

Article

ByNowLife: A Novel Framework for OWL and Bayesian Network Integration

Foni A. Setiawan ^{1,2}, Eko K. Budiardjo ^{1,*} and Wahyu C. Wibowo ¹

¹ Faculty of Computer Science, Universitas Indonesia, Kampus UI Depok, West Java 16424, Indonesia; foni.agus@ui.ac.id (F.A.S.); wibowo@cs.ui.ac.id (W.C.W.)

² Research Center for Limnology, Indonesian Institute of Sciences, Jakarta 12710, Indonesia

* Correspondence: eko@cs.ui.ac.id

Received: 12 December 2018; Accepted: 21 January 2019; Published: 5 March 2019



Abstract: An ontology-based system can currently logically reason through the Web Ontology Language Description Logic (OWL DL). To perform probabilistic reasoning, the system must use a separate knowledge base, separate processing, or third-party applications. Previous studies mainly focus on how to represent probabilistic information in ontologies and perform reasoning through them. These approaches are not suitable for systems that already have running ontologies and Bayesian network (BN) knowledge bases because users must rewrite the probabilistic information contained in a BN into an ontology. We present a framework called ByNowLife, which is a novel approach for integrating BN with OWL by providing an interface for retrieving probabilistic information through SPARQL queries. ByNowLife catalyzes the integration process by transforming logical information contained in an ontology into a BN and probabilistic information contained in a BN into an ontology. This produces a system with a complete knowledge base. Using ByNowLife, a system that already has separate ontologies and BN knowledge bases can integrate them into a single knowledge base and perform both logical and probabilistic reasoning through it. The integration not only facilitates the unity of reasoning but also has several other advantages, such as ontology enrichment and BN structural adjustment through structural and parameter learning.

Keywords: bynowlife; knowledge base; logical reasoning; probabilistic reasoning; ontology enrichment; Bayesian network structural adjustment; owl and Bayesian network integration

1. Introduction

In Web Ontology Language (OWL), reasoning in ontologies is supported using a sub-language that accommodates a description logic (DL)-based reasoning called OWL DL [1]. OWL DL is the most important OWL sub-language because it supports maximum expressiveness while still providing computational completeness and decidability [2]. The fundamental modeling concept in DL is an axiom, a logical statement related to roles and/or concepts. The mechanism of reasoning using such logic is referred to as logical reasoning.

However, the type of required reasoning is not always purely logical (true or false, 1 or 0) but can also be probabilistic (the degree of certainty or probability that an event will happen, represented by a value between 0 and 1). Bayesian networks (BNs) have been chosen by many previous researchers as models for managing probabilistic reasoning in ontologies [3]. This is because in addition to its ability to perform probabilistic reasoning with prior knowledge, a BN has a graphical structure like that supported by ontological representation in OWL.

Various studies related to the integration of probabilistic information into ontology have been carried out and applied to various areas of research such as image classification [4], knowledge repositories for website evaluation methods [5], and decision support systems for terrorist identification [6]. Probabilistic

ontology has great potential as a knowledge base that is able to address challenges of logical and probabilistic reasoning simultaneously.

OWL DL and its variations have been standardised by the World Wide Web Consortium (W3C) for logical reasoning in ontology [7]. However, W3C has not yet established a standard for probabilistic reasoning in ontology. Previous studies have attempted to fill this gap by combining BNs with ontologies to make probabilistic reasoning available for use in ontology, but these have mainly focused on how to represent probabilistic information in OWL notation. For future systems or those that are newly built, the process of establishing an ontological knowledge base containing probabilistic information from the ground up may not be a problem. For systems that already have ontologies and BN knowledge bases, it is unnecessary to rewrite the probabilistic information contained in the BN into the OWL; it will require significant effort, especially if the BN consists of thousands of nodes and relations. In cases such as this, a machine capable of transforming the information contained in the BN into OWL and vice versa is needed.

We have other ideas about this, motivated by the belief that incorporating a BN into an ontology and vice versa should be more than just a transformation process; it should also benefit them both. This paper presents a concept called the Bayesian Network and OWL Integration Framework (or ByNowLife), a novel approach to integrating BNs with OWL by providing an interface for probabilistic reasoning through SPARQL queries. To the best of our knowledge, this work is the first to integrate BNs into OWL and vice versa, and exploit the integration process. The main contributions of this research are as follows: (1) ontology enrichment based on BN knowledge bases; (2) BN structural adjustment through structural learning based on ontology knowledge bases; and (3) support for an integrated probabilistic clause in the SPARQL query format.

This paper is organized as follows: Section 2 outlines previous work related to the merging of ontologies and BNs. Section 3 describes ByNowLife as the proposed approach as well as the translation rules and algorithms used. Section 4 presents the experiment as an implementation framework using two validation cases to demonstrate the feasibility and effectiveness of our approach. Section 5 details the reasoning proof, while Section 6 discusses the results of the experiment and important aspects of the framework. Section 7 concludes the study, and Section 8 suggests several next steps for future work.

2. Related Work

Systematic literature review (SLR) results [3] reveal four published frameworks related to the integration of ontologies with BNs: BayesOWL, Multi-Entity Bayesian Network/Probabilistic OWL (MEBN/PR-OWL), OntoBayes, and HyProb-Ontology. Each framework has pros and cons. Several comparisons of the pros and cons of these frameworks are discussed in [8].

BayesOWL [9] was the first framework to attempt to integrate ontology with BN. It was created with the primary goal of representing the BN in Resource Description Framework (RDF)/OWL notation in a simple structure to be used as the ontology knowledge base. Alongside general nodes, BayesOWL generates bridge and intersection nodes to facilitate the modeling of relations between classes. Reasoning is done using the decomposed iterative proportional fitting procedure (D-IPFP) algorithm [10] to calculate the probability values for each node in the system. In addition to including a graphical user interface (GUI) for direct use, BayesOWL was also designed for software developers by providing application programming interface (API) in the form of Java packages. This is notable because it provided an interface for software developers to create their own ontology-based applications capable of representing probability values in its OWL knowledge base. However, BayesOWL does not have query support in SPARQL format as a query standard in its knowledge base. In addition, BayesOWL does not feature a blending process or “symbiotic mutualism” between ontologies and BNs, so it does not offer ontology enrichment or BN network structural adjustment through structural and parameter learning.

MEBN/PR-OWL does a different way by creating a separate environment for users who want to perform a combined operation between BN and ontology. According to work pertaining to MEBN

and its application characteristics, namely UnbBayes-MEBN [11–14], this application is aimed more at processing probabilistic graphical models (PGMs) but with support for outputting results in RDF/OWL format with PR-OWL style. Because the process must be done in a separate application environment, practically software developers cannot use the engines provided by this framework. Reasoning in the PR-OWL knowledge base is carried out through the generation of a situation-specific BN (SSBN), which is the minimum BN construction sufficient to calculate the probability values of target nodes based on existing evidence values. The SSBN generation algorithm was initially introduced by Laskey in [14] with the bottom-up algorithm, which was later refined by Santos in the Bayes–Ball algorithm [15]. Similarly to BayesOWL, MEBN/PR-OWL cannot query in SPARQL format, does not offer ontology enrichment and does not support BN structural and parameter learning based on ontology knowledge bases.

OntoBayes [16] is the successor of BayesOWL, and the two share many similarities. The knowledge structure in OntoBayes is simpler than that of BayesOWL; OntoBayes does not generate additional nodes or control nodes. Thus, the OWL document structure in OntoBayes can easily be transformed into a BN with simple rules. If BayesOWL uses D-IPFP as its reasoning algorithm, then OntoBayes utilizes the Netica-J API library as its reasoning engine. OntoBayes has logical query support in SPARQL format but does not support probabilistic clauses in it. OntoBayes also does not provide component-reuse support, so its engines are unlikely to be exploited by application developers. Ontology enrichment and BN structural and parameter learning are also not supported by this framework.

HyProb-Ontology [17] is a framework that enhances the data input and reasoning processes of its predecessors. Instead of using ordinary BNs, HyProb-Ontology uses a hybrid BN that incorporates BNs for probabilistic reasoning on discrete variables and conditional Gaussian fuzzification for probabilistic reasoning on continuous variables. There is no significant improvement other than these.

Several algorithms for transforming ontology into BNs have been investigated by previous researchers. Two were proposed by Fenz et al. [18], who outline the procedure for translating an ontology into a BN and Ishak et al. [19], who describe an algorithm for transforming (morphing) an ontology into the form of an object-oriented BN (OOBN). These two algorithms inspired the one we developed for merging an ontology into a BN. In addition, we developed an algorithm to merge a BN into an ontology.

Of the features of previous frameworks, there are three aspects that have never before been successfully implemented in past research: ontology enrichment based on BN knowledge bases, the adjustment of BN structures through structural and parameter learning based on ontology knowledge bases and support for an integrated probabilistic clause in SPARQL query format. This study accomplished these three goals in the form of an enhanced framework, as described in the following sections.

3. Proposed Approach

This section describes our proposed approach, namely the ByNowLife framework, its translation rules, the reasoning proofs and the merging algorithms it uses.

3.1. The ByNowLife Framework

We propose a framework for systems using ontologies as their knowledge bases and requiring both logical and probabilistic reasoning abilities simultaneously. We call it ByNowLife, which stands for Bayesian Network and OWL Integration Framework. It is shown in Figure 1.

This framework consists of three main parts: the Application, the Reasoner and the Knowledge Base. The Application can either be independent software or a software agent written in a common programming language, such as C/C++, Java, PHP, or Python. The Reasoner involves two components: (1) the Logical Reasoner, assigned to perform logical reasoning based on SROIQ specifications in OWL 2 DL and (2) the Probabilistic Reasoner, assigned to perform probabilistic reasoning for the Knowledge Base in the form of a BN, OOBN, or dynamic BN (DBN). The Knowledge Base contains knowledge

of two forms: ontology knowledge in OWL/RDF format and BN knowledge in XML of Decision Systems Laboratory (XDSL) format [20]. In the Knowledge Base, there is a component named the Morpher that conducts data transformation from standardised ontologies (OWL/RDF) to XDSL (and vice versa). The Morpher also serves to enrich the ontology with the probability values of the nodes and links in the BN. The probability values of the nodes in the BN are translated into axioms in the ontology and links are translated into relations. This framework allows the application to query the Knowledge Base in SPARQL format for logical reasoning and the *hasProbValueOf* special property for probabilistic reasoning.

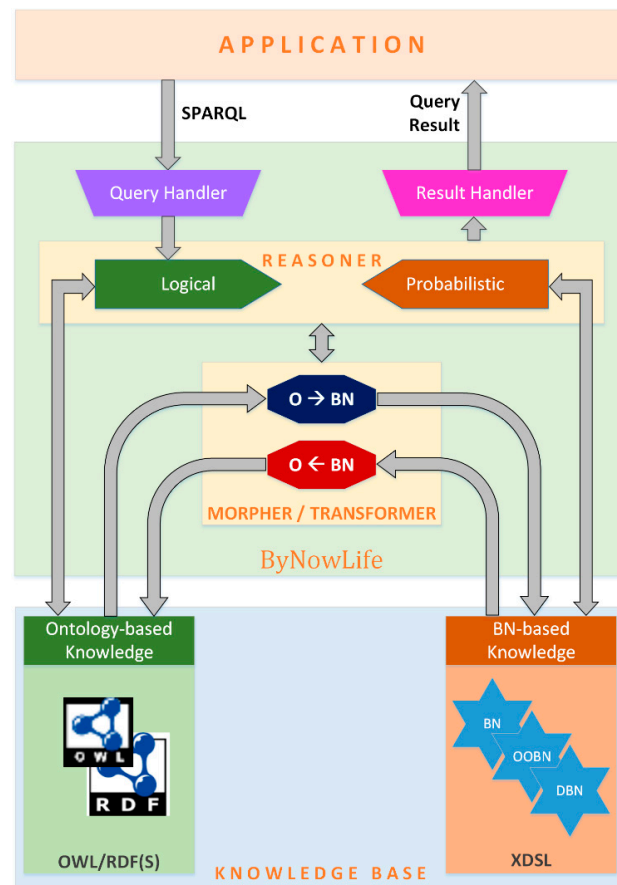



Figure 1. The ByNowLife framework.


ByNowLife was developed not only as a theoretical foundation in combining logical with probabilistic knowledge bases but also a technical implementation that can be tested with various test cases. Online technical implementation of this framework can be found on the ByNowLife website [21]. The system requires input in the form of logical knowledge-based files (.owl) and probabilistic knowledge-based files (.xDSL). After both files are uploaded and the knowledge bases have been integrated, the user can query to communicate with the system. Users can also download the resulting .owl and .xDSL files from the system if needed. Figure 2 shows a screenshot of the website's content.

3.2. Translation Rules

We adopted the translation rules of Fenz et al. [18] for the BN as well as those from Ishak et al. [19] for the OOBN. Both Fenz and Ishak account for the class for BN transformation into ontology, while our approach puts the class aside and prefers to replace it with the individual. This is because in its implementation, an ontology scheme is assumed to be deterministic or precise, but this is not so with



the instance of the ontology. This means that the classes and its hierarchy are deterministic. However, the relationship between an individual and a class/concept is valid with a certain probability. In the same way, the relationship between two individuals or between individuals with data types is valid with a certain probability. Thus, uncertainty operations must be applied to relationships between individuals but not those between classes. See [22] for a more detailed explanation.


ByNowLife
 Bayesian Network and OWL Integration Framework
 Version 0.6

 Hi masagus! [\[Change Password\]](#) [\[Logout\]](#)

Project Details

Project Name : **Investment Problem - Scenario #1**
 Description : An investor has shares in the following stocks: Mandiri, KPC, and Telkomsel. Mandiri is a bank, KPC is a mining company, and Telkomsel is a telecommunication company. Each of these stocks has a Price property. On the other hand, there is a government policy that affects the fluctuation of the stock price.
 Last Modified : 2018-07-02 19:54:23

 FILES
 **QUERIES**

Query Details

The IRI of OWL result files is:

http://www.bynowlife.net/projects/1/result/<OWLFileName>

Please conform the IRI in the SPARQL command.

Query Name

SPARQL Command

Most Profitable Stock

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX stocks: <http://www.bynowlife.net/projects/1/result/stocks#>
SELECT ?StockName ?PriceUpProbValue
WHERE {
  ?StockName rdf:type stocks:Stocks .
  ?StockName stocks:Price ?StockPrice .
  ?StockName stocks:hasProbValueOfProfit ?PriceUpProbValue
  FILTER (?StockPrice >= "5000"^^xsd:double)
}
ORDER BY DESC(?PriceUpProbValue)
LIMIT 1
          
```

Last Modified 2018-06-29 07:26:10

« Back
Save
▶ Execute!

Query result

```

-----
| StockName                                     | PriceUpProbValue
|-----|-----|
| <http://www.bynowlife.net/projects/1/result/stocks#Telkomsel> | "0.430000000000000005"^^<http://www.w3.org/2001/XMLSchema#double> |
-----
          
```



 Export to Text File
 Export to CSV File

Figure 2. The implementation of ByNowLife framework at [21].

A BN contains the following elements:

1. **Node**;
2. Inter-node relationship (**Link**);
3. **Scale** and **Weight** of a node, both independently and in relation to other nodes; and
4. Factual values (**Evidence**) from one or several nodes based on facts.

On the other hand, an ontology contains the following elements:

1. Concept or **Class**;
2. Instance of a class (**Individual**);
3. The attributes inherent in the class or individual (**Properties**) along with their **Values**; and
4. Inter-class or inter-individual **Relationships**.

The rules used to transform elements in BNs into ontologies are as follows:

1. **Node** → **Individual**. Nodes in BNs are transformed into Individuals in ontologies.
2. **Link** → **Relationship**. Links in BNs are transformed into Relationships or Object Properties in ontologies.
3. **Scale** → **Property**. Scales are values that a Node may have. Scales in BNs are transformed into Data Properties in ontologies.
4. **Weight** → **Value**. Weights is the value of the influence of a Node on another Node. Weights in BNs are transformed into Values of Data Properties in ontologies.
5. **Evidence** → **Value**. Evidence in BNs is transformed into Values of Data Properties in ontologies.

3.3. Merging Algorithms

We present the two merging algorithms used in the Morpher. The first algorithm merges BNs into ontologies, and the second merges ontologies into BNs.

3.3.1. Merging BNs into Ontologies

We have developed an algorithm to merge the information contained in a BN into an ontology. The steps are as follows.

1. Check the ontology. If there are no properties as follows, then create the following:
 - a. An object property named *influences_*;
 - b. An object property named *influenced_by_*; and
 - c. A data property named *hasProbValueOf*.
 Create a rule or an axiom so that the object property *influences_* is an inverse of the object property *influenced_by_*, and vice versa.
2. For each node of the BN, do the following:
 - a. If the node is not yet in the ontology:
 - a.1. Get the $T \leftarrow \text{template class}$ selected from one of the nodes in the BN or a default class named *Generic* as a subclass of the owl class: *Thing* (<http://www.w3.org/2002/07/owl#Thing>). The selection rule of T as a template class is explained in the *OntoGetFirstTemplateClass* function.
 - a.2. Create an individual with the same name as the node as an instance of class T;
 - a.3. Create a data property named *hasProbValueOf* for the individual, and set its data property value equal to the node's probability value.
 - b. If the node already exists in the ontology but does not have a *hasProbValueOf* data property, then do the following:
 - b.1. Create a data property named *hasProbValueOf* for the individual and set its data property value equal to the node's probability value.

3. For each node of the BN, do the following:
 - a. Get the individual that matches the node.
 - b. For each parent of the individual:
 - b.1. If the parent does not have an object property named *influences_*, then add the *influences_* object property from the parent to the individual.
 - c. For each child of the individual:
 - c.1. If the child does not have an object property named *influenced_by_*, then add the *influenced_by_* object property from the child to the individual.
4. Finish. The BN has been merged into the ontology.

Below is the intended algorithm based on the steps described above (Algorithm 1).

A description of the functions used in MergeBN algorithm above is given in Table 1.

Table 1. Description of the functions used in the MergeBN algorithm.

Function Name	Description
BNLoad(<i>BNSourceFile</i>)	Load the <i>BNSourceFile</i> into probabilistic knowledge base.
GetAllNodes(<i>BN</i>)	Get all nodes contained in the probabilistic knowledge base <i>BN</i> .
SaveOntology()	Save the resulting ontology in OWL/RDF file format.
AssertInverseObjectProperties(<i>ObjPropInfluences</i> , <i>ObjPropInfluencedBy</i>)	Ensure that the ontology contains object properties with the names <i>ObjPropInfluences</i> and <i>ObjPropInfluencedBy</i> , then create a rule that <i>ObjPropInfluences</i> is an inverse of <i>ObjPropInfluencedBy</i> (and vice versa).
CreateNewDataProperty(<i>DataPropHasProbValue</i>)	Create a new data property named <i>DataPropHasProbValue</i> in the ontology.
CreateNewClassInstance(<i>node</i>)	Create a new individual named <i>node</i> .
GetNodeProbValue(<i>node</i>)	Get the probability value of the <i>node</i> .
AssertNewDataProperty(<i>ind</i> , <i>DataPropHasProbValue</i> , <i>prob</i>)	Ensure that the ontology contains a data property with the name <i>DataPropHasProbValue</i> and that the individual <i>ind</i> has the data property with the value <i>prob</i> .
GetClassInstance(<i>node</i>)	Get a reference to the individual <i>node</i> .
GetAllParents(<i>node</i>)	Get all nodes that are parents of the <i>node</i> .
AssertNewObjectProperty(<i>ind</i> , <i>ObjPropInfluences</i> , <i>nod</i>)	Ensure that the Ontology contains an object property with the name <i>ObjPropInfluences</i> and that the individual <i>ind</i> has the object property in relation to individual <i>nod</i> .
GetAllChildren(<i>node</i>)	Get all nodes that are children of the <i>node</i> .

The *OntoGetFirstTemplateClass* function works by comparing the nodes in the BN that are most similar (i.e., has the same set of parents and children) to the node being investigated in the ontology. This assumption is the most likely to find a template class of a node in the BN that its instance to be created in the ontology. A template class is required to create an individual because, in an ontology, an individual must be an instance of a class. If no node is similar to the node being investigated, the *OntoGetFirstTemplateClass* function returns a *Generic* class, which is a subclass of *owl:Thing* by default. The maximum complexity of the *OntoGetFirstTemplateClass* algorithm is $O(n^2)$, where n is the number of nodes in the BN document.

Algorithm 1. MergeBN: Merging a Bayesian network into an Ontology**Input***OntoSourceFile*: source file of the ontology*BNContextName*: context name of the BN*BNFileName*: file name of the BN**Output***OntoOutputFile*: output file of the ontology that has been merged with the BN**Description**

This algorithm merges a Bayesian network with all its beliefs info into an ontology

<pre> 1 begin 2 <i>O</i> ← OntoLoad(<i>OntoSourceFile</i>) 3 <i>BN</i> ← BNLoad(<i>BNSourceFile</i>) 4 <i>ObjPropInfluences</i> ← "influences_" + <i>BNContextName</i> 5 <i>ObjPropInfluencedBy</i> ← "influenced_by_" + <i>BNContextName</i> 6 <i>DataPropHasProbValue</i> ← "hasProbValueOf" + <i>BNContextName</i> 7 if (<i>ObjPropInfluences</i> ∉ properties of <i>O</i>) then 8 AssertInverseObjectProperties(<i>ObjPropInfluences</i>, <i>ObjPropInfluencedBy</i>) 9 else if (<i>ObjPropInfluencedBy</i> ∉ properties of <i>O</i>) then 10 AssertInverseObjectProperties(<i>ObjPropInfluencedBy</i>, <i>ObjPropInfluences</i>) 11 end if 12 if (<i>DataPropHasProbValue</i> ∉ properties of <i>O</i>) then 13 CreateNewDataProperty(<i>DataPropHasProbValue</i>) 14 <i>Nodes</i> ← GetAllNodes(<i>BN</i>) 15 foreach (<i>node</i> in <i>Nodes</i>) do 16 if (<i>node</i> ∉ individuals of <i>O</i>) then 17 <i>T</i> ← OntoGetFirstTemplateClass(<i>node</i>) 18 <i>ind</i> ← CreateNewClassInstance(<i>T</i>) 19 <i>prob</i> ← GetNodeProbValue(<i>node</i>) 20 AssertNewDataProperty(<i>ind</i>, <i>DataPropHasProbValue</i>, <i>prob</i>) 21 else 22 if (<i>DataPropHasProbValue</i> ∉ properties of <i>O</i>) then 23 <i>ind</i> ← GetClassInstance(<i>node</i>) 24 <i>prob</i> ← GetNodeProbValue(<i>node</i>) 25 AssertNewDataProperty(<i>ind</i>, <i>DataPropHasProbValue</i>, <i>prob</i>) 26 end if 27 end if 28 end foreach 29 foreach (<i>node</i> in <i>Nodes</i>) do 30 <i>nod</i> ← GetClassInstance(<i>node</i>) 31 <i>parents</i> ← GetAllParents(<i>node</i>) 32 foreach (<i>par</i> in <i>parents</i>) do 33 <i>ind</i> ← GetClassInstance(<i>par</i>) 34 if (<i>ObjPropInfluences</i> ∉ properties of <i>ind</i>) then 35 AssertNewObjectProperty(<i>ind</i>, <i>ObjPropInfluences</i>, <i>nod</i>) 36 end foreach 37 <i>children</i> ← GetAllChildren(<i>node</i>) 38 foreach (<i>child</i> in <i>children</i>) do 39 <i>ind</i> ← GetClassInstance(<i>child</i>) 40 if (<i>ObjPropInfluencedBy</i> ∉ properties of <i>ind</i>) then 41 AssertNewObjectAndProperty(<i>ind</i>, <i>ObjPropInfluencedBy</i>, <i>nod</i>) 42 end foreach 43 end foreach 44 <i>OntoOutputFile</i> ← SaveOntology() 45 return <i>OntoOutputFile</i> 46 end. </pre>	<div style="border-left: 1px solid black; padding-left: 10px;"> {Step 1} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 2} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 2.a} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 2.b} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 3} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 3.a} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 3.b} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 3.c} </div> <div style="border-left: 1px solid black; padding-left: 10px;"> {Step 4} </div>
--	---

The complexity analysis of the MergeBN algorithm where n is the number of input nodes in the BN document is explained in Table 2.

Table 2. Complexity analysis of the MergeBN algorithm.

Step	Description	Complexity
1	It consists of six linear steps that do not depend on the number of input nodes in the Bayesian network document.	6
2	It consists of 1 linear step plus 1 quadratic polynomial step for the <i>OntoGetFirstTemplateClass</i> algorithm, plus three linear steps (depending on the number of input nodes in the BN document).	$1 + n^2 + 3n$
3	+	It consists of three linear steps, depending on the number of input nodes in the BN document.
	+	It consists of two quadratic polynomial steps, depending on the number of input nodes in the BN document alongside the maximum number of parents + children = n nodes
	+	It consists of one step that does not depend on the number of input nodes in the BN document.

$$\begin{aligned}
 \text{Step 1} + \text{Step 2} + \text{Step 3} &= 6 + 1 + n^2 + 3n + \left(3n + 2 \times \frac{1}{2} \times n^2 + 2 \times \frac{1}{2} \times n^2 \right) + 1 \\
 &= n^2 + n^2 + n^2 + 3n + 3n + 6 + 1 + 1 \\
 &= 3n^2 + 6n + 8
 \end{aligned}$$

So, the complexity of MergeBN algorithm is $O(n^2)$ or a quadratic polynomial.

3.3.2. Merging Ontologies into BNs

We have developed an algorithm, called MergeOnto, to merge the information contained in an ontology into a BN. The following are some of the terms used in this algorithm:

- *Fixed Nodes*: nodes in a BN that are marked for their unchanging probability values during the parameter learning process.
- *Foreign SubNode*: an individual in an ontology that has a parent in both the ontology and a BN, but does not itself exist in the BN.
- *Sibling*: A sibling of individual X means another individual in an ontology that has the same class as individual X.

The steps in the MergeOnto algorithm are as follows:

1. Get all nodes from the BN.
2. Record each BN node in the *Fixed Nodes* list.
3. Get all *Foreign SubNodes* from the ontology.
4. For each $F \leftarrow \text{Foreign SubNode}$, do the following:
 - a. Get $S \leftarrow \text{Siblings of } F$. If S is not empty, then follow the next steps:
 - a.1. Get a $T \leftarrow$ selected template node from one member of S . The selection rule of T as a template node is described in the *BNGetFirstTemplateNode* function.
 - a.2. Clone T as a new node, N , of the BN. The duplication process brings parents and children information from T .
 - a.3. Remove N along with its children from the *Fixed Nodes* list.
 - a.4. Calculate the probability values of N by looking for the center of distribution (e.g., using a moving windows average algorithm) based on the probability value distribution of nodes in S .

- a.5. Generate sample data for nodes not listed in Fixed Nodes (e.g., using partial orders for the structure discovery algorithm).
 - a.6. Perform the parameter learning process based on the sample data and *the Fixed Nodes* list (e.g., using expectation-maximization [EM] technique).
5. Done.

Below is the algorithm's process based on the steps described above (Algorithm 2).

Algorithm 2. MergeOnto: Merging an ontology into a Bayesian network

Input

OntoSourceFile: source file of the ontology

BNSourceFile: source file of the BN

BNDataFile: file name of the BN data sample

BNResultFile: file name of the resulting BN after the merging process

Output

(none)

Description

This algorithm merges an ontology into a BN

Var

FixedNodes: Collection

<pre> 1 Begin 2 $O \leftarrow \text{OntoLoad}(\text{OntoSourceFile})$ 3 $BN \leftarrow \text{BNLoad}(\text{BNSourceFile})$ 4 $\text{Nodes} \leftarrow \text{GetAllNodes}(BN)$ 5 foreach (<i>node</i> in <i>Nodes</i>) do 6 $\text{FixedNodes.Add}(\text{node})$ 7 end foreach 8 $\text{SubNodes} \leftarrow \text{GetAllForeignSubNodes}(O, BN)$ 9 foreach (<i>node</i> in <i>SubNodes</i>) do 10 $S \leftarrow \text{OntoGetSiblings}(\text{node})$ 11 if ($S \neq \epsilon$) then 12 $T \leftarrow \text{BNGetFirstTemplateNode}(\text{node}, S)$ 13 $N \leftarrow \text{BNCloneNode}(T, \text{node}, \text{ref FixedNodes})$ 14 $\text{FixedNodes.Remove}(\text{node})$ 15 $\text{SetCenterOfDistribution}(\text{node}, S)$ 16 $\text{GenSampleDataFile}(\text{node}, \text{BNDataFile})$ 17 $\text{BNLearn}(\text{BNDataFile}, \text{FixedNodes}, \text{BNResultFile})$ 18 end if 19 end foreach 20 end. </pre>	<pre> {Step 1} {Step 2} {Step 3} {Step 4} {Step 4.a} {Step 4.a.1} {Step 4.a.6} </pre>
--	---

Description of the functions used in the MergeOnto algorithm, where m is the number of input nodes in the ontology document and n is the number of input nodes in the BN document, is given in Table 3.

The *BNGetFirstTemplateNode* function works by comparing the individuals in a set of siblings in the ontology that are most similar (i.e. has the same set of parents) to the individual being investigated in the BN. This assumption is the most likely to find the template node of a class in the ontology that its instance to be created in the BN. If no class is similar to the class being investigated, the *BNGetFirstTemplateNode* function returns null and the class being investigated will not be created in the BN. The maximum complexity of the *BNGetFirstTemplateNode* algorithm is $O(n^2)$, where n is the number of nodes in the BN document.

Table 3. Description of the functions used in the MergeOnto algorithm.

Function Name	Description	Complexity
GetAllForeignSubNodes(<i>O</i> , <i>BN</i>)	Get all <i>foreign subnodes</i> of ontology <i>O</i> against Bayesian network <i>BN</i> .	m^2
OntoGetSiblings(<i>node</i>)	Get all siblings of <i>node</i> .	m
BNGetFirstTemplateNode(<i>node</i> , <i>S</i>)	Get the first template node of the <i>node</i> from the siblings of the <i>node</i> (<i>S</i>).	n^2
SetCenterOfDistribution(<i>node</i> , <i>S</i>)	Set maximum likelihood estimation (MLE) values for each probability value of the <i>node</i> based on the siblings list <i>S</i> using moving average or sliding window algorithms.	n^2
GenSampleDataFile(<i>node</i> , <i>BNDataFile</i>)	Generate files named <i>BNDataFile</i> containing sample data with the suspect node <i>node</i> for each node in the BN that is not included in <i>Fixed Nodes</i> .	n^3
BNLearn(<i>BNDataFile</i> , <i>FixedNodes</i> , <i>BNResultFile</i>)	Perform parameter learning based on the generated sample data inside <i>BNDataFile</i> for each node in the BN that is not included in <i>FixedNodes</i> using EM algorithm. The result is stored in <i>BNResultFile</i> .	n^3

The complexity analysis of the MergeOnto algorithm is explained in Table 4. So, the complexity of the MergeOnto algorithm is $O(n^3)$ or a cubic polynomial.

Table 4. Complexity analysis of the MergeBN algorithm.

Step	Description	Complexity
1	It consists of 1 linear step, depending on the number of input nodes in the BN document.	n
2	It consists of 1 linear step, depending on the number of input nodes in the BN document.	n
3	GetAllForeignSubNodes	m^2
4	OntoGetSiblings + BNGetFirstTemplateNode + BNCloneNode + <i>FixedNodes.Remove</i> + SetCenterOfDistribution + GenSampleDataFile + BNLearn	$m + n^2 + 3n + 1 + n^2 + n^3 + n^3$

$$\begin{aligned} \text{Step 1} + \text{Step 2} + \text{Step 3} + \text{Step 4} &= n + n + m^2 + m + n^2 + 3n + 1 + n^2 + n^3 + n^3 \\ &= 2n^3 + m^2 + 2n^2 + m + 5n + 1 \end{aligned}$$

4. Experiment

We have implemented the ByNowLife framework into an application software prototype. The application was developed using C#. It uses OWLAPI for .Net Libraries as an ontology reader and writer, Pellet as a logical reasoner, and Structural Modeling, Inference, and Learning Engine (SMILE) as its probabilistic reasoner. We built a ByNowLife library package that contains three main modules: (1) a module to handle queries and their execution results, (2) a module that controls and coordinates logical and probabilistic reasoning and (3) a Morpher/Transformer module that translates OWL/RDF into XDSL (and vice versa). Examples of cases solved using this technique are given below.

4.1. Case #1: Investment Problem

An investor has shares in the following companies: Mandiri, KPC and Telkomsel. Mandiri is a bank, KPC is a mining company and Telkomsel is a telecommunications company. Each stock has a Price property. However, a government policy affects the fluctuation of stock prices. The latest

condition of the case, as represented in two documents (OWL and XDSL), can be seen in Figures 3 and 4 below.

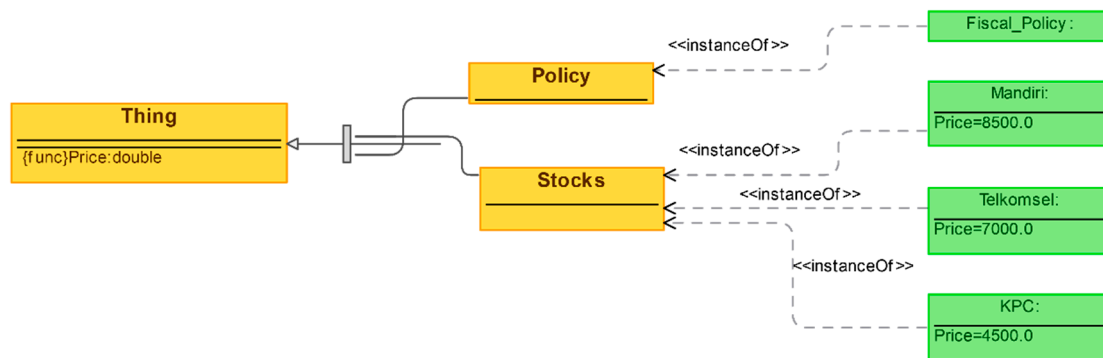


Figure 3. Investment—The structure of classes and individuals in the OWL document.

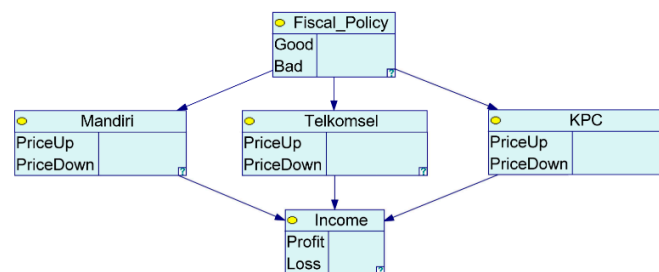


Figure 4. Investment: the relationship between the Fiscal_Policy, Stocks, and Income.

In the XDSL document, there are rules that the Fiscal_Policy influences stock prices and that the fluctuation of stock prices will affect investors' incomes, as shown in Figure 4. This structure was visualized using GeNIe Modeler [23] software.

Suppose that we want to answer the following query: display a stock name that has a price ≥ 5000 and the highest impact on profit of all. In this case, we cannot answer the query by using only one knowledge document nor only one reasoning type. We need both types.

To return stock names with a price ≥ 5000 , we need data from the OWL document, where information about stock prices resides. To determine which of these stocks has the highest impact on profit, we need data from the XDSL document, where rules about the impact of stocks on income resides. The data must be combined to be complete. This is where the Morpher is needed.

The Morpher will create individuals that did not previously exist in the OWL document; in this case, Income is created as a Generic type. The Generic type is a subclass of Thing. The Morpher will then create a special property called *hasProbValue* for every individual in the system. The Reasoner will set probabilistic values in the XDSL document based on the data in the OWL document, perform probabilistic reasoning and then return the result to the OWL document by updating its respective property values. The result of running the program is shown in Figures 5 and 6.

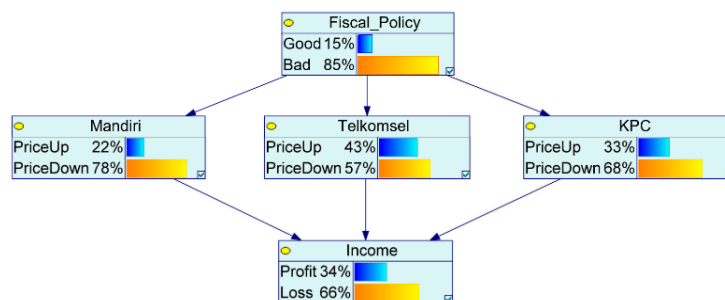


Figure 5. Investment: the result of running the program in the XDSL document.

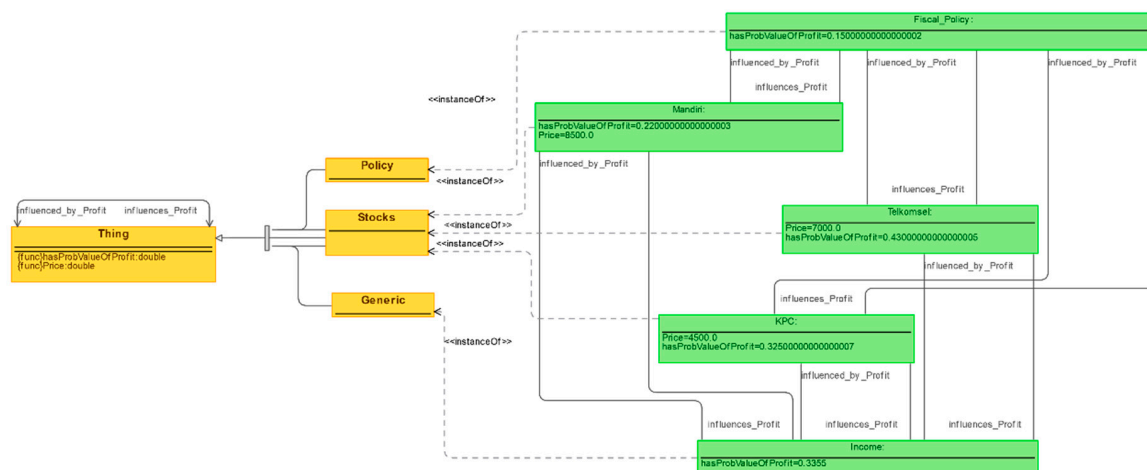


Figure 6. Investment: the result of running the program in the OWL document with updated probability values.

4.2. Case #2: Social Customer Relationship Management (CRM) in Higher Education

This case illustrates the decision-making process in the investment in social CRM development in a higher education institution, namely Bogor Ibn Khaldun University (Universitas Ibn Khaldun Bogor [UIKA]) [24].

Social CRM is CRM combined with the power of social media to generate new approaches in maintaining and enhancing relationships with customers. A social CRM model contains semantic relationships between stakeholders and factors that must be understood thoroughly to identify the roles of each stakeholder and factor in the business process. The stakeholders are the university (including the units inside), prospective students, current students and alumni. Their semantic relationships have consequences for logical reasoning.

The factors are items contained in the critical success factors (CSFs). The CSFs consist of budget and project management, knowledge management, technology selection and adaptation, vendor relationships, information technology (IT) infrastructure, training, culture and leadership. These eight factors determine the successful implementation of social CRM and were identified by Meyliana et al. [25]. This social CRM model examines the cause and effect relationships of CSFs in terms of the successful implementation of social CRM in the university. These causal relationships have consequences in probabilistic reasoning.

UIKA is a university that cares about its customers and pays special attention to its implementation of social CRM. It allocates funds for various activities at annual budget meetings, including those supporting the implementation of social CRM. Departments or units directly involved in the implementation of social CRM at UIKA include the Human Resources (HR) Department, the Public Relations (PR) Department, the Computer and Information Systems (CIS) Unit, the Academic Administration (AA) Bureau, the Administration and Financial Resources (AFR) Bureau and the Quality Assurance (QA) Unit.

Figure 7 shows a simplified model that illustrates the semantic relationships between entities at UIKA. Each department/unit has its own budget for supporting the successful implementation of social CRM. Other information related to CSFs that determines the success of social CRM implementation at UIKA is illustrated in Figure 8.

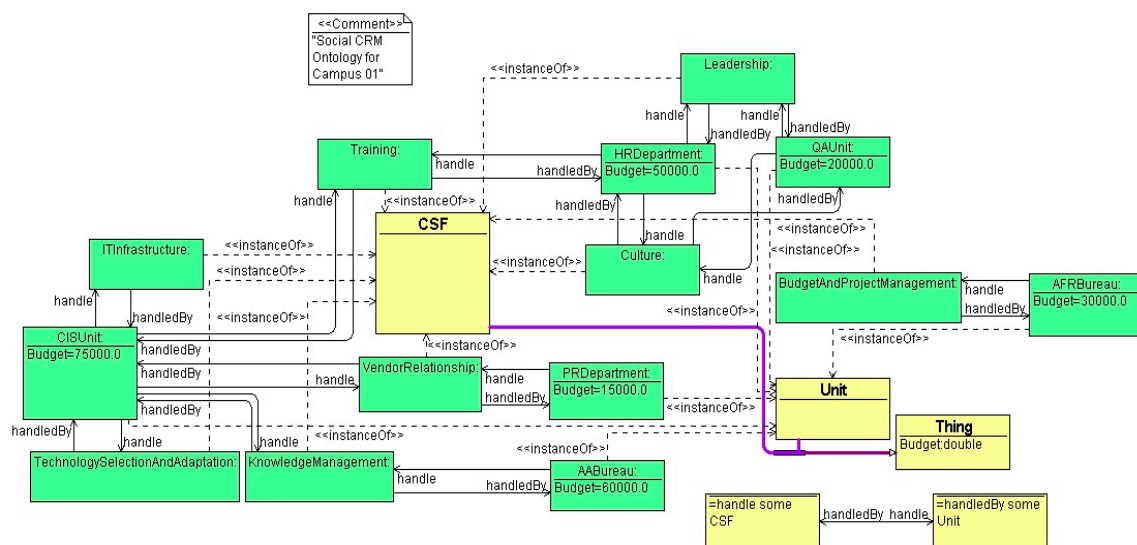


Figure 7. Social customer relationship management (CRM): the structure of classes and individuals in the OWL document.

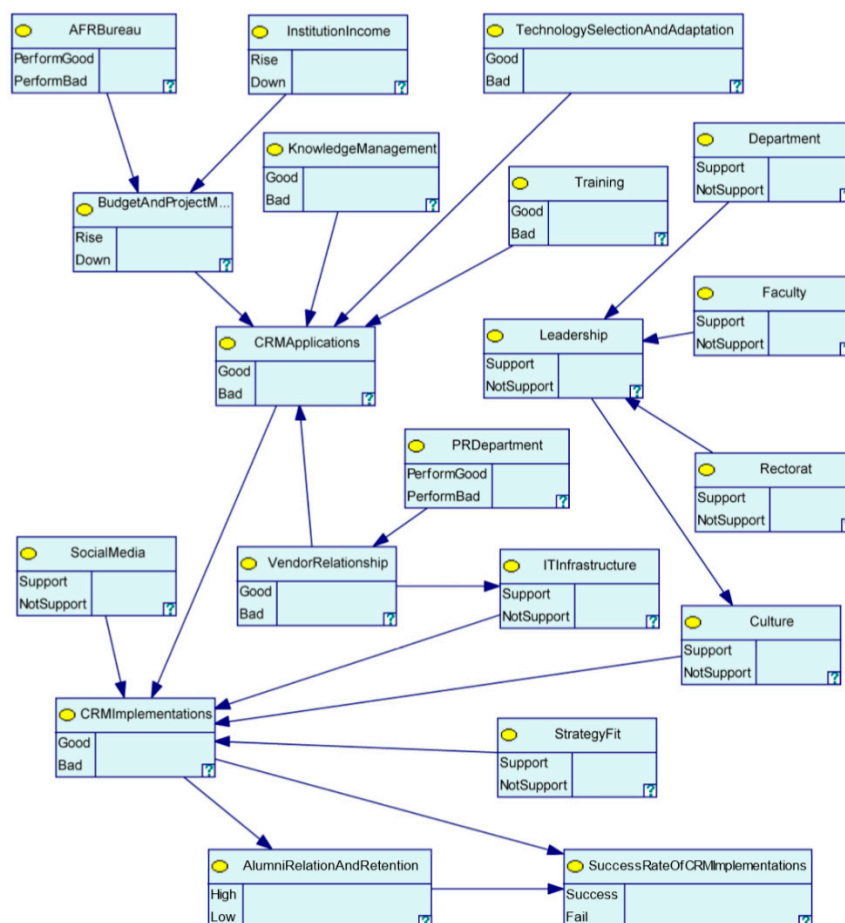


Figure 8. Social CRM: the relationships between the factors.

Suppose that we want to answer the following queries: (1) Display units that handle CSFs and their budget allocations for the successful implementation of social CRM—in this case, we can only query the OWL document for a result; (2) List three factors of CSFs that most influence the successful implementation of social CRM—in this this case, we can only query the XDSL document for a result;

and (3) List three units that handle CSFs and their budget allocations, which are prioritized to address the successful implementation of social CRM—in this case, we need to query both the OWL and XDSL documents simultaneously for a result. An example of SPARQL syntax for query no. 3 is as follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX socialcrm: <http://localhost/latihan/ontologies/socialcrm#>
SELECT ?UnitName ?BudgetAllocation ?CSFVar ?SuccessRate
WHERE {?UnitName rdf:type socialcrm:Unit.
      ?CSFVar rdf:type socialcrm:CSF.
      ?UnitName socialcrm:Budget ?BudgetAllocation.
      ?UnitName socialcrm:handle ?CSFVar.
      ?CSFVar socialcrm:hasProbValueOfSuccessRate ?SuccessRate}
ORDER BY DESC(?SuccessRate)
LIMIT 3
```

The system will generate a knowledge base that combines data from both the OWL and XDSL documents. The query will then be executed against the generated knowledge base to get a result. Examples of results can be seen in Figures 9 and 10 below.

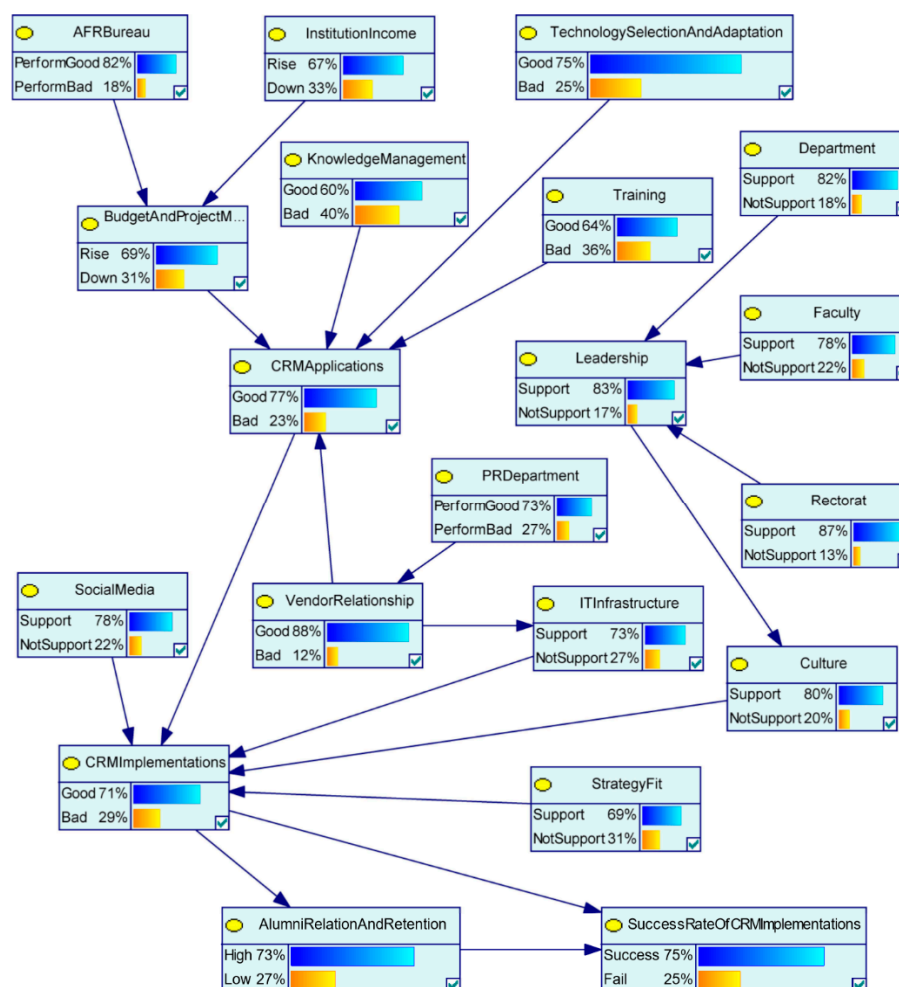


Figure 9. Social CRM: the result of running the program in the XDSL document.

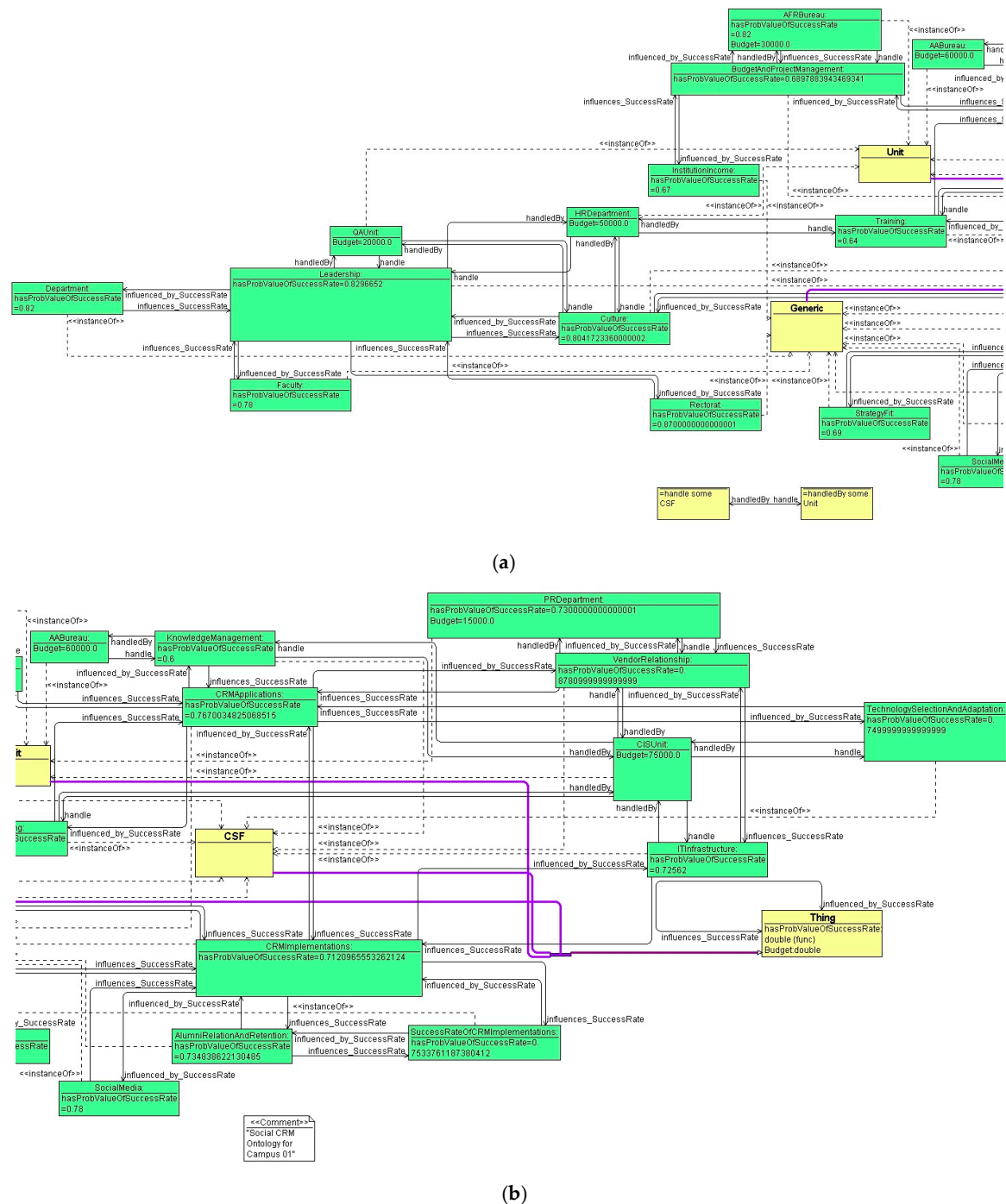


Figure 10. Social CRM: the result of running the program in the OWL document with updated probability values. (a) left part; (b) right part (cont'd).

5. Reasoning Proof

The conventional reasoning query process is done in two steps. The first step is to perform probabilistic reasoning within the BN knowledge base. The second step, based on the reasoning in the first step, is reasoning within the ontology knowledge base; this returns the final result of the reasoning. The reasoning query process on ontology-based systems that utilize ByNowLife is completed in just one step by sending logical and probabilistic reasoning queries in SPARQL format. This is because ByNowLife transforms BNs into ontologies (through the merging process) automatically before reasoning is performed on the ontology knowledge base. It can be proven that the conventional

two-step process of reasoning is a subset of ByNowLife's reasoning process; that is, the reasoning done by ByNowLife encompasses conventional reasoning.

For example, O is an ontology:

$$O = (C_1, I_1, P_1, R_1, X_1) \quad (1)$$

where C are classes or concepts, I are individuals, P are properties attached to a class or individual, R are inter-class or inter-individual relationships and X are axioms. BN is a Bayesian network:

$$BN = (N, V, \rho) \quad (2)$$

Here, N represents a set of nodes, V represents a set of vertices (links) and ρ represents the set of conditional probability values of every node in the BN .

When BN transformed into O , the transformation rules as described in Section 3.2 are applied:

$$N \rightarrow I_2$$

$$V \rightarrow R_2$$

$$\rho \rightarrow P_2$$

The result of the integration between O and BN can then be calculated as follows.

$$O' = O \cup BN \text{ with}$$

$$C' = C_1$$

$$I' = I_1 \cup I_2$$

$$P' = P_1 \cup P_2$$

$$R' = R_1 \cup R_2$$

$$X' = X_1$$

5.1. The Two-Step (conventional) Reasoning Process

The logical reasoning process (\mathcal{R}_O) in O and the probabilistic reasoning process (\mathcal{R}_{BN}) in BN :

$$\mathcal{R}_O(O) \in (C_1, I_1, P_1, R_1, X_1)$$

$$\mathcal{R}_{BN}(BN) \in \{P_2\}$$

Thus, the following can be obtained:

$$\mathcal{R}_{BN}(BN) \cup \mathcal{R}_O(O) \in (C_1, I_1, \{P_1 \cup P_2\}, R_1, X_1) \\ (C_1, I_1, P', R_1, X_1)$$

5.2. The Simultaneous Reasoning Process Used by ByNowLife

The logical reasoning process (\mathcal{R}_O) and the probabilistic reasoning process (\mathcal{R}_{BN}) in O' are

$$\mathcal{R}_O(O') \in (C', I', P', R', X')$$

$$\mathcal{R}_{BN}(O') \in \{P_2\} \subseteq \{P'\}$$

Thus, the following can be obtained:

$$\mathcal{R}_{BN}(O') \cup \mathcal{R}_O(O') \subseteq \mathcal{R}_O(O')$$

Therefore, the integration of probabilistic and logical reasoning in the transformed ontology is a subset of logical reasoning in the ontology. This is because:

$$(C_1, I_1, P', R_1, X_1) \subseteq (C', I', P', R', X')$$

so

$$\mathcal{R}_{BN}(BN) \cup \mathcal{R}_O(O) \subseteq \mathcal{R}_{BN}(O') \cup \mathcal{R}_O(O') \subseteq \mathcal{R}_O(O') \quad (3)$$

It can be proven that probabilistic reasoning in BN followed by logical reasoning in O is a subset of probabilistic and logical reasoning in O' , which is a subset of logical reasoning in O' . Because probabilistic reasoning in BN followed by logical reasoning in O is a subset of probabilistic and logical reasoning in O' , there are cases where queries cannot be answered except by combining these

knowledge bases and querying them. This occurs, for example, when there is an individual that does not exist in the probabilistic knowledge base but exists in the logical knowledge base while the individual is an entity that is taken into account during probabilistic reasoning.

Investment Problem Case—Scenario 2. In Case #1 concerning an Investment Problem (see Section 4.1), there was another individual in the ontology named “Indocement” (as depicted in Figure 11), but in the probabilistic knowledge base, no Indocement node was found (see Figure 4). The resulting reasoning flow from the query in Section 4.1 is shown in Table 5.

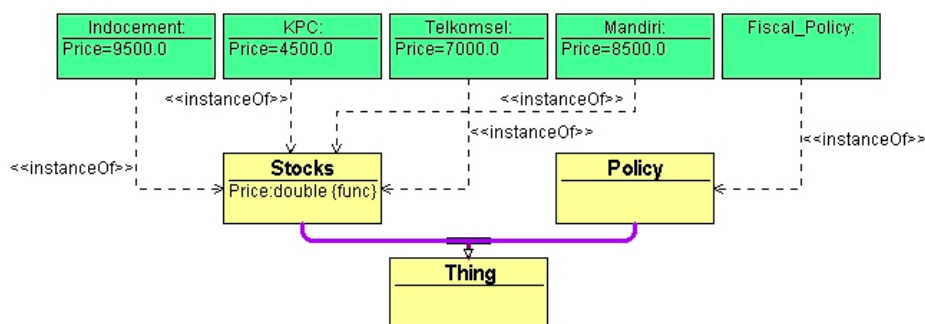


Figure 11. Investment #2: the structure of classes and individuals in the OWL document.

Table 5. Investment #2: determining the query answer using conventional two-step reasoning.

Clause/Terms	Reasoning to	Result
<i>stocks with prices ≥ 5000</i>	OWL	<i>Mandiri, Telkomsel and Indocement</i> <i>Mandiri.Price=8500; Telkomsel.Price=7000;</i> <i>Indocement.Price = 9500</i>
<i>has the highest impact on profit of all</i>	BN	<i>Telkomsel</i> <i>Mandiri.PriceUp: 0.22 < Telkomsel.PriceUp: 0.43</i>

Using the ByNowLife framework, the system will create the Indocement node in a probabilistic knowledge base and then calculate its probability value using a parameter learning algorithm as follows (Figure 12).

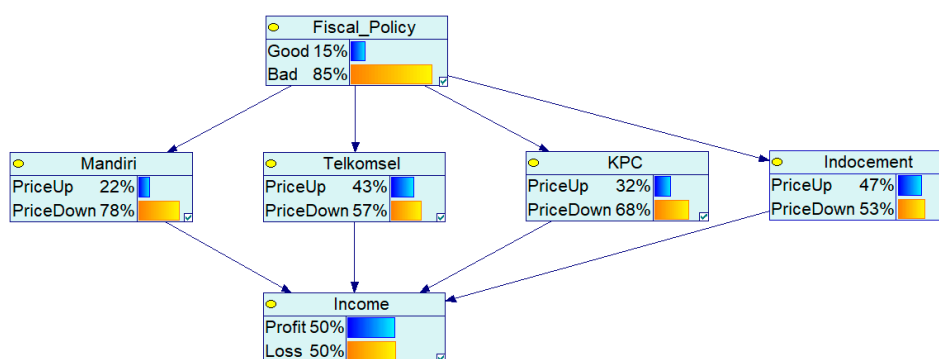


Figure 12. Investment #2—The calculation of probability values using the ByNowLife reasoning technique.

The reasoning flow for the query answer is illustrated in Table 6.

From this case, it can be seen that two-step reasoning provides different results than the simultaneous reasoning of ByNowLife: two-step reasoning returns Telkomsel while simultaneous reasoning by ByNowLife returns Indocement. This is because through two-step reasoning, the BN system cannot calculate probability values for the Indocement node because it does not exist in the BN document.

Table 6. Investment #2—Determining the query answer using the ByNowLife reasoning technique.

Clause/Terms	Result
<i>stocks with prices ≥ 5000</i>	<i>Mandiri, Telkomsel and Indocement</i> <i>Mandiri.Price=8500; Telkomsel.Price=7000;</i> <i>Indocement.Price = 9500</i>
<i>has the highest impact on profit of all</i>	<i>Indocement</i> <i>Mandiri.hasProbValueOfProfit: 0.22 <</i> <i>Telkomsel.hasProbValueOfProfit: 0.43 <</i> <i>Indocement.hasProbValueOfProfit: 0.47</i>

6. Discussion

ByNowLife was developed as a framework using different concepts or paradigms from existing frameworks that integrate BNs with ontologies. Existing integration frameworks only focus on how to represent probabilistic information by using ontology notation and perform reasoning with it. ByNowLife was created not only to represent probabilistic information in ontology notation and perform reasoning with it but also to enhance integration through ontology enrichment and structural and parameter learning mechanisms. With ByNowLife, ontology is enriched with information from probabilistic structures and values in the BN and, conversely, the BN structure is augmented with information contained in the ontology through structural learning. The determination of the probability values of new nodes added to the system and those related to them is accomplished through a parameter learning mechanism.

Related studies on representing probabilistic information in ontological notation and reasoning with it are more focused on the rules of representation than on the transformation of existing BN knowledge bases. Consequently, an assumption arises that the inclusion of probabilistic information in ontologies must be done from the beginning when designing the ontology itself. In practice, however, the domain engineer and the knowledge engineer have roles in preparing the knowledge base. Domain engineers struggle with the concept structure of a domain along with its logical rules, and the knowledge engineer examines causality models of inter-node relationships in a problem domain. The domain engineer's work results in a logical knowledge base (ontology), while the knowledge engineer's work results in a probabilistic knowledge base (BN). A union of the two types of knowledge bases through transforming existing knowledge bases so that the processing time is much faster is a more efficient solution than having to redesign them from the beginning.

We found that a number of related studies have also touched on this concept, even though partially, by transforming ontologies into OOBNs [16,19,20]. However, we did not find the opposite, transformers/morphers that transform BNs into ontological notations. We see that this is possible due to the assumption of previous frameworks that the integration of probabilistic information into ontologies must be done at the beginning of ontology construction, not through the transformation process. One work that discusses topics somewhat similar to the concepts examined within this study was completed by Ishak et al. [19]. Instead of using BNs, this team used OOBNs.

Such related studies discuss the transformation process of an overall ontology knowledge base into the form of a BN or OOBN. ByNowLife has a different approach; it uses the Markov blanket [26] principle. For reasoning in a case, only related nodes are necessary to answer a given query. The transformation of the ontology is only carried out on individuals belonging to the Markov blanket of nodes contained in the BN.

7. Conclusion

The concept of ontology and BN integration was developed based on the need for a framework able to integrate logical and probabilistic reasoning simultaneously to exploit integration. The integration algorithms were developed to integrate BNs into ontologies and vice versa, as shown in Section 3.3. The major benefits obtained from the proposed concept are as follows:

1. Ontology enrichment based on probability values from BNs, as shown in Case #1: the Investment Problem and Case #2: Social CRM in Higher Education;
2. The adjustment of BN structure through structural and parameter learning, as shown in the two cases, which are discussed clearly in Section 4; and
3. The ease of querying a knowledge base with probabilistic clauses in SPARQL format.

The framework and its implementation in prototype form have been demonstrated in the Investment Problem and Social CRM for Higher Education cases. The findings from these scenarios show that this framework efficiently performs both logical and probabilistic reasoning simultaneously.

8. Future Work

In future work, we want to enhance the template class selection technique as implemented in the *BNGetFirstTemplateNode* function. Currently, the function's implementation is only based on the similarity of the direct descendant class to the class being investigated. We must consider the next steps to be taken when the template node is derived from the indirect descendant class or when the function returns several alternative template nodes because, in an ontology, an individual can be derived from several classes at once. The template class selection technique will also become more complex by including not only parental similarities but also the similarities of children in determining the template class.

Author Contributions: Conceptualization, F.A.S.; Data curation, W.C.W.; Investigation, F.A.S.; Methodology, E.K.B.; Software, F.A.S.; Supervision, E.K.B. and W.C.W.; Validation, E.K.B.; Writing—original draft, F.A.S.; Writing—review & editing, E.K.B. and W.C.W.

Funding: This research was funded by Universitas Indonesia under grant no. 1266/UN2.R3.1/HKP.05.00/ 2018.

Acknowledgments: We thank Meyliana from Bina Nusantara University for her material assistance in the construction of the social CRM model for the Social CRM in Higher Education scenario. We also thank the Bogor Ibn Khaldun University for the material assistance provided as a validation case for our framework.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Horrocks, I.; Patel-Schneider, P.F.; McGuinness, D.L.; Welty, C.A. OWL: A Description Logic Based Ontology Language for the Semantic Web. In *The Description Logic Handbook: Theory, Implementation, and Applications*; Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., Eds.; Cambridge University Press: New York, NY, USA, 2007; pp. 1–35.
2. Zhang, X.; Xiao, G.; Lin, Z.; Van Den Bussche, J. Inconsistency-Tolerant Reasoning with OWL DL. *Int. J. Approx. Reason.* **2014**, *55*, 557–584. [[CrossRef](#)]
3. Setiawan, F.A.; Budiardjo, E.K.; Basaruddin, T.; Aminah, S. A Systematic Literature Review on Combining Ontology with Bayesian Network to Support Logical and Probabilistic Reasoning. In *Proceedings of the 2017 International Conference on Software and e-Business—ICSEB 2017*; ACM Press: New York, NY, USA, 2017; pp. 1–12.
4. Apicella, A.; Corazza, A.; Isgro, F.; Vettigli, G. Integration of Context Information through Probabilistic Ontological Knowledge into Image Classification. *Information* **2018**, *9*, 252. [[CrossRef](#)]
5. Ziemia, P.; Wątróbski, J.; Jankowski, J.; Wolski, W. Construction and Restructuring of the Knowledge Repository of Website Evaluation Methods. In *Lecture Notes in Business Information Processing*; Springer International Publishing: Cham, Switzerland, 2016; Volume 243, pp. 29–52. ISBN 9783319305288.
6. Haberin, R.; da Costa, P.C.G.; Laskey, K.B. Probabilistic Ontology Architecture for a Terrorist Identification Decision Support System. In *19th International Command and Control Research and Technology Symposium, 16–19 June 2014*; Command and Control Research Program (U.S.): Alexandria, VA, USA; pp. 1–30.
7. OWL Web Ontology Language Guide. Available online: <https://www.w3.org/TR/owl-guide/#OwlVarieties> (accessed on 9 December 2018).

8. Setiawan, F.A.; Wibowo, W.C.; Ginting, N.B. Handling Uncertainty in Ontology Construction Based on Bayesian Approaches: A Comparative Study. In *4th International Conference on Soft Computing, Intelligent Systems, and Information Technology*; Intan, R., Ed.; Springer International Publishing: Cham, Switzerland, 2015; Volume 516, pp. 539–550.
9. Zhang, S.; Sun, Y.; Peng, Y.; Wang, X. BayesOWL: A Prototype System for Uncertainty in Semantic Web. In *Proceedings of the 2009 International Conference on Artificial Intelligence, 13–16 July*; IC-AI: Las Vegas, NV, USA, 2009; pp. 678–684.
10. Ding, Z.; Peng, Y. A Bayesian Approach to Uncertainty Modelling in OWL Ontology. In *Proceedings of the International Conference on Advances in Intelligent Systems -Theory and Applications*, Centre de Recherche Public Henri Tudor, Luxembourg-Kirchberg, Luxembourg, 15–18 November; Volume 10, pp. 1–9.
11. Carvalho, R.N.; Costa, P.C.G.; Laskey, K.B.; Chang, K.C. PROGNOS: Predictive situational awareness with probabilistic ontologies. In *Proceedings of the 2010 13th International Conference on Information Fusion*, Edinburgh, UK, 26–29 July 2010; pp. 1–8.
12. Boruah, A.; Hazarika, S.M. An MEBN Framework as a dynamic firewall’s knowledge flow architecture. In *Proceedings of the International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, India, 20–21 February 2014; pp. 249–254.
13. Laskey, K.B.; Costa, P.C.G.; Janssen, T. Probabilistic ontologies for Knowledge Fusion. *Front. Artif. Intell. Appl.* **2010**, *213*, 147–161. [[CrossRef](#)]
14. Laskey, K.B. MEBN: A language for first-order Bayesian knowledge bases. *Artif. Intell.* **2008**, *172*, 140–178. [[CrossRef](#)]
15. Santos, L.L.; Carvalho, R.N.; Ladeira, M.; Weigang, L. A new algorithm for Generating Situation-Specific Bayesian Networks Using Bayes-Ball Method. In *CEUR Workshop Proceedings*; CEUR-WS: Kobe, Japan, 2016; Volume 1665, pp. 36–48.
16. Yang, Y. A Framework for Decision Support Systems Adapted to Uncertain Knowledge. Ph.D. Thesis, der Universität at Fridericiana zu Karlsruhe (TH), Karlsruhe, Germany, 2007.
17. Mohammed, A.W.; Xu, Y.; Liu, M. Knowledge-oriented semantics modelling towards uncertainty reasoning. *Springerplus* **2016**, *5*, 1–27. [[CrossRef](#)] [[PubMed](#)]
18. Fenz, S.; Tjoa, A.M.; Hudec, M. Ontology-based generation of bayesian networks. In *Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2009*; IEEE: Fukuoka, Japan, 2009; pp. 712–717.
19. Ishak, M.B.; Leray, P.; Amor, N. Ben Ontology-based generation of Object Oriented Bayesian Networks. In *CEUR Workshop Proceedings*; CEUR-WS: Barcelona, Spain, 2011; Volume 818, pp. 9–17.
20. BayesFusion Appendices: XDSL File Format—XML Schema Definitions. Available online: https://dslpitt.org/genie/download/xdsl_schema.zip (accessed on 30 March 2017).
21. ByNowLife—Bayesian Network and OWL Integration Framework. Available online: <http://www.bynowlife.net> (accessed on 9 December 2018).
22. Pileggi, S.F. An Individual-Centric Probabilistic Extension for OWL: Modelling the Uncertainty. In *Procedia—Procedia Computer Science*; Elsevier Masson SAS: Issy les Moulineaux, France, 2015; Volume 51, pp. 1742–1751.
23. BayesFusion Downloads for Academia. Available online: <https://download.bayesfusion.com/files.html?category=Academia> (accessed on 9 December 2018).
24. Universitas Ibn Khaldun Bogor. Available online: <http://www.uika-bogor.ac.id> (accessed on 9 December 2018).
25. Meyliana; Hidayanto, A.N.; Budiardjo, E.K. The Critical Success Factors for Customer Relationship Management Implementation: a Systematic Literature Review. *Int. J. Bus. Inf. Syst.* **2016**, *23*, 131–174. [[CrossRef](#)]
26. Pearl, J. *Markov Networks. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1988; pp. 96–104. ISBN 0934613737.

