

# Concept and Sensor Network Approach to Computing: The Lexicon Acquisition Component

Jan Smid<sup>1</sup>, Marek Obitko<sup>2</sup>, Andrej Bencur<sup>3</sup>

<sup>1</sup> SKS Enterprises, Finksburg, MD 21048, USA, [jsmid@sks99.com](mailto:jsmid@sks99.com)

<sup>2</sup> Department of Cybernetics, Czech Technical University, Prague, Czech Republic,  
[obitko@labe.felk.cvut.cz](mailto:obitko@labe.felk.cvut.cz)

<sup>3</sup> Department of Measurement and Control, VSB – Technical University Ostrava,  
Czech Republic, [andrej.bencur@vsb.cz](mailto:andrej.bencur@vsb.cz)

**Abstract.** In this paper, we describe an on-going project called Concept and Sensor Networks (CSN). The development of this project has been described in past PSMP workshops [1]. The purpose of the project is to develop a framework for entities that can process sensor information into concepts. One of the features of this proposed network is the ability of the entities to use language communication to exchange concepts. These entities can potentially represent concepts, knowledge and information using different kinds of semantics. To further this project, we will implement the proposed framework using physical and virtual sensors. In this paper, we overview the key components of the project, primarily focusing on lexical acquisition and the corresponding algorithm.

## 1 Introduction

Concept networks represent knowledge and information. They consist of communicating nodes of different power, which monitor and control physical environments, collect environmental data, infer meaning, build lexicon, and communicate this acquired information. The specific concepts within the network need to be communicated and updated based on information provided by agents and sensors. Some of our efforts in this area are summarized in [2]. Many of the aspects of these networks have been captured during the PSMP (Presentation Sharing, Mining and Protection) workshops [1].

The CSN project is important because it introduces new definitions of networks and addresses the question of how to build a system that possesses the six main features: universal data communication, universal semantic communication, system tolerance, recursive action semantics, lexicon evolution, learning by sharing. To our knowledge, there is no such theory available in the literature.

Two main terms we are dealing with are concepts and sensors. There is no universally accepted definition of a “concept.” Concepts are usually defined respective to an application. We will work with any heuristic definition of the concept that reflects a compact description of the world in the sense expressed by Baum [16].

Examples of such heuristic concepts include rules, programs, a set of regression coefficients, or a subset of object defined by specific attributes. Our working assumption is that complexity does not depend primarily on the complexity of the application domain.

Sensors can be virtual or physical. An example of a physical sensor is a piece of wire that detects temperature. An example of a virtual sensor is a query counter. Our view of sensors and nodes is radically different from the centralized approach in which a few sensors and nodes are directly connected to a powerful computing unit. Sensor networks consist of small, low-cost sensors, typically with wireless connection and with limited capabilities and operating range. One of the virtues of these networks is their ability to accept new nodes. Another benefit is fault tolerance and the distributed processing of the data. These networks can be modeled using the standard multi-agent systems paradigm. The agent can be a simple unit carrying a sensor that reports the sensed value. It can also be a more powerful unit that is able to analyze data from multiple sensors.

A critical issue for sensor networks is the communication between nodes. There are two aspect of communication: data transport and information semantics. The routing protocols in a dynamically changing topology are already available, for example in Intel-Berkeley notes [5], also standard sensor networks are summarized by Akyildiz [3]. On the semantic side, the current approaches require pre-programmed and fixed way of exchanging information. We believe that in order to ensure truly autonomous and fault-tolerant systems, the agents must be able to acquire knowledge and skills from other agents and humans.

The rest of this paper is organized as follows: first, we summarize the overall project goals and principles. We then discuss entities used in sensor networks and discuss the hardware layer. The core of the paper is the explanation of the lexicon acquisition and its advantages. Here we start with explaining our view of the role of the language and discuss its usage, particularly the correspondence between utterances and semantics, together with inference. Then we describe in detail our algorithmic layer of lexicon acquisition, including applications and implementations. We conclude with the overview of the future work.

## **2 Overall Project Goals and Principles**

Our project consists of four layers - cognitive models, message transport models and implementations, hardware and software embedding, and applications. The long-term plan follows several important steps that have been discussed under the umbrella of the PSMP workshops [1].

### **2.2 System Main Principles**

The main system principles are:

1. Universal data communication

A number of protocols can be used to transmit data. The most common protocol for basic data transmission is the serial protocol. This protocol is often inconvenient because it usually based on an application and the designer of the application. We use wireless connections in our networks. Hypertext Transfer Protocol (HTTP) of the Internet stack is more appropriate for our purposes because it satisfies the universality requirement. Most agents and human users are able process the data when we use HTTP.

## 2. Universal semantic communication

Even if distributed systems are built around different knowledge representations, they should be able to incorporate new information from different agents and share it. The information communicated should have in general two components, the description (utterance), and the semantics. The interpretation of utterances is a notoriously hard problem for standard syntactically based linguistics. Language with emphasized semantic components seems to be closer to methods that humans use well.

## 3. System tolerance

A flexible system needs to be able to incorporate new members and remove existing members. We will refer to this feature as system tolerance.

## 4. Recursive action semantics

In general, most researchers use semantics to refer to the meaning of an utterance. We add to this definition by incorporating the ability to perform actions on a given item, such as the merging of two actions.

## 5. Lexicon and semantics

The system is able to build and utilize the lexicon. The lexicon builder is based on an algorithm operating on pairs of descriptions (utterances) and semantics (rules, programs, objects, actions).

## 6. Learning by Sharing and Communicating

Learning can be accomplished in a number of different ways. Solving problems, mathematical and game puzzles, searching for new theorems and mathematical proofs was the traditional focus in artificial intelligence. Unfortunately, there are very few of us that can claim to have discovered a novel theorem within our lifetime. A more realistic avenue for learning is by sharing information. We advocate learning by primarily sharing as a predecessor of the highest intellectual abilities.

## **2.3 Project Goals**

The project goals can be summarized as follows:

### 1. Information sharing

Based on the two universal principles introduced above, we want to develop a framework that allows the system to accept solicited or given information without manual reprogramming.

### 2. Semantics

Every piece of information has syntactical and semantic components that are often similarly structured. Fundamental operations over pairs of syntactical-semantic components allow for the use of language for communication between entities.

### 3. Applications

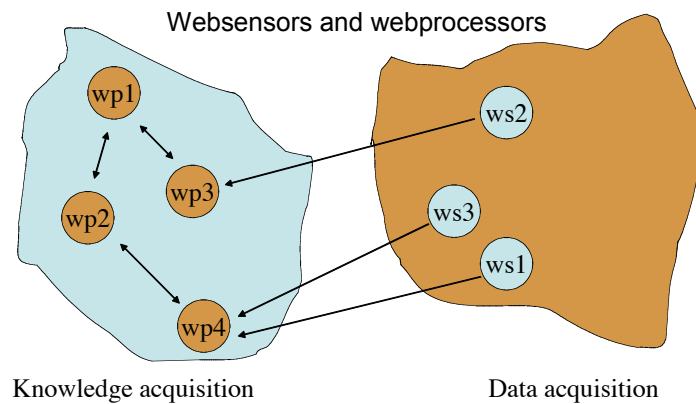
The purpose of the system is to perform useful tasks, such as providing information sharing, information merge and object monitoring without repeated re-programming.

Current applications range from a simple temperature reading to a world of agents that are able to build temperature profiles and then recognize and report abnormal profiles.

## 3 Entities Used in Sensor Networks

We distinguish between passive agents which merely report data, stand-alone agents which are able to process data on a lower layer, and language-based agents which are able to communicate about information and exchange semantics. We define

$$\begin{aligned} \text{websensors} &= \text{sensors} + (\text{micro})\text{processor} + \text{http}(\text{tcp}/\text{ip}) \text{ interface}, \\ \text{webprocessors} &= \text{http}(\text{tcp}/\text{ip})\text{interface} + \text{processor} \end{aligned}$$



**Fig. 1.** Websensors and Webprocessors

The purpose of the hardware layer is to form a network that communicates using universal protocols, such as TCP/IP and HTTP. There are several different types of processors and interfaces we have used for our experiments. One group consists of simple processors such as Basic Stamp2 and Javelin, both from Parallax [6] or the HC08 by Freescale. These processors communicate using mini-web servers such as Red-i and Siteplayer [7] that provide HTTP connections. The other group of processors consists of integrated sensors, micro-controllers and transmitters. One outstanding instance of this group is the mica mote by Intel Berkeley [5] and

distributed by Crossbow. Figure 2 models the types of processors and interface we used.

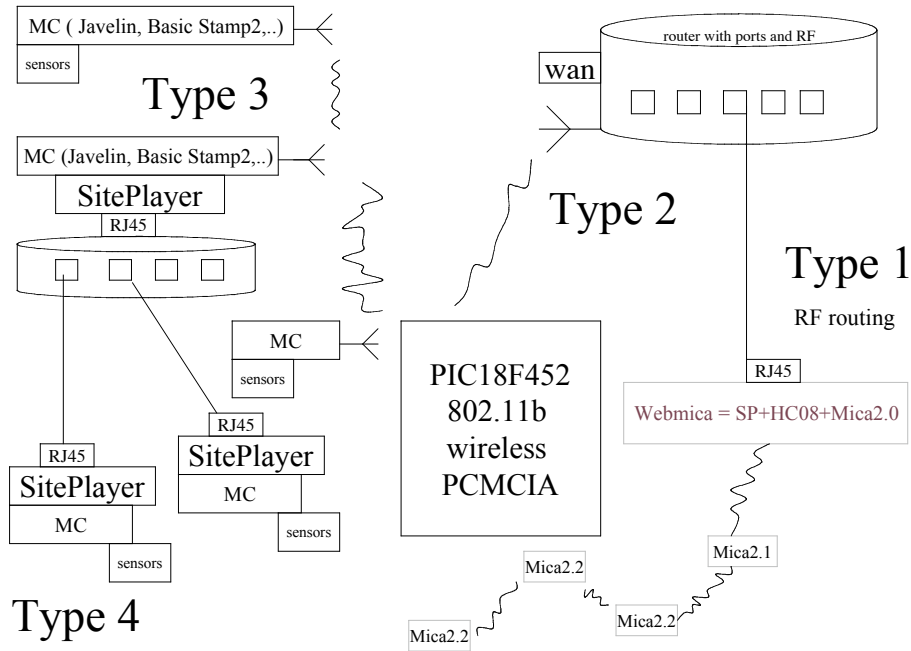


Fig. 2. Types of websensors

### 3.1 Webmica

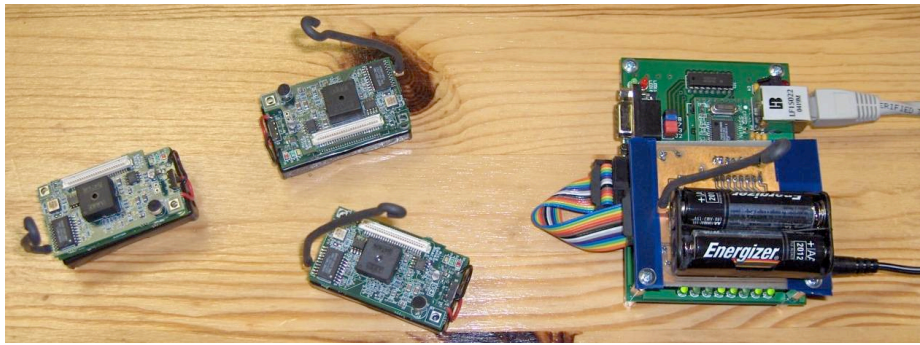
We have modified mica motes and created an example of a websensor (see figure 3). The interface for standard mica2 is provided by a custom made board consisting of generic micro-controllers. For practical reasons, a Freescale HC08 microcontroller was programmed and it is able to interface of-the-shelf webserver Site-Player distributed by NetMedia [7].

The mica wireless nodes redirect data to a designated mica2 node that serves as a base of the network. The base outputs serial data that are converted into a command for SitePlayer by additional HC08 processor. A webpage is created by a SitePlayer module. A command for writing byte 0x55 to the memory location 0x20 reads:

(0x80, 0x20, 0x50)  
(write-command, address-to-write-to, data-to-be-written).

Freescale's MC68HC908KX8 8-bit microcontroller has been chosen for our application, but any other device with similar features could be used as well. One of our reasons for this choice was the minimal number of external components necessary for running this controller. Typical quartz crystal does not have to be put on the board,

because of internal oscillator, which can be easily trimmed for exact frequency to generate baud rate for the serial communication module (UART).



**Fig. 3.** Webmica and mica2 nodes cluster

The main task of this processor is to wait for a new packet to arrive from the wireless sensor network's node 0 (MICA2) and then to delete certain bytes carrying data based on the number and order of the bytes by Node ID, Packet number. Any sensor can be connected to MICA sensor board, allowing for future expansions into object monitoring with infrared sensors, ultrasound sensors, and smoke detectors.

#### **4 The Role of Language**

Semantics has the central role in our approach to CSN. Language has several functions from the semantics point of view.

1. Language utterance can indicate actions or to declarative statements
2. Language provides convenient and critical shortcuts decreasing complexity
3. Language is context depending and hierarchical

String languages have dominated mathematics and computer science since the beginning of the 20th century. The formal definition of the Thue word game was presented by a Norwegian mathematician Alex Thue around 1914. The words in this game are equivalent if there is a finite sequence of substitutions, from a given dictionary, taking one word into other. This game was as an example of an unsolvable problem by a computer. In 1967, E. Mark Gold showed [10] that in some cases it is impossible to identify even a simple regular language. In our opinion, these examples provide fascinating results about syntax but they do not address primary issues of communication and the role of language. These types of problems and syntax analysis have prevailed in computer science because of their usefulness for computer programming languages. We agree with Winograd and Flores [11], p.124, who challenge the common understanding of how these techniques are related to the human use of language: "What is important is that people using the system recognize (as those duped by ELIZA did not) two critical things. First, they are using the

structure of their natural language to interact with a system that does not understand the language but is able to manipulate some of those structures. Second, the responses reflect a particular representation that was created by some person or group of people, and embodies a blindness of which even the builders cannot be fully aware.”

## 5 Semantics, Inference and Utterances

Jackson [14] discusses “semantic information” carried by a sentence as simply the data structure that the program creates when it processes (“understands”) the sentence. The general *problem of semantics* is to discover and define the kinds of “semantics” data structure represents.

In addition to the defining the semantic data structure, we also need to address semantic inference and semantic retrieval. In the natural world, we see many examples of learning without the use of language. Modules performing useful functions can be controlled by logic provided by some other modules. Regardless of how a response to an inquiry is obtained, it needs to be communicated using language. This leads to the problem of generating utterances from semantics.

These procedures can be best outlined using the simplest database semantics and inference. One of the simplest models which incorporates logic is Datalog (a version of Prolog for databases). We use Datalog as a paradigm for implementation and as an illustration for traditional concept of semantics.

Standard database semantics (e.g. Ullman [11]) can be represented by first-order predicate logic. There are three alternative ways to define “meaning” or semantics. Regardless of which definition is used, the inference procedure of the system leads to a generation of a new frame, object or semantics and generating a corresponding utterance. Even in the simple case of declarative semantic, there is no guarantee that a unique answer is defined, or that there is a reasonable way to turn the declarative program into a sequence of steps that will compute the answer (Ullman [11]). The alternative ways of defining semantics are:

1. Proof-theoretic interpretation of rules is the set of all the facts that can be proved in all possible ways.
2. Model-theoretic interpretation of rules is a collection of predicates assigned the truth or falsehood to every possible instance of those predicates. Usually, an interpretation is represented by its set of true instances.
3. Computational definition of meaning provides an execution algorithm to tell whether a potential fact (predicate with constants for its arguments) is true or false.

Standard semantics, used by e.g. Poole [15] for modeling robots, is a kind of database type of semantics described by Ullman.

We believe that the standard definition and declarative rules must be expanded by using procedures, programs and recursive operations, or functions that operate on themselves. We call this new process recursive action semantics. This means that an object is always bundled with other objects and functions that operate on them.

Semantics become a domain in which the system “thinks.” A special case of this process includes rules, standard logic and thinking using language or words.

Let us consider knowledge as a pair (utterance, semantics) acquired by a node. Simple semantics is particularly associated with declarative facts. One type of simplest processing is to apply standard logic operation for declarative facts. Typical logic operations that could be arguably innate or provided by the system creator are modus ponens and modus tollens. We emphasize that the system is designed to accept any pair of the type (utterance, semantics). This pair can represent a rule, a declarative fact, or some other characteristics. A special case of semantic databases are pictures that are annotated.

The retrieval step is based on both the current utterance and the semantics. The simplest task is analyzing an utterance which points to a unique semantic situation, which can be translated using a known lexicon. The next more general step calls for a semantic situation that is found based on a given utterance and a current or preferable semantics.

## 6 Algorithmic Layer-Lexicon Acquisition

Our approach emphasizes that knowledge representation and acquisition tasks should be based on *flexible* language communication. The central idea is that virtual agents communicate using evolving lexicon and language. In the traditional context-free setting, language acquisition and identification is a difficult task that cannot be successfully accomplished by using only positive examples [10]. Traditional artificial intelligence approaches difficult problems as the responsibility of an individual agent. We redefine an agent’s task as the ability to accept knowledge and skills provided by other agents and humans without strong logical reasoning capabilities. We can observe that most humans learn the majority of their capabilities from other people. These learned concepts are sometimes slightly adapted by some people and applied for ever-changing environments and conditions. Only in very exceptional cases, we are required to create a completely novel solution. Consequently, we believe we should “start small” with agents allowing them to learn from other agents and develop skills to tackle completely novel situations.

We have attempted to apply a more human-inspired method of knowledge acquisition to our network. One of the main features of our artificial system is that information and knowledge for agents is provided in the form of pairs (utterance-semantic observation). Siskind [8] proposes that an utterance can be paired to its semantics. The semantic component of the pair can be expressed as a program or a semantic expression. The mapping between utterances and semantics needs to be discovered by the agents. In doing so, agents are able to learn to understand new utterances, and moreover are able to acquire skills. These agents are then able to run newly acquired programs by themselves. This approach emphasizes distributed nature of agent systems. Our approach is consistent with speech acts and the semantic orientation in the sense that our framework supports any pair .

We describe a simple algorithm that allows us to expand an agent’s lexicon. The methods Siskind used in [8] were based on the arc-consistency algorithm [9].

Standard backtracking can work as well however, we use what we call the semantics matching algorithm. First, an agent is presented several utterances and their semantics (expressed in a symbolic or actual programming language):

Utterance	Symbolic Utterance Translation - Semantics
foo bar	LIST temperature
baz bar	LIST humidity
quux bar	LIST acceleration
baz bleen quux plugh	COPY humidity TO acceleration
baz bleen cruft foo	COPY humidity FROM temperature

The arc consistency algorithm can be visualized as a graph. The nodes show variables  $V1$  *foo*,  $V2$  *bar*, ...  $V7$  *cruft*, and their possible values (e.g. only one value *temperature* for the variable  $V1$  *foo*) that are filtered out using the domain consistency rule and the semantic matching. The links connect overlapping sets of possible values. The link numbers give the order in which they are deleted when applying the Arc Consistency [9] algorithm.

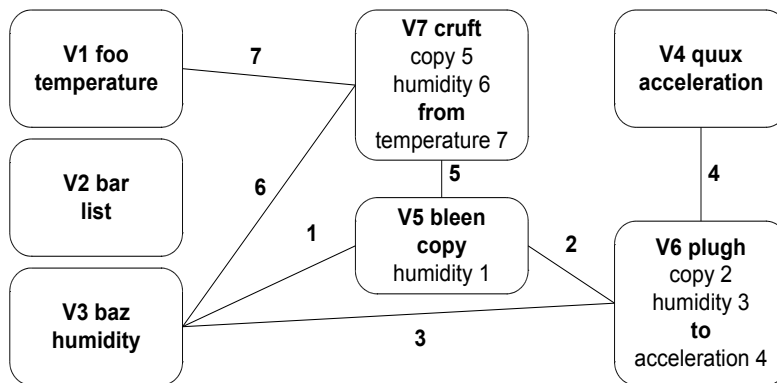


Fig. 4. Result of the Arc Consistency algorithm.

This algorithm provides the following lexicon.

Term	Translation
bar	LIST
baz	humidity
bleen	COPY
cruft	FROM

Term	Translation
foo	temperature
quux	acceleration
plugh	TO

The “translation” refers to elements in a programming language. Thus, whole utterances can then be translated into an executable program, which can be performed by agents as a response to the utterance. The algorithmic layer of this translation game involves incorporating several tasks, which correspond to additional algorithms.

These include effective information representation and search using such ideas as the small world paradigm, semantics incorporation and reasoning in the semantic layer.

Semantics matching is much simpler and faster for the semantic mapping problem. Let us illustrate the semantic matching algorithm using an example of two utterances. Two utterances and corresponding semantic expressions are

$$\begin{aligned} \text{foo bar} &= \text{list}(\text{temperature}) \\ \text{baz bar} &= \text{list}(\text{humidity}) \end{aligned}$$

The “*list*” is the only item that shows twice and therefore the “*bar*” corresponds to the “*list*”. That implies that  $\text{foo} = \text{temperature}$  and  $\text{baz} = \text{humidity}$  and this case is solved. Similarly we can proceed for pairs involving more items. The  $(m, n)$  refers now to a case where one utterance consists of  $m$  items and the other of  $n$ .

- Case (2, 2)  $(a, b) (c, b)$  implies  $b$  is known
- Case (3, 2)  $(a, b, c) (b, d)$  implies  $b$  is known
- Case (3, 3)  $(a, b, c) (b, c)$  implies  $a$  is resolved,  $b, c$  domains of values are reduced
- Case (4, 2)  $(a, b, c, d) (b, c, f, e)$  implies  $b, c, d$  are reduced to 2 semantic expression values each
- Case (4, 3)  $(a, b, c, d) (a, b, c, e)$  implies  $a, b, c$  are reduced to 3 semantic values each

By induction it can be shown that any  $(m, n)$  situation can be reduced to  $(m-1, n)$ ,  $(m, n-1)$  or better.

## 8 Implementations

We have implemented several hardware components and software components of the CSN system. The hardware components, such as webmica were already described earlier. In this section, we focus on the implementation of the software components.

The web interface accepting pairs is implemented in PHP and can be accessed via web browser by a human or by any agent via HTTP. The interface allows us to view, enter and edit semantic pairs. The semantics of these pairs can be immediately evaluated.

In our implementation, we use PHP and SQL, which can be both easily exchanged and executed by agents without any need for the compilation step or a special programming context. A script example corresponding to an utterance “*tell temperature*” is

```
<?php require_once("csn.php");
      require_once("semantics.php");
      PrintValue("tempovsd");           ?>
```

The `csn.php` provides the basic database connectivity. The `semantics.php` contains hard coded functions that express the basic semantics used in utterances. For example, the `PrintValue` function is defined (using `csn.php`) as

```
function PrintValue($what) {
```

```
print getSingleValueViaSQL(  
    "SELECT $what FROM sensors LIMIT $max,1"); }
```

which means selection and printing the last measured value from the SQL table. The function `getSingleValueViaSQL()` provides basic MySQL support function that allows to extract required information in this environment. The mapping from the utterance “*tell temperature*” to “*PrintValue('tempovsd')*” is then obvious – *tell* maps to *PrintValue*, while *temperature* maps to *'tempovsd'*. This mapping is predicated on the selection of the correct semantic frame, however, this is beyond the scope of this paper.

It is important to notice that when agents are able to exchange and execute these scripts then they are able to perform these operations themselves. Thus, after learning the lexicon and correspondence to the programs, they acquired the other agent’s skills.

The semantic mapping algorithm is implemented in Java. The component accepts pairs of utterances and semantics, and is able to find mappings between them. For each pair, both utterance and semantics is tokenized. The tokens are used in the mapping graph. The mapping graph is built incrementally as new pairs arrive. It represents the graph of possible mappings (i.e. where utterance and semantics occur in at least one pair) minus the graph of impossible mappings (i.e. where utterance and semantics do not occur in at least one pair). We do not permit uncertainties in mappings so far – the mapping can be only strict 1:1, without any noise in both utterances and semantics. Using the algorithm described above, when a mapping is exact (i.e. one to one), it is removed. This procedure is repeated for each mapping that becomes exact during the process of lexicon and semantics acquisition. In the end, an utterance can be mapped to one element of semantics (procedure/function or its parameter) or to nothing (when the utterance doesn’t have impact on semantics). This mapping is then used for translation from utterance to semantics or vice versa.

## 9 Conclusion and Future Work

We have presented both hardware and software components for the CSN project. These basic components support information and concept exchange and algorithms for lexicon acquisition. The current implementations are mainly in PHP, Java, and MySQL because of the simplicity of the implementation. We expect the communication component to be implemented using the transport layer protocol TCP/IP and HTTP to provide extensible agent communication.

With regard to immediate future work, there are many components of the project that need to be refined or implemented. The next practical application that we intend to implement is database merge and intelligent object monitoring. These applications consist of two or more webprocessors and associated websensors (figures 1 and 2). One processor archives data and knowledge from its sensors. There is a need to merge this knowledge with a similar archive of data and concepts from the other webprocessor. More generally, more webprocessors and websensors (small hardware boxes) can be distributed as independent groups. When these groups contact each

other, they will have the capability to communicate using the acquired lexicon resulting in new profiles, reasoning rules and sets of programs.

The second application deals with a task of building a model of temperature (or other quantity) in a spatial distribution. Some agents might know how to build these models and can share this skill with other agents that are needed to building the complete temperature model. The CSN project is important because it addresses the question of how to build a system that possesses the six main features including universal data communication, universal semantic communication, system tolerance, recursive action semantics, lexicon evolution, learning by sharing. Implemented CSN will be useful for home monitoring because of ease of use and for professional monitoring because of the flexibility of the system.

## Acknowledgements

J. Smid was supported in part by the CIBAC grant, Bowie State University.

A. Bencur was supported in part by the grant GACR 102/05/H525, TU Ostrava.

## References

1. PSMP (Knowledge Presentation Sharing, Mining and Protection) Workshop Series Website. <http://sks99.com/psmp.php> (2003-2005)
2. Bencur, A., Pokorny, J., Smid, J.: Communication of Autonomous Systems Based On Wireless Sensor Motes. The 7th WSEAS. Prague (2005)
3. Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A Survey on Sensor Networks. IEEE Communications Magazine (2002)
4. J. Smid, M. Obitko, V. Snasel. Semantically Based Knowledge Representation. Communications in Computing (2004)
5. Crossbow Technology Inc. Website. <http://www.xbow.com/>
6. Parallax Website. <http://www.parallax.com/>
7. Siteplayer Website. <http://siteplayer.com/>
8. Siskind, J.: Learning Word-to-meaning Mapping. In Broeder, P., Murre, J. (eds): Models of Language Acquisition. Oxford University Press (2000)
9. Mackworth, A. K.: Constraint Satisfaction. In Shapiro, S. (ed.): Encyclopedia of Artificial Intelligence. John Wiley 1987, pp. 205-211.
10. Gold, E. M.: Language Identification in the Limit. Information and Control. 10. (1967) 447-474
11. Winograd, T., Flores, F.: Understanding Computers and Cognition: A New Foundation for Design. Addison-Wesley (1988)
12. Dale, R., Moisl, H., Somers, H. (eds): Handbook of Natural Language Processing
13. Ullman, J.D.: Principles of Database and Knowledge-based Systems. Vol. I and II. Computer Science Press (1988)
14. Jackson, P. C.: Introduction to Artificial Intelligence. Second Enlarged Edition. Dover Publications (1984)
15. Poole, D., Mackworth, A. K., Goelble, R.: Computational Intelligence: A logical Approach. Oxford University Press (1998)
16. Baum, E. B.: What Is Thought? The MIT Press (2004)