A Common-Sense Hardware Accelerated Approach for Context Modeling and Reasoning

Jesús Barba, María J. Santofimia, David Villa, Félix J. Villanueva and Juan C. López
Department of Technology and Information Systems
Computer Engineering School, University of Castilla-La Mancha
Ciudad Real, Spain
Email: {jesus.barba,mariajose.santofimia,david.villa}@uclm.es
Email: {felixjesus.villanueva,juancarlos.lopez}@uclm.es

***Abstract*—Enabling Ambient Intelligence systems to understand the activities that are taking place in their surroundings is a rather complicated task that cannot be successfully addressed if the mechanisms that entitle humans to succeed in this task are ignored or overlooked. In this sense, it is the common-sense knowledge and reasoning capability which characterizes human rationality and cognition. This work is therefore motivated by the conviction that only by encompassing common-sense into the Ambient Intelligence systems could they be entitled to understand the context and to react to it. However, there are some difficulties that need to be resolved before common-sense capabilities can be fully deployed to Ambient Intelligence. Among those issues, response time and efficiency features are two aspects that yet need to be improved. In this endeavor, this work presents a hardware accelerated implementation of a common-sense knowledge-base system.**

***Keywords*-common-sense; context reasoning and understanding; FPGA; hardware-accelerated;**

I. Introduction

The notion of context is at the heart of the Ambient Intelligence paradigm because of its role in narrowing down the meaning of those events that take place in a supervised environment, and in determining the most suitable means to react to those situations. However, the task of modeling and reasoning about context yet remains one of the most challenging topics of Ambient Intelligence.

An analysis of how humans succeed in understanding our surroundings brings into light that rather than specific or expert knowledge, the task of understanding the dynamics of a context involves holding a great deal of knowledge about how the world works, or what traditionally has been referred to as *common-sense* knowledge.

In this respect, in the late 60s McCarthy postulated in [1] that only by endowing programs with common sense could they be able to achieve the pursued intelligence. Doug Lenat in [2] also pointed out to the same direction, highlighting the importance that common-sense knowledge has on dealing and reacting to novel situations. The agreement on the key role that common sense plays in building intelligent systems is one of the axiomatic facts of the approach presented here. Efforts are therefore addressed to enacting common-sense knowledge and reasoning capabilities as a prerequisite to the automation of the cognitive and understanding processes demanded in Ambient Intelligence. Nevertheless, Ambient Intelligence peculiarities poses some serious restrictions on response times upon which the usefulness of the provided responses depend on. In this sense, it is preferable to achieve a "*not-so-good*" response on an appropriate time rather than a complete one that arrives too late to be useful.

In essence, the way how search and inference mechanisms are implemented is what makes the difference between the approaches proposed to date. In this regard, the Scone Knowledge-Base system,which has inspired this work, adopts a marker-passing strategy showing excellent results in minimizing the time employed in search and inference tasks. The fact that the marker-passing mechanism implemented by Scone was ideally devised for massively parallel processing helps in providing very good results regarding the scalability and expressiveness demands.

Despite the good results achieved by Scone, the complexity of the understanding mechanisms, involved in interpreting context events, poses an arising concern for minimizing the latency time. Motivated by this need, this work proposes a hardware implementation of the marker-passing strategy, used in Scone, and intended to minimize response times and also intended to improve the efficiency of the reasoning mechanism.

This rest of this paper is structured as follows. First, in section II a summary of the related work addressing ad-hoc hardware architectures for different computing paradigms in Artificial Intelligence is presented. Special emphasis is made in *common-sense* systems. In section III, the Scone knowledge-base system is introduced in order to provide the reader with the necessary background upon which the proposed hardware implementation is inspired in. Section IV is fully dedicated to the hardware implementation details, although some inner aspects of the Scone approach are also provided. For understanding purposes an incremental approach has been adopted, starting up with a description of the basic architecture and operations that are supported. In section V, performance results are assessed by comparing

the proposed hardware implementation approach with the Scone system, which works as the reference software implementation. Finally, last section is devoted to drawing the conclusions achieved as a result of the proposed approach.

II. Related Work

A context model for Ambient Intelligence comprises the rules that establish how to map sensor data values into high level knowledge. These rules, far from being unique and common to context-aware systems, they tend to be tailored-made solutions that prevent context-aware systems from sharing and leveraging the knowledge they hold. Context models are therefore characterized for their lack of interoperability. The work in [3] provides an appropriate starting point for surveying the existing modeling techniques.

Regarding the semantics that should be also captured in the context model, although primarily devoted to the field of meaning in natural language, the theory of *situation semantics*, proposed by Barwise and Perry [4] has been extrapolated to context-awareness. However, as stated in [5], situations cannot be completely described by propositionally enumerating all the aspects involved in the situation since, aspects such as intuitions about context escape from that modeling strategy.

Sowa proposes in [6] a theory for context modeling, based on conceptual graphs of semantic networks which, as explained underneath has many aspects in common with the approach adopted here. Under this theory, contexts are modeled as propositional containers of additional conceptual graphs. Scone, directly implements the notion of `context` as an effective mean to store and retrieve large amounts of knowledge pieces, by means of a `multiple-context` mechanism. McCarthy's *ist(c, p)* predicate [7], which can be read as "*proposition p is true in context c*" was his attempt to provide an universal mechanism to overcome the large number of arising logic for different reasoning theories.

The meaning of propositions is unavoidable associated to the context in which they are being considered. A plausible way of doing so is by means of the "*possible world*" theory. As stated in [8] there are two possible ways of describing what a *possible world* is. On the one hand it can be described as a set of consistent propositions that are true in a given *world*. On the other hand, a *possible world* can also be explained as an account for how things can be interpreted in a given *world*.

The work presented here adopts an strategy in which contexts are devoted to encompass the knowledge about a certain situation. Inferences and deductions about the information held in that context are therefore the enacted mechanisms so as for the Ambient Intelligence system to understand the situations that are undergoing in the supervised context.

Regarding hardware approaches for fast reasoning and context analysis and evaluation, very little work has lately been done. Implementations of current accelerated AI reasoning platforms mostly rely on distributed grids or many-cores architectures to take advantage of a hypothetical parallelism of the operations to be executed. The bulk of the optimization to speed up execution times stems in efficient data structures implementations and effective load balancing and synchronization mechanism. TROJAN [9] is a well representative of this kind of systems.

It is also worth mentioning the compilation work dine by Delgado-Frias in [10] in which an overview of the very first attempts to implement tailored architectures for semantic networks operations is presented. Additionally, it is also relevant for this work the SNAP system [11], and the PNW Machine [12]. The Connection Machine [13] aims to realize the theoretical artifacts posed in the NETL [14] description; the reference hardware architecture for marker-parsing algorithms. However, all these early approaches are based on massively parallel and fixed configurations that are very difficult to scale since each processing element in the datapath corresponds with a single element in the semantic tree.

III. Scone

The Scone project, leaded by Scott E. Fahlman at Carnegie Mellon University, represents an open-source knowledge-based approach which, in contrast to approaches such as Cyc [15], WordNet [16], or ConceptNet [17], place the focus not at collecting common-sense knowledge but rather at providing the means for supporting common-sense reasoning mechanisms. The Scone system therefore pays special attention to providing an expressive, easy to use, scalable and efficient approach for accomplishing search and inference operations.

The main difference between this and other approaches lies in the way in which search and inference are implemented. As previously stated, Scone adopts a marker-passing algorithm [18] devised to be run in the NETL machine [14]. Despite the fact that these marker-passing algorithms cannot be compared with general theorem-provers, they are indeed faster, and most of the search and inference operations involved in common-sense reasoning are supported: inheritance of properties, roles, and relations in a multiple-inheritance type hierarchy; default reasoning with exceptions; detecting type violations; search based on set intersection; and maintaining multiple, immediately overlapping world-views in the same knowledge base.

One of the main objectives with which Scone was conceived for was to emulate humans' ability to store and retrieve pieces of knowledge, along with matching and adjusting existing knowledge to similar situations. To this end, the multiple-context mechanism implements an effective means to tackle this objective. The multiple-context mechanism also provides an efficient solution by which to

tackle a classical problem of Artificial Intelligence, as it is the *frame problem*.

The great potential of the multiple-context mechanism used by Scone can be better stated by using the example described in [18]. Since "Harry Potter World" is quite similar to the real world, a new context, "HPW", could be created as an instance of the real world[1]. Nevertheless, there are differences between these two contexts, such as the fact that in the "HPW" context a broom is a vehicle. This fact can be easily stated in the "HPW" without affecting real world knowledge, in the same way that knowledge of the real world could be canceled so as to not be considered in the "HPW" context. The way in which Scone handles multiple contexts so as to avoid incongruities problems is by activating one context at a time. By doing this, only the knowledge contained in the active context is considered for the reasoning and inference task.

Unless otherwise stated, the knowledge described in a parent context is inherited by the child context. The context itself is also a node and, like the other the nodes, it stores a set of maker-bits. One of these marker-bits is the context-marker. This bit, when enabled, determines the activation of all the nodes and links that are connected to the active context.

IV. Hardware-Based Reasoning

As it has already been mentioned, the hardware accelerated approach presented here is inspired in the software implementation of the Scone system which, at the same time, implements the marker-passing approach initially devised for the NETL machine. This section starts by providing the reader with a comprehensive overview of the most relevant aspects of the Scone system. Provided the foundations of the NETL and the Scone system, the next subsection is devoted to describing how those have been mapped into hardware implementation decisions.

A. Scone system overview

One of the main endeavors of Scone is to optimize the implicit knowledge management, so that even for the worst case scenario the time spent a in exploring the facts and properties of the knowledge-base remains constant, regardless of the knowledge-base size. This achievement is basically grounded in the architectural approach adopted by the knowledge-base.

The declarative knowledge kept in the Scone knowledge-base system complies with a semantic network built upon two basic concepts, as they are *nodes* and *links units*. Nodes are the abstractions in charge of representing the conceptual knowledge, whereas links are devoted to representing relational knowledge.

[1]In Scone terminology, "general" is the context node that holds knowledge about the real world, and "HPW" would be an individual node, connected by an is-a link to the "general" node.

Additionally, both abstractions are implemented by means of structures in which several bits are dedicated to propagate information during the deduction process. This activation mechanism is implemented by means of a bits activation process or, as referred by the Scone literature, by the so-called marker-passing algorithm.

Due to the key role played by the marking-passing algorithm, understanding how such process is implemented in Scone is essential for the hardware optimization pursued by this work. The work in [18] provides a thorough description of how, by means of sequential activations, markers are used to support the inference and search mechanisms that comprise the deductive activity.

Obviously, the fact that the semantic network is implemented by means of a hierarchical structure also plays a relevant role in determining how the deductive search should behave. In this regard, Scone proposes an efficient mean of managing duplicated knowledge, as it is the *virtual copy* abstraction. The fact that the different levels of the semantic tree also imply different levels of inheritance can complicate and overload the knowledge-base with information that it is already held in it. Scone proposes an efficient solution based on knowledge copies that rather than being physically, they are virtually present. The most relevant implication of this *virtuallity* is the fact that only those nodes that are indeed providing new information about the knowledge already held in the knowledge-base, are being physically created. The way in which the remainder properties are inherited is implemented at the level of the marking-passing mechanisms.

B. Hardware implementation decisions

As mentioned in the previous section, the *marker-parsing algorithm* implemented by Scone is one of its main strengths. The marker-pasing mechanisms is mainly devoted to extracting the implicit knowledge that it is contained in the *semantic network* thorough the complex network of nodes and links that comprise it. Thus, three main challenges arise when facing the hardware implementation of such a kind of knowledge-base systems: (a) to simplified the representation and storage of the semantic network information; (b) to optimize the memory organization for efficient implementations of the marker-parsing algorithm; and (c) to provide a scalabe distributed architecture that simplifies the task of adding new knowledge and that it is capable of parallelizing searches.

Bearing these goals in mind we started the design and implementation of a hardware platform, based on the use of FPGAs (*Field Programmable Gate Arrays*), for fast reasoning under the Scone framework. In figure 1, the reader can find a high level picture of the proposed hardware reasoning platform. The system is composed of the following elements:

- The *Microblaze* soft processor in which a modified version of the software implementation of the Scone's machinery runs.

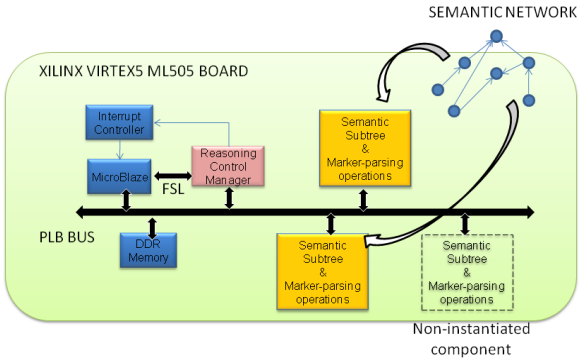

Figure 1. Architecture of the Scone System-on-Chip

- A DDR memory to store a table that relates the names of the entities represented in the semantic network with their identification numbers.
- The *Reasoning Control Manager* (RCM) that is a coprocessor intended for HW/SW interfacing Applications communicates with the RCM using a FIFO like interface, placing commands in its input FSL channel and retrieving the results out of its output FSL channel.
- One or more *Semantic Nodes* (SN) in which the data concerning semantics are mapped into. Each semantic node owns several memories for storage and indexing purposes and a specialized control logic and data path to implement the supported marker-parsing operations.

The device chosen for the implementation of the hardware prototype is a Xilinx XUPV5 board[2]. It is based on a Virtex5 LX110T chip, equivalent to four million logic gates with run-time partial reconfiguration capability. Our *reasoning hardware platform* (RHP) takes advantage of this dynamic feature provided by the FPGA in order to adapt itself to unforeseen scenarios and be scalable. If the situation requires it, a new SN would be instantiated in a free *partial reconfiguration area* (see transparent dotted box in figure 1) in order to expand, for example, the semantic network.

C. Overall System Operation

This section is aimed to present an overview of how the RHP works at system level, before getting into the implementation details of the individual Semantic Nodes. From the user's perspective, the presence of the RHP is totally transparent to him. The Scone's python libraries have been modified in order to forward the requests through the RCM to the SNs in the platform. The way user interacts with Scone have been left without a change.

Internally, requests to the Scone reasoning engine are codified as 32 bit commands which are processed by the RCM once at a time. The RCM issues the corresponding bus transactions and waits for its completion before signaling the software such condition by means of an interruption. Table I summarizes the supported commands and the *instruction format* for each kind of operation.

[2]http://www.xilinx.com/products/boards-and-kits/XUPV5-LX110T.htm

Table I
SUPPORTED COMMANDS BY THE REASONING HARDWARE PLATFORM

Name	**upscan/downscan**	
Description	Mark the *StartingNode* with *MarkerField* and propagates it through *RelationType* in the specified direction	
Field Name	Description	Bits
OPCode	Operation code (00/01)	2
StartingNode	Entity ID the upscan process starts off	20
MarkerField	Boolean mask indicating the indexes to activate	8
RelationType	Type of link	2
Name	**get_nodes**	
Description	Retrieve all nodes that matches the condition specified	
Field Name	Description	Bits
OPCode	Operation code (10)	2
Condition1	*is-set* (1),*is-clear* (0)	1
MarkerField1	Boolean mask indicating the indexes that must satisfy *condition1*	8
BooleanOp	Operation to relate both conditions. 00 *nothing*, 01 *and*, 10 *or*, 11 *and-not*	2
Condition2	*is-set* (1),*is-clear* (0)	1
MarkerField2	Boolean mask indicating the indexes that must satisfy *condition1*	8
Name	**clear_markers**	
Description	Set to zero all marker bits	
Field Name	Description	Bits
OPCode	Operation code (11)	2

In order to control the distributed execution of an operation, each SN core must notify back to the RCM the *initiation* and *finalization* of the local activity. The RCM holds another RAM for this purpose where it scoreboards the start/end notifications. When all the pending work has come to an end, the RCM is ready to pop the next command out of the input FIFO. In the case of a *get_nodes* command, the SN sends to the RCM the list of node identifiers before the *finalization* which are temporarily stored in the output channel FIFO. The software is responsible for reading those values.

D. Optimized Marker Propagation

Upscan and *downscan* operations are the most costly in terms of the time required to be completed. Unlike the *get_nodes* and *clear_markers* commands, which can be triggered in all SNs concurrently, these tree searches lacks of parallelism because of the precedence dependencies inherent to the tree topology. However, it would be desirable to figure out a method to exploit concurrency in the RHP. We propose the use of two simple techniques in order to reduce the overall latency of *upscan/downscan* executions

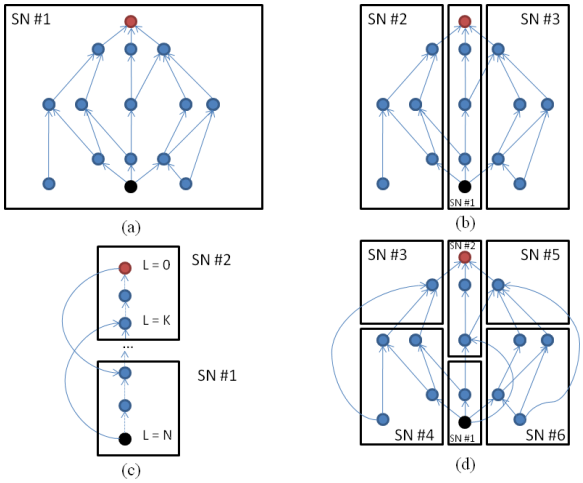

Figure 2. For possible distributions for a semantic tree: (a) single SN, (b) branch partitioning, (c) vertical partitioning with forwards links, and (d) arbitrary combination of both.

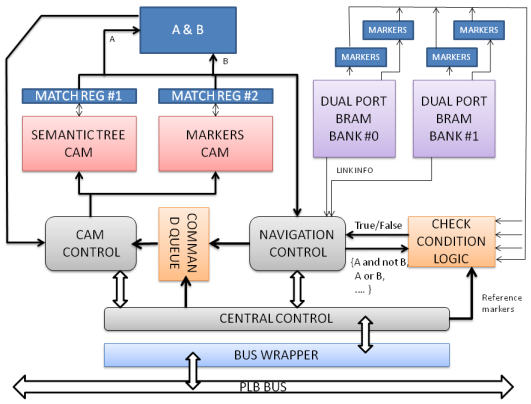

Figure 3. SN internals block diagram

in our platform: (a) horizontal or branch partitioning; and (b) vertical partitioning with *forward links*. These methods are based on a smart spatial distribution of the semantic tree among several SNs. Let us illustrate how the marker propagation process is improved through a series of simple examples.

Figure 2 represents an schematic picture of a simple semantic tree and four possible distribution configurations among one or more SNs. First case (figure 2.a) is the non-optimized version where the whole tree data is stored in a single SNs. Although individual SNs are able to perform marker operations quite efficient due to the specialized hardware architecture (more on this in the next subsection), this configuration prevents from concurrent propagation of marker and searches. Within a SN, tree traveling through the links and nodes is perform serially (see IV-F), in the order specified by the tree traversal algorithm. Thus, the necessary time to complete one of this operations mainly depends on the total number of links that must be visited which is related to the depth and width of the tree.

Now, let us imagine that we start and *upscan* or *downscan* operation from the black and red nodes respectively on the same tree. If the tree is broken down in three parts, taking as a reference the main branches of the tree, and each part is sent to different SNs, the propagation can be done in parallel. Figure 2.b depicts this scenario. The depth of the resulting subtrees is equal to the former tree but their width and, therefore, the number of links to be traversed is reduced. Activation of the operations in all the SNs involved in a search takes place this way:

- First, the RCM requests the initiation of the upscan or downscan commands to all the active SNs.
- Each SN look into its own tree data and determines wether it has to start marker propagation or not. Since a node is only owned exclusively by one SN, only one SN is activated (in our example, *SN #1* for both cases).
- Once the propagation process reaches an *edge link*, the SN requests the *target SN* to initiate the same command. An edge link is a regular link tagged as *external* and for which it is also stored the number of the SN that owns the node at the other end of the link. This way, in figure 2.b after marking the red or black node, two upscan or downscan operations will be triggered on *SN #2* and *SEN #3* that will run in parallel.

Another possible optimization to accelerate marker-parsing algorithms in RHP is the one represented in figure 2.c. In this case, the semantic tree or branch is split *vertically*. This technique reduces the total depth of each resulting partition and, therefore, the time to visit each node. However, just the partitioning would not have the desired effect. In order to early activate a parallel search or propagation, it is necessary to add extra knowledge to the semantic tree in form of *forward links*. A forward link is an external link that captures the transitive property of a given relation.

In both cases (vertical and horizontal partitioning), to reduce the number of internal operations on the local memories, the SN keeps trace of the already visited/traverse node/links for an operation. The SN logic will cut the propagation of any marker that has been set/clear before. This has to be taken into account since one node can be reached through different paths.

Finally, arbitrary combinations of these techniques can be done (see figure 2.d). It is out of the scope of this work the algorithms or approaches that would obtain the best partitioning for a given semantic tree.

E. Architecture of the Semantic Nodes

In this section we cover the internal architecture of the individual Semantic Nodes as hardware cores attached to the system bus. The SN supports the actual execution of the RHP commands (Table I) on the portion of the semantic

tree assigned to it. RHP commands are translate to bus write transactions by the RCM, where the data bits encapsules the 32 bit representation of the operation. Figure 3 shows a high level diagram block of the main components that conform the SN architecture. Following, we proceed to give a brief explanation of each one:

- **Bus wrapper**. It is the adapter to the bus in charge of isolating the core logic of the SN from the implementation details of physical protocol bus. This makes it easy to port component to other bus infrastructures.
- **Central control**. It is the component in charge of the execution management of each operation and synchronization of the other control components.
- **CAM control**. It is responsible for interpreting the actions to be performed on the CAM (*Content Access Memories*) memories: *match* and *write*.
- **Command queue**. It is a FIFO to store the CAM control commands. The *match* command is intended to check whether an input bit pattern matches any content in the CAM. The *write* command modifies the content of the CAM.
- **Content Access Memories**. They are used to index the Block Rams. Due to technological restrictions, the maximum addressable content in these memories is 4096 entries, each. Approximately 20 SNs are needed to store a semantic tree of 10^6 elements (including nodes and relations). There are two types of CAMs in the design:
  - The *Semantic Tree CAM* (STC) is used for navigation purposes. Each entry in this CAM represents a relation between two entities in the semantic tree as it is explained below.
  - The *Markers CAM* (MC) maintains a copy of the marker bits for each relation which applies to both nodes at the ends. This CAM is necessary to implement conditional selections (i.e. all the predecessors of a node that has markers M1 and M2 activated): one pattern is given to the Semantic Tree CAM and the other to the Markers CAM.
- **Match registers**. Temporal registers where are stored the hits found by a CAM. The output of the CAMs is configured not to be coded. Then, *one hot coding* is used instead which simplifies the control process of the operation.
- **Navigation control**. Given a mask of hits, schedules the read/write operations on the block RAMS. Also, it feeds the command queue with new operations. For example, in a downscan operation, the CAMs are used to spot where, in the BRAMs, are the successors of a node. The node numbers are read from the BRAMs and the navigation control inserts new match command in the queue, this time to find the successors of the child nodes.

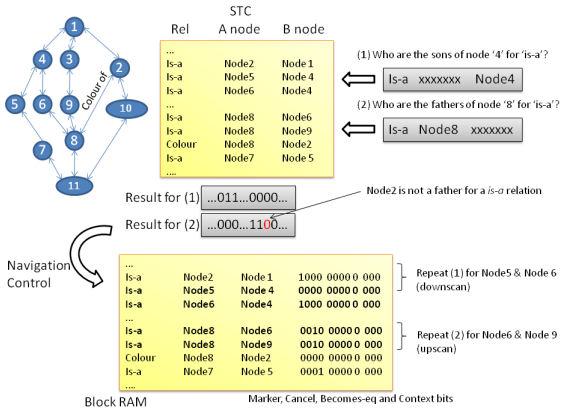

Figure 4. Memory content and basic tree search functionality

- **Dual port block RAMs**. A replica of the Markers and Semantic Tree CAMs is stored in these memories together with status and property bits to control the navigation/propagation process (i.e. the external link flag). The data is interleave among the two memories banks in order to perform up to four memory operations simultaneously.
- **Check condition logic**. It is a combinational configurable (through a select operation port) block that implements the logic comparison between the marker bits in memory and the marker masks in a *get_nodes* command.

F. Basic Reasoning Operations

Upscan and downscan operations are mainly a way to traverse the tree structure in order to activate a set of marker bits. Thus, the way semantic tree relations are represented in the different SN memories matters. Each word in the STC contains the following information for a link of the semantic tree: kind of relation (i.e. is-a, equals,...), A node identifier and B node identifier. In Scone, the A node plays the role of *son* in a relationship while the B node is the *father*. With this simple configuration we can perform fast retrieval operations such as *"find all the predecessors of this node for a relation"* or its complementary with the successors. Putting the right search mask at the input, in one cycle the CAM output register tells which memory addresses satisfy that condition.

In figure 4 the reader can find two simple examples that help to understand the whole process. First, the semantic tree is represented as a collection of links or relations that goes to the STC. The BRAM contains the same data plus the marker, cancelation, becomes-equal and context bits. These bits are interpreted by the navigation control module to determine the next set of nodes to visit in the next iteration. For basic reasoning with no exceptions, only the marker bits are sounded.

In *upscan* and *downscan* operations, a marker is propagated upwards or downwards iteratively through the tree hierarchy following a specific relation type. Marker propagation actually means set to one the corresponding market bit both in the BRAM and the MC. Recall a copy of the marker bits is maintained in the MC so it is possible to select only those nodes that have the proper marker bit activated (logic and between the STC output register and the MC).

As it can be seen in the picture, check for the children (query number one) or parents (query number two) of a node is easy. The STC output feeds the navigation control module which reads the selected memory addresses in order to get the set of node identifiers (field A or B, depending on the operation performed).

G. Advanced Features: Cancelation, Virtual Copies and Contexts

Advanced Scone features have also been considered in this work. Due to the lack of space, only a brief explanation about how exceptional reasoning scenarios have been incorporated is given.

We start this exposition with the *cancelation mechanism*. This mechanism is intended to model exceptions in the norm such as "all birds can fly but penguins can not fly although they are birds". To model this situations, Scone uses special markers (cancelation bits) that block the propagation of another markers. In the hardware implementation, First we look at *cancelation links* for a node before initiating the propagation to the next level. A cancelation link is a regular link but with complementary meaning (for example, a *is-not-a* represents the cancel link for a *is-a* relation). Second, a propagation of the cancel bit starts, which means set to one the cancel bit in the STC and BRAMs memories (this propagation is done using the regular propagation procedure). Third, the former *upscan/downscan* operation can proceed but this time, if the corresponding cancel bit is activated for a link, the navigation control will stop the propagation and will not include that node in the *nodes to be processed* list.

Virtual copies semantics in Scone avoids the unbounded growth of the semantic data when trying to model complex relations. Typically, those relations involved ternary relations to model different roles a node can play in different relations. For instance, one person can play the role "daughter" and "mother" in two distinct families. Without the existence of Scone's virtual copies semantics, the information to model such scenarios should be replicated. Generally speaking, the goal of those ternary representations is to enable reasoning paths where it was impossible before because marker can not cross different relation domains (e.g. if Mary *is-a* Mother in Smith Family and Ashley *is-a* Child in Smith Family and Mothers *love* Children, then Mary *loves* Ashely). The implementation of virtual copies is realized by means of *qualified* links that represent such ternary relations. In hardware, those relations are taken apart in a separate CAM, the *Virtual Copies CAM* (VCC) which was not depicted in figure 3 for the sake of simplicity. The standard propagation mechanism is altered this way:

- If the node participates in any ternary relation, then set to one the *becomes-equal bit* in every node that participates in a ternary relation with the same owner.
- Proceed with the standard propagation procedure.
- Continues propagation for every node with the operation marker bit and the *becomes-equal bit* activated.

To finish this discussion, context modeling in the HRP is sketched. As stated in section III, the potential of Scone's multiple context modeling resides in the fact semantic information can be reused from one context to another just adding the context specific information or canceling facts. Following the HPW example, it is necessary to state a broom *is-a* vehicle and also *is-a* flying thing. That could be done only by adding the necessary links to the real world context. However, such links are only meaningful in the case the HPW is activated. Context activation is handled in the same way Scone proposes. We store a *context* bit mask in the memories, if the right bit is set to one then the link (STC entry) is considered in the reasoning operations. Context activation must be performed beforehand, for such purposes the context-specific semantic data is loaded in the SNs, each entry with an associated context bit already activated. To simplify the reasoning procedures, the entries with all their context bits set to zero are said to belong the "general context" and participates in all the operations.

V. Experimental Results

The evaluation of the HRP presented here needs to be performed in comparison with the software implementation of the Scone system. The set of tests has been design with the following strategy in mind: reproduce similar scenarios than the ones undertaken in [18].

The computer where the software execution took place is a Dell XPS 8300 workstation with 8GB of DDR3 memory, an Intel i7-2600 processor at 3.4 Ghz and a 64-bit GNU/Linux (kernel version 2.6.16) operating system. Due to space constrains, Table II only shows the results for the most relevant tests executed on both the workstation and the FPGA based prototype of the HRP.

The hardware accelerated implementation presented here has paid special attention at improving the ternary relationships that emulate the Scone *virtual copies* abstraction. Further tests need to be carried out in both implementations in which virtual copies plays a key role, so that a better comparison of both implementations can be provided.

VI. Conclusions

This work has been mainly devoted to propose a hardware accelerated implementation of the Scone commonsense knowledge-based system. The ultimate goal of this

Table II
MEAN EXECUTION TIME OF RELEVANT SCONE REASONING OPERATIONS IN THE HRP AND DELL WORKSTATION. 500 EXECUTIONS OF EACH TEST WERE CARRIED OUT.

Test	SW Time	HW Time
Downscan the semantic network tree (1M elements)	4.5 sec	0.77 sec
Check the type of a given individual	0.23 msec	0.04 msec
Mark & intersect 2 sets with 10K members, one winner	20.71 msec	2.88 msec
Mark & intersect 3 sets with 10K members, one winner	36.9 msec	5.59 msec

implementation is to provide a feasible mean upon which context modeling and reasoning tasks can be undertaken at run time.

One of the main challenges that need to be faced by Ambient Intelligence systems is that of understanding the situations that are taken place in the context while at the same time devising the most appropriate response that complies with both the context requirements and the underlying goals. In this endeavor, the work proposed here advocate the benefits of a hardware accelerated approach of a particular common-sense knowledge-based system, enabling these understanding and decision making activities to be undertaken at a reasonable time.

ACKNOWLEDGMENT

This research was supported by the Spanish Ministry of Science and Innovation under the project DAMA (TEC2008-06553/TEC), and by the Spanish Ministry of Industry and the Centre for the Development of Industrial Technology under the project ENERGOS (CEN-20091048), and by Indra under the Cátedra Indra UCLM 2010.

REFERENCES

[1] J. McCarthy, "Programs with common sense," in *Semantic Information Processing*, vol. 1, 1968, pp. 403–418.

[2] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd, "CYC: toward programs with common sense," *Commun. ACM*, vol. 33, no. 8, pp. 30–49, 1990.

[3] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, pp. 161–180, April 2010.

[4] J. Barwise and J. Perry, "Situations and attitudes," *Journal of Philosophy*, vol. 78, no. 11, pp. 668–691, 1981.

[5] J. F. Sowa, "Syntax, semantics, and pragmatics of contexts," in *Proceedings of the Third International Conference on Conceptual Structures: Applications, Implementation and Theory*. London, UK: Springer-Verlag, 1995, pp. 1–15.

[6] J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*. Reading, MA.: Addison-Wesley, 1984.

[7] J. McCarthy, "Notes on formalizing context," in *Proceedings of the 13th international joint conference on Artifical intelligence - Volume 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 555–560.

[8] G. Primiero, "Information and knowledge. a constructive type-theoretical approach." in *Logic, Epistemology, and the Unity of Science*. Berlin: Springer, 2008, vol. 10.

[9] C.-W. Lee, C.-H. Huang, and S. Rajasekaran, "Trojan: a scalable distributed semantic network system," in *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, nov. 2003, pp. 219 – 223.

[10] J. Delgado-Frias and W. R. Moore, "Parallel architectures for ai semantic network processing," *Knowledge-Based Systems*, vol. 1, no. 5, pp. 259 – 265, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0950705188900792

[11] D. I. Moldovan and Y.-W. Tung, "Snap: A vlsi architecture for artificial intelligence processing," *Journal of Parallel and Distributed Computing*, vol. 2, no. 2, pp. 109 – 131, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0743731585900310

[12] P. Sapaty and I. Kočiš, "A parallel network wave machine," in *Proc. of the Third International Workshop on Parallel processing by cellular automata and arrays*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 1987, pp. 267–273. [Online]. Available: http://portal.acm.org/citation.cfm?id=39696.39729

[13] S.-H. Chung and D. I. Moldovan, "Modeling semantic networks on the connection machine," *J. Parallel Distrib. Comput.*, vol. 17, pp. 152–163, January 1993. [Online]. Available: http://portal.acm.org/citation.cfm?id=163506.163524

[14] S. E. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*. Cambridge, MA: MIT Press, 1979.

[15] D. Lenat, "Cyc: A large-scale investment in knowledge infrastructure," *Communications of the ACM*, vol. 38, pp. 33–38, 1995.

[16] C. Fellbaum, *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, C. Fellbaum, Ed. The MIT Press, May 1998.

[17] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu, "Open mind common sense: Knowledge acquisition from the general public," in *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*. London, UK, UK: Springer-Verlag, 2002, pp. 1223–1237.

[18] S. Fahlman, "Marker-passing inference in the scone knowledge-base system," in *First International Conference on Knowledge Science, Engineering and Management (KSEM'06)*. Springer-Verlag (Lecture Notes in AI), 2006.