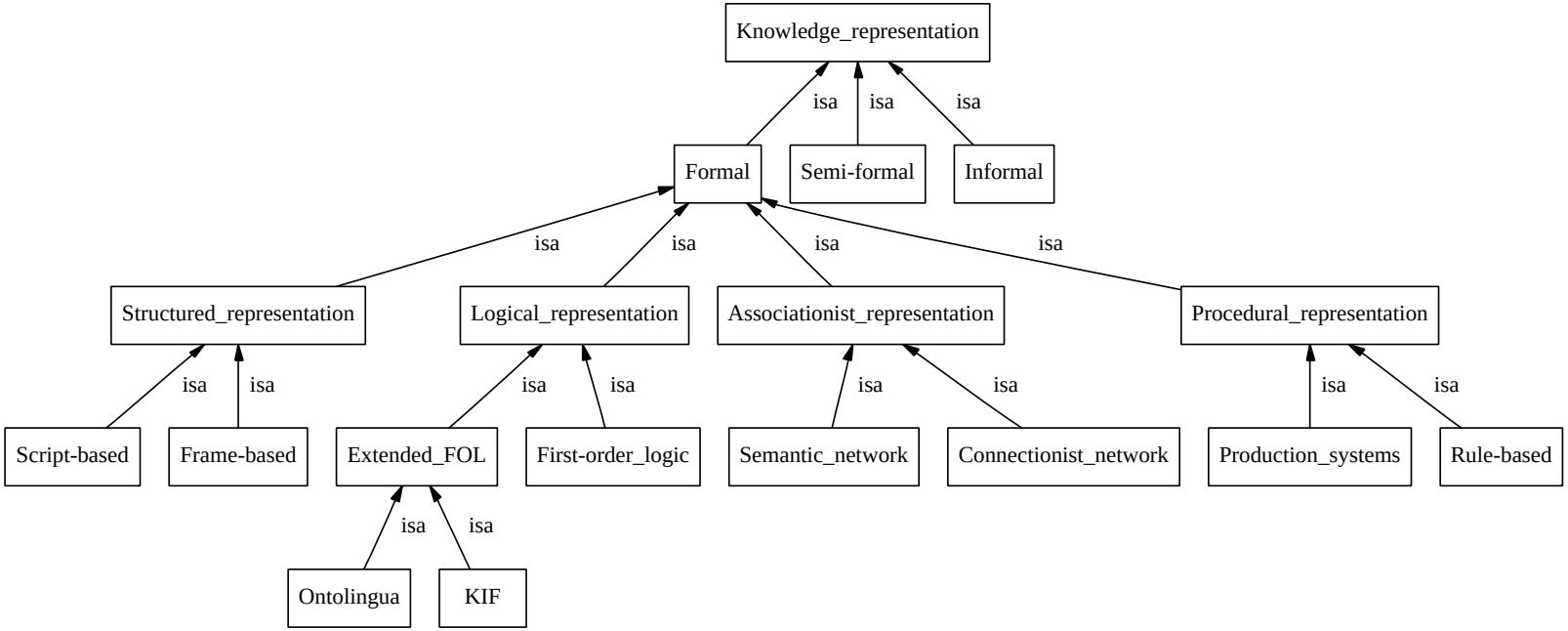
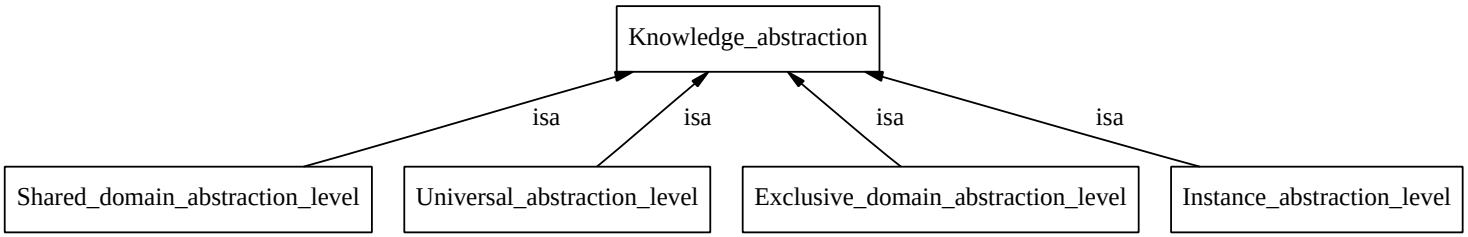
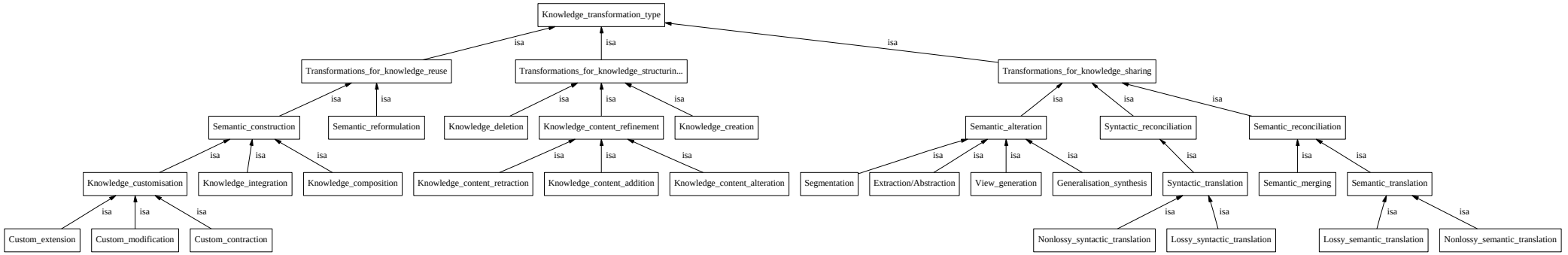


Figure 5. Example representation of an ontology "cleanup and prune" transformation


Knowledge_representation
isa
isa
isa
Formal
Semi-formal
Informal
isa
isa
isa
isa
Structured_representation
Logical_representation
Associationist_representation
Procedural_representation
isa
isa
isa
isa
isa
isa
isa
isa
Script-based
Frame-based
Extended_FOL
First-order_logic
Semantic_network
Connectionist_network
Production_systems
Rule-based
isa
isa
Ontolingua
KIF


Knowledge_abstraction
isa
isa
isa
isa
Shared_domain_abstraction_level
Universal_abstraction_level
Exclusive_domain_abstraction_level
Instance_abstraction_level


Knowledge_transformation_type
isa
isa
isa
Transformations_for_knowledge_reuse
Transformations_for_knowledge_structurin...
Transformations_for_knowledge_sharing
isa
isa
isa
isa
isa
isa
Semantic_construction
Semantic_reformulation
Knowledge_deletion
Knowledge_content_refinement
Knowledge_creation
Semantic_alteration
Syntactic_reconciliation
Semantic_reconciliation
isa
isa
isa
isa
isa
isa
isa
isa
isa
Knowledge_customisation
Knowledge_integration
Knowledge_composition
Knowledge_content_retraction
Knowledge_content_addition
Knowledge_content_alteration
Segmentation
Extraction/Abstraction
View_generation
Generalisation_synthesis
Syntactic_translation
Semantic_merging
Semantic_translation
isa
isa
isa
isa
isa
isa
Custom_extension
Custom_modification
Custom_contraction
Nonlossy_syntactic_translation
Lossy_syntactic_translation
Lossy_semantic_translation
Nonlossy_semantic_translation