USAGE OF RDF BLANK NODES IN PARTIAL LEARNING

Velimir Graorkoski
Z-SoftNet
Portland, USA

Ana Madevska-Bogdanova
Faculty of Computer Science and Engineering
Skopje, Macedonia

ABSTRACT

The partial learning [1], as an important part of the advanced adaptive learning [1], provides opportunities for developing more efficient adaptive learning [2] process than the one occurring in the basic adaptive learning [1]. This paper explains the representation of the learning environment providing the partial learning, by using the introduction of the blank nodes in the resource description framework [3, 4].

KEY WORDS

adaptive learning - AL, partial learning - PL, advanced adaptive learning - AAL, resource description framework – RDF, blank nodes, sufficient concept - SC, sufficient set of in concepts - SS, basic adaptive learning - BAL, learning environment - LE, adaptive learning environment - ALE, in concept - IC, blank concepts - BC

INTRODUCTION

Our definition of PL involving the SCs [1] and SSs [1] as the providers of the knowledge for a certain concept is different from the representation of the AL in BAL through ALE [2] consisting only from the ICs [2] as the knowledge providers. In spite of our goal to represent the PL as simple as possible, the relations of its components make it hard to do it.

We started to examine the PL's concept with multiple IC sets [1], treating them as disjunctive, but in general, especially for very large LEs, the ICs could belong to more than one IC set. It is foolish to make a restriction – only one IC set for a single IC, because this will lead to appearance of significant number of concepts which provide duplicate knowledge and thus increase the number of relations in the ALE. That is why we stayed on course with the non disjunctive IC sets as shown in Fig. 1.

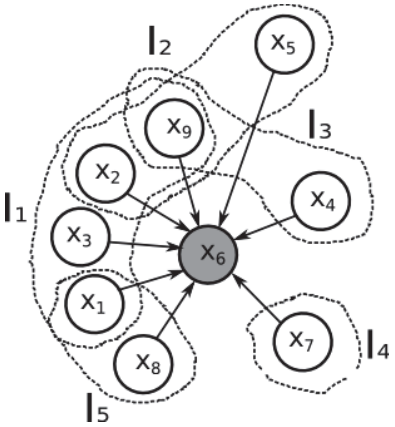

Figure 1: Example of 5 non disjunctive IC sets for a single concept.

According to the example, 8 ICs (x_1, x_2, x_3, x_4, x_5, x_7, x_8, x_9) provide knowledge for the concept x_6. The ICs belong to 5 different IC sets:

- $I_1 = \{x_1, x_2, x_3, x_9\}$;
- $I_2 = \{x_2, x_5, x_9\}$;
- $I_3 = \{x_4, x_9\}$;
- $I_4 = \{x_7\}$;
- $I_5 = \{x_1, x_8\}$;

The concept x_7 is sufficient for x_6 and the concept with the most occurrences in the IC sets is x_9. Although not a great number, those 3 occurrences mean that 3 dotted lines describing the IC sets I_1, I_2 and I_3 will be needed to encircle the concept x_9. So for a larger number of occurrences of a single IC in the IC sets, more space will be needed for this graphical representation.

The situation complicates even further if we try to give a visual representation of a whole ALE, even a small one, as seen in Fig. 2.

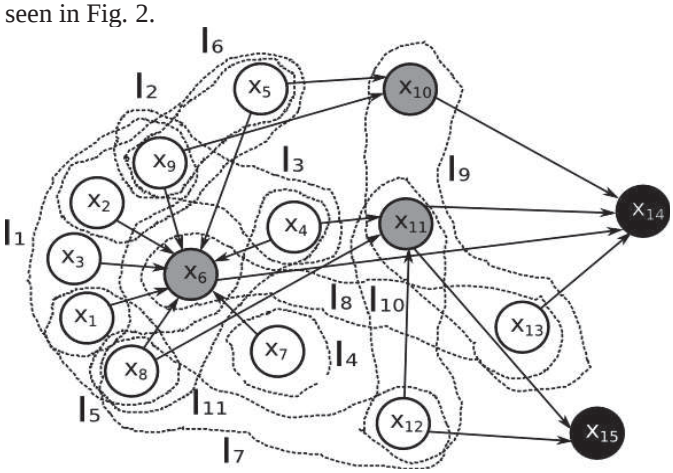

Figure 2: Example ALE with non disjunctive IC sets.

This example of ALE describes the following IC sets providing knowledge to a certain concept:

- $I_1 = \{x_1, x_2, x_3, x_9\} \rightarrow x_6$;
- $I_2 = \{x_2, x_5, x_9\} \rightarrow x_6$;
- $I_3 = \{x_4, x_9\} \rightarrow x_6$;
- $I_4 = \{x_7\} \rightarrow x_6$;
- $I_5 = \{x_1, x_8\} \rightarrow x_6$;
- $I_6 = \{x_5, x_9\} \rightarrow x_{10}$;
- $I_7 = \{x_8, x_{12}\} \rightarrow x_{11}$;
- $I_8 = \{x_6, x_{13}\} \rightarrow x_{14}$;
- $I_9 = \{x_{10}, x_{11}, x_{13}\} \rightarrow x_{14}$;
- $I_{10} = \{x_{11}, x_{12}\} \rightarrow x_{15}$;
- $I_{11} = \{x_4, x_8\} \rightarrow x_{11}$;

The learning arrays for the first terminal concept x_{14} determined by the ICs are the following:

- $\{x_1, x_2, x_3, x_9, x_6, x_{13}\}$ – from I_1 and I_8;
- $\{x_2, x_5, x_9, x_6, x_{13}\}$ – from I_2 and I_8;
- $\{x_4, x_9, x_6, x_{13}\}$ - from I_3 and I_8;
- $\{x_7, x_6, x_{13}\}$ - from I_4 and I_8;
- $\{x_1, x_8, x_6, x_{13}\}$ - from I_5 and I_8;
- $\{x_5, x_9, x_{10}, x_{11}, x_{13}\}$ - from I_6 and I_9;
- $\{x_8, x_{12}, x_{10}, x_{11}, x_{13}\}$ - from I_7 and I_9;
- $\{x_4, x_8, x_{10}, x_{11}, x_{13}\}$ - from I_{11} and I_9;

And for the second terminal concept x_{15}:

- $\{x_8, x_{12}, x_{11}\}$ - from I_7 and I_{10};
- $\{x_4, x_8, x_{12}, x_{11}\}$ - from I_{11} and I_{10};

The most efficient way to learn a concept is determined by the shortest array. According to these arrays, the fastest way to learn the concept x_{14} is by following the array $\{x_7, x_6, x_{13}\}$ and in case of the concept x_{15} it is the array $\{x_8, x_{12}, x_{11}\}$. So in order to learn the goal from the ALE we need total of 3 + 3 = 6 concepts.

But this is not a correct method to find the fastest way to learn the terminal concepts because we treat them separately, as if they do not have common ICs or even IC sets. If we observe the array $\{x_8, x_{12}, x_{10}, x_{11}, x_{13}\}$ we can conclude that only by learning 5 concepts, we can provide all the knowledge needed to learn the two terminal concepts. So the problem of determining the shortest possible way of learning the ALE's goals in the PL is not easy to solve like in the conditions of BAL and more preparations are needed in order to develop an effective strategy.

Not only that in the PL's ALE, the containing ICs, in general, can provide knowledge towards more than one concept, but the concepts inside the IC sets could be related as well (as the case of I_{10} where x_{12} provides knowledge for x_{11}). It means that the 4 occurrences of the concept x_9 in 4 different ICs is not very complicated compared to the encirclement of the related concepts. And this example is hard enough to visualise even with only two terminal concepts. The representation will look like a mess in case of large number of terminal concepts.

To avoid these representation obstacles, primarily the dotted lines, we came up with two more suitable and understandable visual appearances before taking the step towards the pure graph representation easier to implement.

Decomposition of ALE

One method to achieve pure graph representation of ALE in the AAL is by decomposition of its dotted line-based visual representation, resulting in several oriented graphs.

This is very simple solution because all it is needed is the determination of the combinations of PL processes where only one IC set per concept is allowed. The number of those combinations is determined by the formula (1) – the product of the number of IC sets for every concept.

$$\left|\left\langle G_i \right\rangle\right| = \prod_{x \in V_{ALE}} \left|I(x)\right| \qquad (1)$$

For the example in Fig. 2. there are 5*1*2*2*1 = 20 possible combinations, hence 20 different graphs in which every concept is provided with knowledge from only one IC set. Some of those 20 graphs for the decomposed ALE from the Fig. 2. are given in Fig. 3-6.

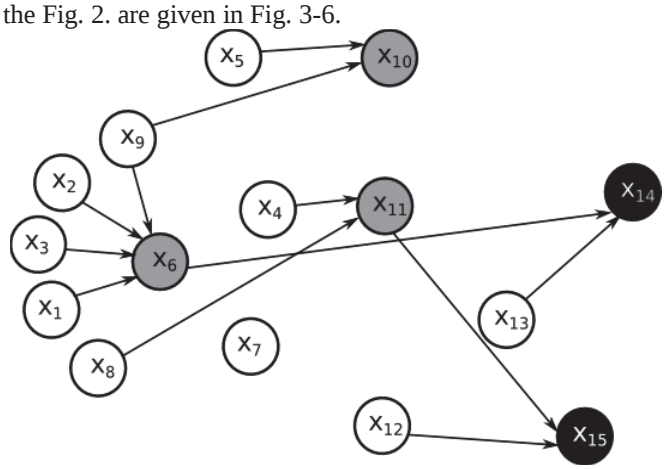

Figure 3: One possible result from the decomposition of ALE from Fig. 2

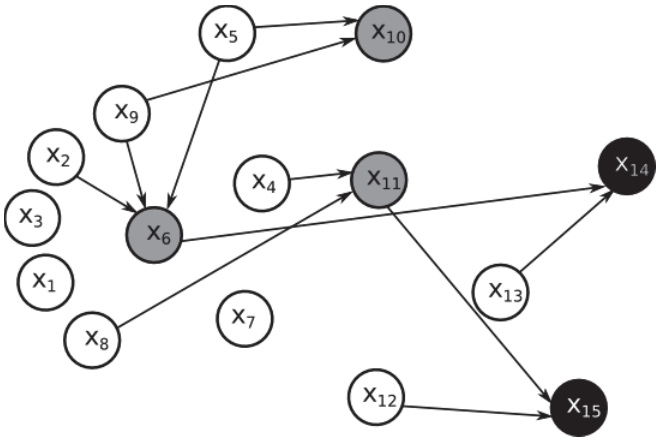

Figure 4: One possible result from the decomposition of ALE from Fig. 2

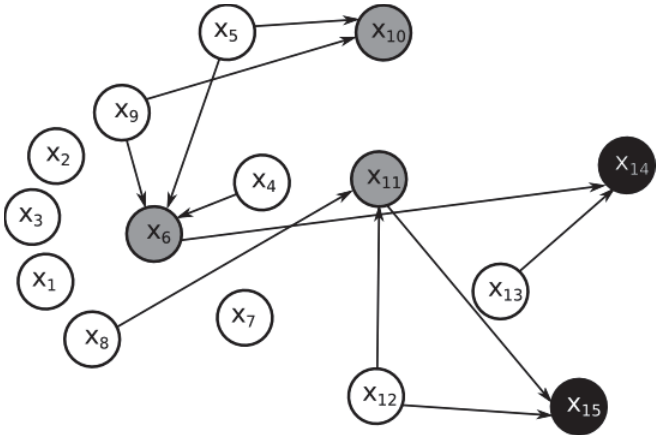

Figure 5: One possible result from the decomposition of ALE from Fig. 2

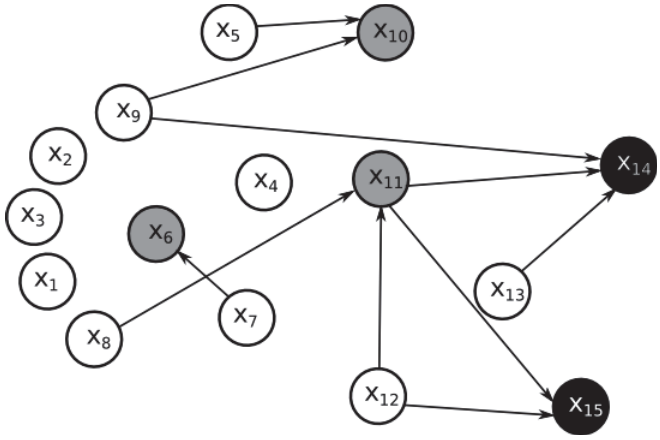

Figure 6: One possible result from the decomposition of ALE from Fig. 2

Having the ALE decomposed with the resulting graphs, the second step will be to compare the shortest learning arrays[1] for every one of them. The one with the smallest number of concepts will be the output of the method for the most efficient learning way.

By comparing the resulting graphs in Fig. 3-6, it is obvious that the smallest number of ICs in the learning array has the last graph, totalling 7.

Given this approach, we "chopped" one complex problem in several simple ones, relatively easy to solve with the traditional AL methods for the knowledge delivery order in BAL.

RDF Blank Nodes

In the theory of graphs, especially when it comes to the weighted oriented graphs [5], the existence of two non adjacent vertices is considered as they have two edges in between with weight equal to 0. This treatment is useful when representing the oriented graphs as adjacency matrix [5] – data type suitable for further computations. Although this is more edge's feature, rather than vertex's, it could be interpreted as the vertex does not exist at all, of course, if it is not adjacent to any other vertex[2].

This behaviour is not often used as a special case in the implementations of the graph theory, primarily because of the popular algorithms using the adjacency matrix, and because in general all vertices in a graph are treated as equal with the edges as the carriers of the differences.

The forcing of the vertex similarity came to an end with the introduction of the blank nodes in the RDF. RDF as a major component of the semantic web, for describing the metadata expressions [3], is based on the oriented graphs and their features. Having found this common ground with our knowledge representation of the AL structures, we managed to "translate" the relation between the concepts in the knowledge structures in BAL, into RDF expression. The RDF expressions generated from the BAL's ALE always have "ProvidesKnowledgeFor" as predicate, with the IC as subject and the concept which receives the knowledge as object. We made this generation almost fully functional in our AL system Awareness (by using and modifying RAP [6] libraries) which supports initializing rdf file for an ALE consisting of RDF expressions appropriate to the relation between two of ALE's concepts. One such example is shown in Fig. 7. During this realization, we had no restrictions, because the BAL does not allow one concept to be learned by different sets of ICs.

```
<rdf:Description rdf:about="#x5">
   <ns1:ProvidesKnowledgeFor>x10</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x7">
   <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x8">
   <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x9">
   <ns1:ProvidesKnowledgeFor>x10</ns1:ProvidesKnowledgeFor>
   <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x11">
   <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
   <ns1:ProvidesKnowledgeFor>x15</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x12">
   <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
   <ns1:ProvidesKnowledgeFor>x15</ns1:ProvidesKnowledgeFor>
</rdf:Description>
<rdf:Description rdf:about="#x13">
   <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
</rdf:Description>
```

Figure 7: RDF expressions generated in Awareness[3] using the example in Fig. 6

Although we stopped experimenting with the RDF as an alternative realization of our knowledge structures in BAL using Awareness, it still serves as a proof that the relations between the concepts represented in a graph can be transformed into RDF expressions.

The representation of ALE in Fig. 2 can not be immediately transformed into a set of RDF expressions because it is not a pure graph at all. But using a blank node in the RDF expression can provide a bridge between the need for decomposition and the RDF generation.

The purpose of the blank node in the RDF is to define the usage of the unnamed resources in order to classify and link the subjects with object over more than one predicate. We will use this feature of the blank nodes to create special concepts in the ALE for every IC set.

In PL we will refer to the blank nodes as BCs because of the synchronized terminology, since the knowledge units of the knowledge structures are called simply concepts. This method of creating a new visual representation based on graphs starts with inserting a BC between the IC set and the concept which gains knowledge from it, while removing the appropriate dotted line bordering the concepts for the same IC set. All of the concepts from the IC set will now act as ICs for the BC and the BC will act as an IC for the concept which previously gained knowledge from the IC set. According to this, the BC does not posses any knowledge to add, but simply passes the gained knowledge from its ICs to another concept.

The essential differences between the BC and the regular concept will be expressed by Def. 1 and Fea. 1.

1 To find the shortest learning array for the resulting graph, the appropriate strategies from BAL can be used without restrictions

2 If it does not have edge with weight greater than 0

3 Although not correct for the original RDF, there are duplicate predicates for the ICs x_9, x_{11} and x_{12} in order to make the RDF file smaller and thus save space

Definition 1: BC vs. regular concept

> One BC can provide knowledge towards only one regular concept, while one regular concept can provide knowledge towards more than one BCs.

Feature 1: BCs in ALE

> In ALE with inserted BCs, any two BCs can not be adjacent, and also any two regular concepts can not be adjacent.

The oriented graph with inserted BCs is shown in Fig. 8. The BCs are named after the IC sets, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10} and I_{11}, respectively.

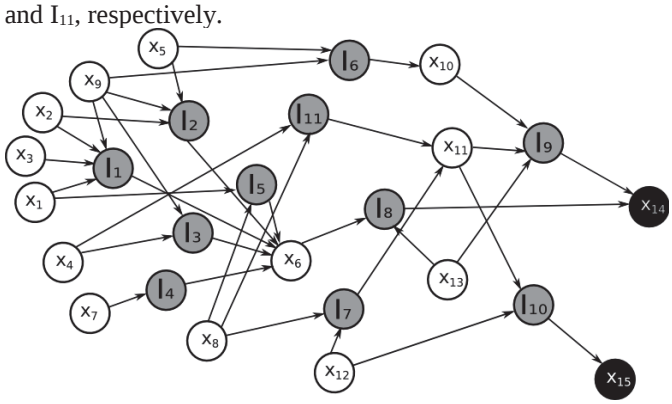

Figure 8: ALE with inserted BCs

The generated RDF file for the example in Fig. 8 is shown in Fig. 9.

The BCs are not included in the learning array, but they do determine the learning order i.e. the way the array is formed. They mark the disruption of the learning order according to the following rules:

- in order to get to a BC, all of its ICs have to be passed first;
- in order to get to a regular concept, it is enough for only one of its BCs to be passed first;

Using these rules it is very easy to construct a learning array in which the blank concepts are not considered. For now, we will not consider choosing our curriculum sequencing techniques[4] such as BFL or DFL [2], since their involvement will decrease the variety of learning arrays. Also we will not make any assumptions as the one which states that every concept will get the learning priority according to its index but only in the scope of their child blank concept. Hence, the blank concepts' priorities will not be determined by the same rule, although we will give some examples which coincidentally match these strategies.

In order to save space, we will always treat the sub array of the regular concepts as ordered, no matter the combination of the concepts' indexes. So for the example on Fig. 8, it does not matter whether the learning order prior to providing knowledge towards the BC I_1 is x_2, x_9, x_3, x_1 or x_9, x_2, x_1, x_3, the sub array will be marked according to their priority determined by the indexes.

4 In our previous papers examining the process of learning in BAL conditions, we refer to them as knowledge delivery order strategies

```
<rdf:Description rdf:about="#x1">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x2">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x3">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x4">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x5">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x10</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x6">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x7">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x8">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x9">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x6</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x10</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x10">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x11">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x15</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x12">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x11</ns1:ProvidesKnowledgeFor>
    <ns1:ProvidesKnowledgeFor>x15</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
<rdf:Description rdf:about="#x13">
  <ns1:BC>
    <ns1:ProvidesKnowledgeFor>x14</ns1:ProvidesKnowledgeFor>
  </ns1:BC>
</rdf:Description>
```

Figure 9: RDF expressions generated for the ALE in Fig. 8

Here are several learning arrays for the example on Fig. 8:

- $x_1, x_2, x_3, x_9 \rightarrow I_1 \rightarrow x_5 \rightarrow I_2 \rightarrow x_4 \rightarrow I_3 \rightarrow x_7 \rightarrow I_4 \rightarrow x_8 \rightarrow I_5 \rightarrow I_6 \rightarrow x_{12} \rightarrow I_7 \rightarrow x_6, x_{13} \rightarrow I_8 \rightarrow x_{10}, x_{11} \rightarrow I_9 \rightarrow I_{10} \rightarrow x_{14}, x_{15}$;
- $x_4, x_8 \rightarrow I_{11} \rightarrow x_{11} \rightarrow x_{12} \rightarrow I_{10} \rightarrow x_{15} \rightarrow x_5, x_9 \rightarrow I_6 \rightarrow x_{10} \rightarrow I_9 \rightarrow x_{14}$;

In the first array it can be noticed that x_{11} does not have to wait for I_{11} because it is enough that I_7 appears with lower index in the learning array. This satisfies the second disruption rule. However this array unnecessary includes all concepts of the ALE thus making the PL redundant.

The second learning array is shorter and represents a typical example of the PL's usage where every concept does not have to be learned in order to reach the goal. Therefore it is shorter than the first one by the concepts x_1, x_2, x_3, x_6, x_7 and x_{13} which are not obligatory in the learning process, as well as the blank concepts I_1, I_2, I_3, I_4, I_5, I_7 and I_8.

AAL Database

The addition of BCs in our development of the AAL system means that certain changes had to be made in our database realization of the ALE, deriving from the previous database design of Awareness. In Awareness database we used dynamic creation of tables for every set of concepts with equal structure[5] with the number of columns according to the number of their ICs.

With the involvement of the PL rules, especially with the usage of different IC sets, we could not effectively sustain the dynamic creation of tables since the structure of a concept is no longer constant – the number of ICs for a concept is variable according to the chosen IC set. It implies risking the newly created tables to have ambiguous labels and need of additional logic of initialization in order to solve it. The usage of the blank concepts will have a final influence of the database design for our new AAL model.

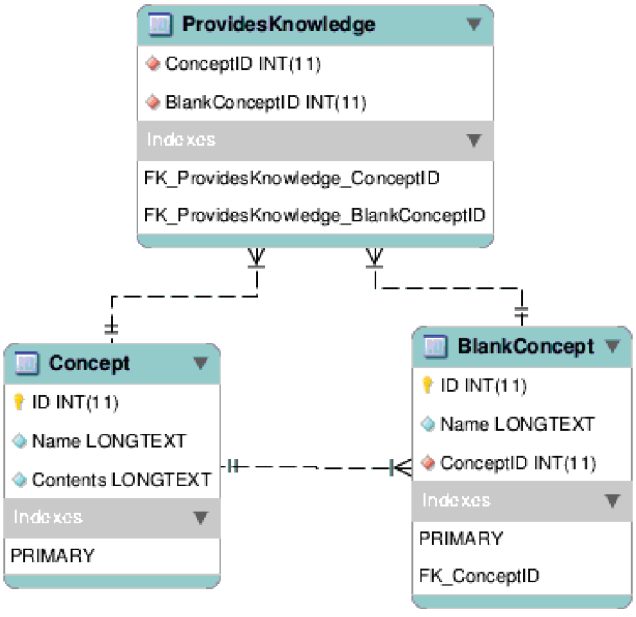

Figure 9: Diagram of AALDB database

5 Equal number of ICs [2]

Our new database – AALDB, will have a design far simpler than the Awareness database. As it can be seen in Fig. 9, all concepts, no matter the number of ICs they have, are stored in a single table - Concept. Same is the situation with the table BlankConcept where the affiliation with the IC is provided for every BC.

The table ProvidesKnowledge expresses the relation between the ICs and the BCs, by following the Def. 1 – an IC may provide knowledge for more than one BC. The reason for non existence of a table-relation between the tables BlankConcept and Concept also lays in the Def. 1 – a BC can pass knowledge only for one regular concept.

Conclusion

Although the two approaches for constructing the best possible ALE's representation in AAL result in almost similar implementations, the one using the idea of the RDF blank nodes saves more space and it is easier to follow the PL process with it. But this is only the impact on the theoretical and experimental side.

When it comes to the implementation of the PL rules in the AALDB, the addition of BCs helps in making a classification of the regular concepts according to the IC set. Having this done with the addition of a single table, only adds to the simplicity and improvements it gives to the AAL operations[6]. The only negative effect we came up so far is the increasing number of rows inside every table[7], although the removal of the classification by the concepts' structure is partially responsible for that.

References

[1] V. Graorkoski, A. Madevska-Bogdanova, M. Gusev, *Beyond the basics of the adaptive learning.* ICT-ACT conference, Skopje, 2011.

[2] V. Graorkoski, M. Gusev, *Model of adaptive e-learning*. master thesis, Institute of Informatics, PMF, University "Sv. Kiril i Metodij", Skopje, 2010.

[3] M. Nilsson, *The semantic web: How RDF will change learning technology standards*. Center for User-Oriented IT-design, Royal Institute of Technology, Stockholm, 2001.

[4] A. Mallea, M. Arenas, A. Hogan, A. Polleres, *On Blank Nodes*. 10th International Semantic Web Conference - ISWC2011, 2011.

[5] D. Jungnickel, *Graphs, networks and algorithms*. University of Augsburg, 1998.

[6] R. Oldakowski, C. Bizer, D. Westphal, *RAP: RDF API for PHP*. Freie Universität Berlin, Institut für Produktion, Wirtschaftsinformatik und OR Garystr. 21, D-14195 Berlin, Germany, 2005.

6 The db operations are executing approximately by 14% faster than in the Awareness database

7 Even having greater number of rows per table, the overall space needed for the AALDB is still by over 70% less than the one required for the Awareness database