Modern Applied Science; Vol. 12, No. 3; 2018
ISSN 1913-1844E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Computer Hardware Components Ontology

Ahmad K. AL Hwaitat[1], Ameen Shaheen[1], Khalid Adhim[1], Enad N. Arkebat[1] & Aezz Aldain AL Hwiatat[1]

[1] Dept. Computer Science, King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan

Correspondence: Ameen Shaheen, School King Abdullah II School for Information Technology, The University of Jordan, Amman, Jordan. E-mail: Aminalshahin@gmail.com

Received: January 6, 2018 Accepted: January 18, 2018 Online Published: February 27, 2018

doi:10.5539/mas.v12n3p35 URL: https://doi.org/10.5539/mas.v12n3p35

Abstract

A computer system consists of Hardware components that integrate with each other .The purpose of this paper is to create the hardware components of a computer system by formalizing a number of concepts that represent the knowledge of this dolman. Description logic anddefensible logic are used in this paper to achieve our goal.

Keyword: central processing unit (CPU), Memory, input/output devices(I/O) devices

1. Introduction

Ontology is a body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them (Genesereth & Nilsson, 1987). Ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where Ontology is a systematic account of Existence. For AI systems, what "exists" is that which can be represented. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such ontology, definitions associate the names of entities in the universe of discourse (e.g., classes, relations, functions, or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, ontology is the statement of a logical theory. (Gruber, 1993; Fridman & Hafner, 1997).Computer Hardware Components has many parts that connected to each other's to make a full computer.In Computer Hardware Components ontology we will formal explicit description of concepts in a Computer Hardware Components of discourse (classes […]), properties of each concept […] (slots […]), and restrictions on slots (facets […])".The objective from this paper is to support the sharing and reuse of formally represented knowledge among others related problems.

2. Ontology of Computer Hardware Components:

By using the top-down method we divided the concepts of the CHC (**Computer Hardware Components)** into three parts:

A) CPU

B) Memory

C)I/O devices

The classes above have a subclasses and a set of instances for example: Memory class is divided to two subclasses which are: Main and Secondary Memory. Figure (1) is a hierarchical graph of the ontology Computer Hardware Components concepts. Which the rectangles are represent the concepts and edges are represent the relations.

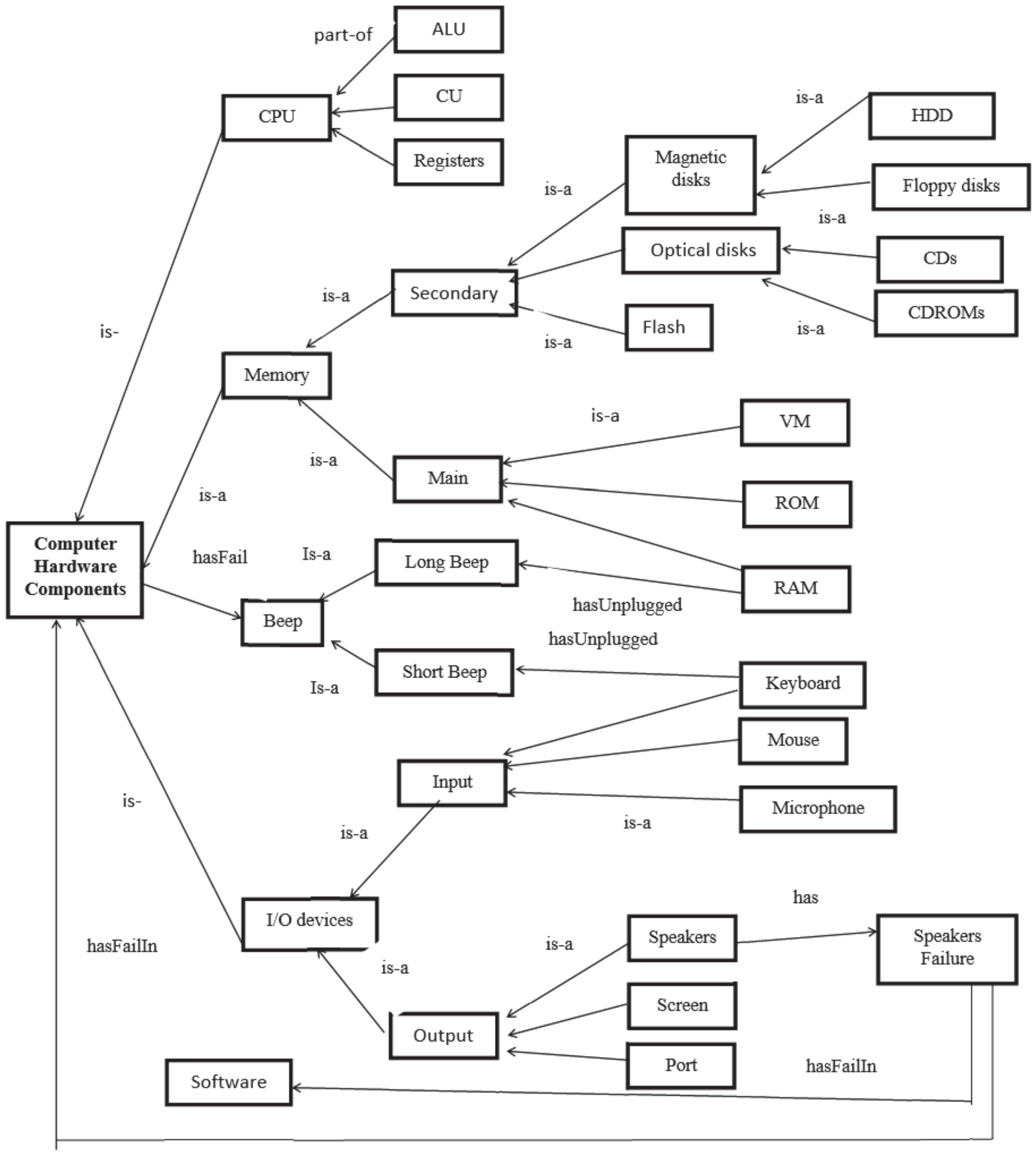

Figure 1. Hierarchical graph of the ontology

2.1 Description Logics and Its Representation

Description Logics DL is a family of formalisms based in first-order +logic for knowledge representation; they are considered to be the most important formalism for the representation of knowledge, unifying and providing logic base for traditional systems in this area (frames, semantic networks, object-oriented representations, semantic data models and systems of types). Description Logics is used to design systems that have a language to define a KB (Knowledge Base), and tools to make inferences on the basis set (Baader et al., 2003). With Description Logics knowledge representation takes place through functional approach, i.e., precise specifications are provided of functionalities to be given by knowledge base and of the inferences to be made (Paulo & Karina, 2011). We will describe one of the most important and influential description logics, called ALC. Other description logics are best understood as restrictions or extensions of ALC.

2.2 The Description Logic ALC

A description logic theory consists of statements about concepts, individuals, and their relations. Individuals correspond to constants in first-order logic, and concepts correspond to unary predicates. Concepts can be named concepts or anonymous (composite) concepts. Named concepts consist simply of a name, say “human”, which will be mapped to a unary predicate in first-order logic. Composite concepts are formed from named concepts by use of concept constructors, similar to the formation of complex formulas out of atomic formulas in first order logic. In ALC, we have the Boolean constructors:

- conjunction $\sqcap$, which is binary,
- disjunction $\sqcup$, which is binary
- negation $\neg$, which is unary

Hence, if C and D are concepts, then C $\sqcap$ D, C $\sqcup$ D, and $\neg$C are also concepts.

ALC statements relate named or anonymous concepts by means of one of the following:

- inclusion $\sqsubseteq$,
- inverse inclusion $\sqsupseteq$
- equivalence $\equiv$

Also, ALC contains a Special concepts and which are:

- $\top$ (aka top, Thing, most general concept)
- $\bot$ (aka bottom, Nothing, inconsistent concept)

They used as abbreviations for:

- (A $\sqcup$ $\neg$ A) for any concept A
- (A $\sqcap$ $\neg$ A) for any concept A

The Role expressions in DL is^{-}, an example Loves^{-}which mean Loves(Y, X) that mean Y is love X and the secondrole is o ,example, hasParent$\circ$hasBrother , which mean X has parent Z and Z has brother Y.

A Knowledge Base (KB) in Dlis just a TBox plus an Abox and which often written as K = {T, A}. theTbox is a set of axioms like $\sqcap$,$\sqcup$,$\equiv$,$\sqsubseteq$, $\sqsupseteq$ and theAbox is a set of facts like, (HappyParent(John).

3. Ontology of Computer Hardware Components by Description Logics

As we see in the previse section, the knowledge base in DL are consist of Abox and Tbox, inthis section we will write a part of the Computer Hardware Components by using the Description Logics as graph 2:

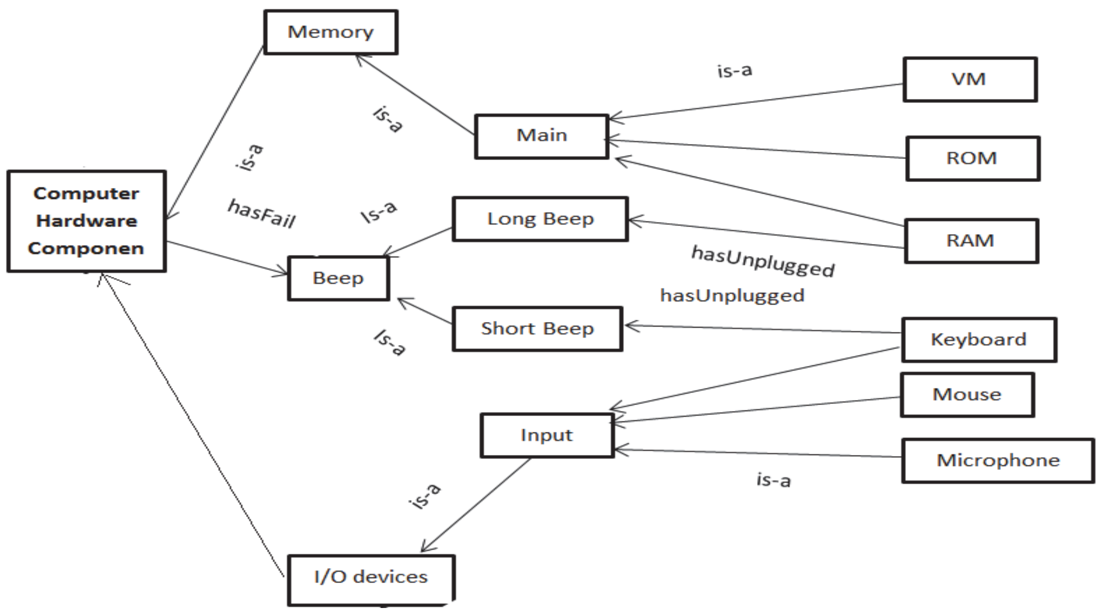

Figure 2. Part of ontology hierarchical graph in DL

TBox :

$$Beep \equiv \exists hasFailure.CHC \quad (1)$$

$$LongBeep \sqsubseteq Beep \quad (2)$$

$$ShortBeep \sqsubseteq Beep \quad (3)$$

$$LongBeeb \equiv \exists hasUnplugged.Ram \quad (4)$$

$$ShortBeeb \equiv \exists hasUnplugged.Keyboard \quad (5)$$

The Axioms from the TBox in natural language is:

(1) The Beep is a Failure in the Hardware components.

(2) Long Beep is a Beep.

(3) short Beep is a type of Beep

(4) The Long Beep is appearing when the Ram slot is unplugged correctly.

(5) Short Beep it is appearing when the Keyboard is unplugged correctly in its port.

From Tbox we can make automaticreasoning suchas:

From 1 &4 we could derive If thereis a Beep in the Hardware this is because of the Ram isUnplugged:

$\exists hasBeep.HCH \equiv hasUnplugged.Ram$

From 1 & 5 could derive the Beep in the hardware can be appear because of the Keyboard is unplugged correctly:

$\exists hasBeep.HCH \equiv hasUnplugged.Keyboard$

And from 1 & 4 & 5 we could derive when the Hardware has aBeep this is because there isunpluggedin the Ram or the Keyboard:

$\exists hasBeep.CHC \sqsubseteq hasUnplugged.Ram \sqcup hasUnplugged.Keyboard$

4. Defensible Reasoning

Classical reasoning for logic-based KR (Knowledge Representation) systems is in general, monotonic. That is, there is an assumption in these systems that there is complete information about a domain. This means that they generally cannot deal with any new information arising which contradicts with the current information. This is not an appropriate model for reasoning in many applications. Therefore, alternative non-monotonic systems have been investigated which can reason under uncertainty or with incomplete information. Defeasible reasoning is one particular model for implementing non-monotonic reasoning. It is concerned with representing and reasoning with defeasible (nonstrict) facts about a domain. The defeasible counterpart of the strict fact: “All birds fly” is the defeasible fact: “Most birds fly” (or the alternative phrasing “Birds usually fly”)(Moodley, Meyer & Varzinczak, 2012).

In Computer Hardware Components there is some non-monotonic reasoning that can’t describe in DL so we need to use the Defeasible reasoning to work with it like:

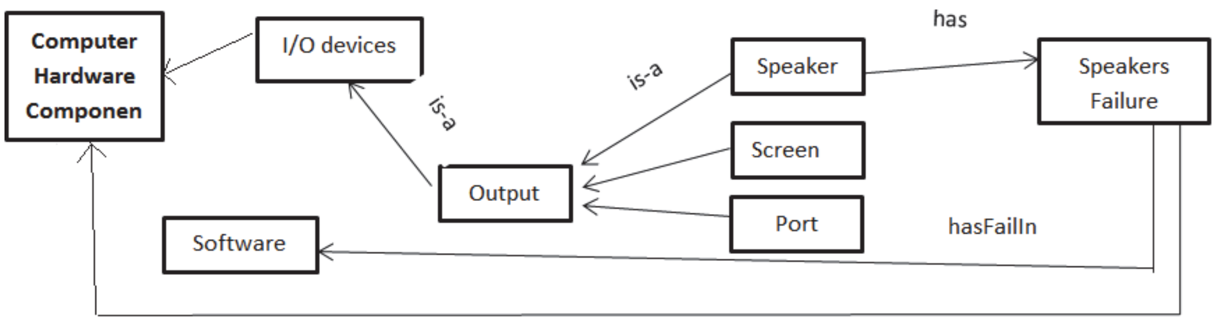

Figure 3. Part of ontology hierarchical graph in Defeasible Logic

The Speakers Failure is a Hardware Failure but it’s also can appear from a software Failure so here we have some exaptation and itsnon-monotonic, we can’t say($SpeakersFailure \sqsubseteq \exists hasFailure.HW$) because of its also can be from the Software but we can writeSpeakers $\sqsubseteq$ Output because of the Speakers is a output device .

In (Moodley, Meyer & Varzinczak, 2012) the authors proposed algorithms to converts the Defeasible reasoning to Classical reasoning to representing in DL .they propose a defeasible subsumptionoperator ($\sqsubset\!\!\!\sim$) which mean a 'supraclassical' to the classical subsumption operator ($\sqsubseteq$) in DLs, example: C $\sqsubset\!\!\!\sim$ D where C and D may be complex ALC concepts and this will solve the issue because of this is mean the most typical C's are also D's.

By this operator we can write (SpeakersFailure $\sqsubset\!\!\!\sim$ $\exists$hasFailure.HW), and (SpeakersFailure $\sqsubset\!\!\!\sim$ $\exists$hasFailure.SW).

The algorithm begins by performing a classical transformation of the input KB. Essentially this amounts to rewriting all defeasible statements in the KB as their classical counterparts and all classical statements into a specific normal form. In our Ontology Let K be the input of transformKB

$$K = \begin{cases} \text{SpeakersFailure} \sqsubset\!\!\!\sim \exists\, \text{hasFailure.HW} \\ \text{SpeakersFailure} \sqsubset\!\!\!\sim \exists\, \text{hasFailure.SW} \\ \text{Speakers} \sqsubseteq \text{H.W} \\ \text{Speakers} \sqsubseteq \neg \text{S.W} \end{cases}$$

If we execute the procedure for K we get K$\sqsubseteq$:

$$K^{\sqsubseteq} = \begin{cases} \text{Speakers} \prod \neg\, \text{H.W} \equiv \perp \\ \text{Speakers} \prod \text{S.W} \equiv \perp \\ \text{Speakers Failure} \sqsubseteq \exists\, \text{hasFailure.HW} \\ \text{Speakers Failure} \sqsubseteq \exists\, \text{hasFailure.SW} \end{cases}$$

This is done by the first definition(transformKB) by the author of(Moodley, Meyer & Varzinczak, 2012)and the second definition which is(Ranking):Let K $\sqsubseteq$ be the classical counterpart of some defeasible KB: A ranking for K$\sqsubseteq$ is a total preorder on the elements (axioms) in K $\sqsubseteq$, with axioms higher up in the ordering interpreted as having a higher exceptionality or importance.

Finally, by 3ed definition (Exceptionality) its specify a sub-procedure called exceptional ($\mathcal{E}$) which computes a more exceptional (specific) subset $\mathcal{E}$of some input set of sentences $\mathcal{E}$.the output of the 3ed definition will be as follow:

$$\mathcal{E}_1 = \begin{cases} \text{Speakers} \prod \neg\, \text{H.W} \equiv \perp \\ \text{Speakers} \prod \text{S.W} \equiv \perp \\ \text{Speakers Failure} \sqsubseteq \exists\, \text{hasFailure.SW} \end{cases}$$

The output from these algorithms will be like:

Dmax = {Speakers $\prod \neg$ H.W $\equiv \perp$

Speakers$\prod$S.W $\equiv \perp$ }

D2 = { SpeakersFailure $\sqsubseteq$ $\exists$hasFailure.SW}

D1 = { SpeakersFailure $\sqsubseteq$ $\exists$hasFailure.HW}

Dmax represents the infinite rank which contains the classical (non-defeasible) statements from the KB.and D2 is represents the more specific of the input that mean the one have a higher ranking and D1 represents the lowest ranking. The output will give apriority for each defeasible statement and the one which have the high priority will look about firstly, the priority are given by the specification ,the concept with more specification it will have a high priority, Which mean the $\exists$hasFailure.SW is more specific than the hasFailure.HW so in DL when we have a SpeakersFailure then it's first will check the SW of the speakers and if the Software are work well then it will check the Hardware of the speakers.

5. Conclusion

In this paper we create the ontology of the computer hardware components and we formalizing a number of concepts by using description logic and defeasible logic. We can use the ontology and the formalization tosupport the sharing and reuse of formally represented knowledge among others related problems.

References

Baader, F., Calvanese, McGuinness, D., Nardi, D., & Patel-Schneider, P. F. (2003). *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press.

Fridman, N., & Hafner, C. D. (1997). The State of the Art in Ontology Design. *AI Magazine,18*(3),53-74.

Genesereth, M. R., & Nilsson, N. J. (1987). Logical Foundations of Artificial Intelligence.Morgan Kaufmann, Los Altos, CA.

Gruber. (1993). Toward principles for the design of ontologies used for knowledge sharing.

Moodley, K., Meyer, T., & Varzinczak, I. J. (2012). A defeasible reasoning approach for description logic ontologies. SA Institute for Computer Scientists and Information Technologies (SAICSIT) Conference, Pretoria, South Africa, 1-3 October 2012.

Paulo, A. L., & Karina, P. (2011). Description Logics as information resource: as example of its application in cardiology. Rio de Janeiro.