Towards a uniform ontology-driven approach for modeling, checking and executing WSANs

Claudia Vannucchi, Diletta Romana Cacciagrano, Rosario Culmone, Leonardo Mostarda
Computer Science Division
School of Science and Technology, University of Camerino
Camerino, Italy
Email: *Claudia.Vannucchi,Diletta.Cacciagrano, Rosario.Culmone, Leonardo.Mostarda* @unicam.it

***Abstract*—Wireless sensor and actuator networks (WSANs) refer to a group of sensors and actuators linked by wireless medium to perform distributed sensing and acting tasks. Being reactive systems, quite often WSANs are programmed by Event-Condition-Action (ECA) rule-based languages. Because of potential interactions among the rules themselves and their surprising effects on the system behaviour, it is quite difficult to ensure a safe behaviour of a WSAN at design time. This paper proposes a semantic-driven approach for *modeling*, *checking* and *executing* ECA rule-based WSANs. The approach can be considered uniform since any of the previous phases works on a common ontology-based WSAN model integrating and linking three different system views: the static one (i.e., sensor and actuator types in the WSAN, description of its environment, deployment information etc.), the dynamic one (i.e., a set of ECA rules programming the given WSAN) and the behavioural one (i.e., a finite state machine (FSM)-based representation of the WSAN behaviour w.r.t. the given dynamics). We show how the proposed approach provides an agile verification of system properties involving (and also mixing) static, dynamic and behavioural concepts. In particular, with regard to the behavioural view, we will focus on properties like consistency, correctness and termination. In order to better explain our approach, we present a home automation case study.**

I. Introduction

With the rapid advancement in wireless communications technology and micro-electromechanical systems technology, the wide deployment of large-scale wireless sensor networks (WSNs) has been made possible [1]. Due to their features of reliability, accuracy, flexibility, cost-effectiveness and ease of deployment, WSNs are used in a wide range of applications (e.g., environment monitoring, health care, etc.). At the same time, there are an increasing number of applications that require the use of actuators along with sensors [2]. This occurs when the network system needs to interact with the physical system or the environment via actuators. Actuators transform an input signal into an action upon the environment. Typical examples of actuators are robots, electrical motors, and humans. For these reasons, wireless sensor actuator networks (WSANs) are emerging as the next generation of WSNs [2]. WSANs refer to a group of sensors and actuators linked by wireless medium to perform distributed sensing and acting tasks [3]. Differently from WSNs, WSANs are capable of changing the environment and physical world. More precisely, sensor nodes are used to gather information from the environment, and actuator nodes are used to change the behavior of the environment [2]. WSANs behave like *reactive systems* and, for this reason, Event-Condition-Action (ECA) rules-based languages are often used in order to program them. On the one hand, ECA rules provide for high-level and flexible description for WSANs; on the other hand, it is well known that ECA rule-based programming is an error-prone process [4], due to the possibility of complex interactions among rules. Very often such interactions introduce, in the system dynamics, behavioural errors which are often detected only at run time (i.e., when a WSAN has been already programmed and deployed). For this reason various formal methods have been suggested (e.g., [4] approaches based on Petri Net (PN) [5], SMV [6] and SPIN [7]). However, they have not been tailored to the context of WSANs.

Contribution of the paper. This paper aims to contribute in this direction, proposing a semantic-driven approach for *modeling*, *checking* and *executing* ECA rule-based WSANs. The approach can be considered uniform since any of the previous phases works on a common ontology-based WSAN model integrating and linking three different system views: the static one (i.e., sensor and actuator types in the WSAN, description of its environment, deployment information etc.), the dynamic one (i.e., a set of ECA rules programming the given WSAN) and the behavioural one (i.e., a finite state machine (FSM)-based representation of the WSAN behaviour w.r.t. the given dynamics). We show how the proposed approach allows us an agile verification of system properties involving (and mixing) static, dynamic and behavioural concepts. On the one hand, the type of proposed approach presents numerous benefits arising from the ontology features (e.g., interoperability, knowledge reuse and information integration with automatic validation). On the other hand, the integration of static, dynamic and behavioural aspects implements a monolithic model that reduces inconsistency problems. Furthermore, relating ECA rule semantics to that one of FSM-based models, widely used for verification and execution, we obtain a framework supporting both compilation-time analysis and run-time enforcement of coordination of rules.

A uniform ontology-based model integrating dynamic and behavioural aspects of a WSAN also allows us to analyze the system outcome after its execution. For instance, it would be possible to run queries for extracting information about “which rules have been fired in which states", "what the emergent behaviours of the whole system have been ' ', as well as statistics concerning system rules and states. Ontology features make such functionalities dynamic and modular. Furthermore, the behavioural component in the semantic model can drive both the execution/simulation phase and also the debugging one of the modeled system.

Plan of the paper. The remainder of the paper is structured as follows: Section II presents a review of the literature concerning the topics addressed in this work. Section III provides a description of the proposed approach. Section IV illustrates how the proposed approach can be useful for designing and programming a home automation WSAN, ensuring its behavioural "correctness" at design time. Finally Section V provides a conclusion and outlines future work.

II. Semantic approaches for WSANs and reactive rules in literature

In general, expressiveness requirements in reactive system representation include the ability to represent hierarchical structures, complex relationships among context instances and complex definitions based on simpler ones, usually using restrictions that may be spatial or temporal [8]. Ontologies have shown to be one of the most promising tools to achieve these goals, since they provide a way to represent and share knowledge by using common vocabulary, to separate declarative and procedural knowledge, to make information machine readable and processable, as well as to derive implicit information from explicit context data. In a scenario characterized by heterogeneous devices connected together into a network, ontologies are becoming widely used to describe the domain and achieve efficient interoperability of information system [9], [10].

There is a broad variety of ontologies and vocabularies to model WSANs and, more in general, domotic/ambient intelligence systems.

A first example is found in [11], where an ontology is implemented for both formally expressing the domotic environment (e.g., sensors, gateways and network) and providing reasoning mechanisms. This reasoning allows to supports automatic recognition of device instances and to verify the formal correctness of the model. The DogOnt ontology supports device/network independent description of houses, including both controllable and architectural elements. It is currently adopted to provide house modeling and reasoning capabilities to a domotic gateway called DOG (Domotic OSGi Gateway), a coordination framework for supporting dynamic module activation, hot-plugging of new components and reaction to module failures [11]. The combination of DOG and DogOnt supports the evolution of domotic systems into IDEs by providing means to integrate different domotic systems, to implement inter-network automation scenarios, to support logic-based intelligence and to access domotic systems through a neutral interface. In this context, a third component of DogOnt, namely DogOnt queries supports runtime control of the IDE.

Within DomoML, DomoML-env is a vocabulary through which is possible to define physical resources of a domestic environment [12]. Resources are described through a standardized mark-up language based on RDF/XML, which any appliance constructor (or integrator) can use to describe and represent its own products. The adhesion to the DomoML-env can guarantee to different constructors that their appliances will be able not only to communicate with other DomoML-env compliant devices, but also to share semantics mainly about their functionalities, and interoperate on more complex integrated operations.

A previous work with relevant objectives about pervasive computing is the SOUPA Project [12], where a Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) is defined and expressed using the Web Ontology Language (OWL). It includes modular component vocabularies to represent intelligent agents with associated beliefs, desires, and intentions, time, space, events, user profiles, actions, and policies for security and privacy [13] .

In [14] an ontology for enabling Ambient Intelligence in a Smart Building, named BOnSAI (Smart Building Ontology for Ambient Intelligence), is proposed. The ontology extends and benefits from existing ontologies in the field, but also adds classes needed to sufficiently model every aspect of a service-oriented smart building system. There exist already domain-independent upper ontologies (not officially proclaimed standards yet) that enable the vision of Semantic Web services. The ontology is domain-dependent and specializes such ontologies in order to model the domain-specific concepts of the Ambient Intelligence application.

Most of the above ontologies are used only for integration over various domains. Furthermore, any of them provides a semantic model of the system focusing only on its static features (e.g., system and context components, deployment information, etc.). They ignore dynamics and behaviour, two aspects of WSANs - and in general of reactive systems - that should be taken into account already during the modeling phase. In fact, being reactive systems, WSANs can employ the ECA rule-based programming mechanism [15], increasing the difficulty to predict and analyse the WSAN behaviour because of the ability of rules to interact with each other. Particularly in IoT area, the rules that govern the relations between sensors and actuators can lead to highly distributed collaborative applications. It follows that run time coordination and formal analysis for WSANs becomes a necessity to avoid side effects mainly when applications are critical.

Reaction rules constitute a promising approach to specify and program reactive systems in a declarative manner [16]. In particular, they provide the ability to reason over events, actions and their effects, and allow detecting events and responding to them automatically. A great variety of approaches have been developed for reaction rules, which have for the most part evolved separately and have defined their own domain and platform specific languages [17], [18], [19]. Novel semantics are being devised, including for the Logic-based agent and Production System language (LPS) and KELPS [20].

In [16] the authors address the Reaction RuleML subfamily of RuleML and survey related work. Reaction RuleML is a standardized rule markup/serialization language and semantic interchange format for reaction rules and rule-based event processing. Reaction rules include distributed Complex Event Processing (CEP), Knowledge Representation (KR) calculi, as well as ECA rules, Production (CA) rules, and Trigger (EA) rules.

Reaction RuleML provides several layers of expressiveness for adequately representing reactive logic and for interchanging events (queries, actions, event data) and rules. The first level defines general concepts (e.g., space, time, event, action situation, process, agent, etc.) in a modularized ontological top-level structure, with a left to right vertical order in the top-level ontologies. For instance, the concepts and relations for time and space are used in the event and action ontology, which is employed in the situation ontology etc. These general concepts defined in the top-level ontologies can be further specialized with existing domain ontologies and ontologies for generic tasks and activities (e.g., situation processing, processes/workflows, agents including their pragmatic protocols, etc.). The application ontologies specialize these domain and task concepts w.r.t. a specific application, often on a more technical platform specific level. The second level defines the event, action, and interval algebra operators for complex events, actions, and intervals as well as other semantic concepts in the respective ontologies.

In [21] the authors propose an ontology-based approach for describing (reactive) behavior on the Web and evolution of the Web that follows the ECA paradigm: MARS is a modular framework for composing languages for events, conditions, and actions by separating the ECA semantics from the underlying semantics of events, conditions and actions. This modularity allows for high flexibility w.r.t. the heterogeneity of the potential sublanguages, while exploiting and supporting their meta-level homogeneity on the way to the Semantic Web.

A markup proposal for active rules can be found already in RuleML [22], but it does not tackle the complexity and language heterogeneity of events, actions, and the generality of rules, as described here. The RDF level of rules provides the base for reasoning about rules. Using the above ontologies, every rule is interpreted as a network of RDF resources of the contributing ontologies (ECA, event algebras, OWLQ specifications, application domains etc.). RDF/OWL reasoning can be applied for analysis of structural correctness, safety of variables and evaluation order.

III. Ontological Approach

In this section the ontological approach to build a uniform three-view model of a WSAN is presented. Such an approach aims at providing a combined semantic description of static, dynamic and behavioural aspects of the system, so that supporting querying and reasoning mechanisms for the formal verification of static and behavioural correctness of the system. In order to build such an ontological framework, the description of static, dynamic and behavioural aspects is required.

A. Ontology architecture

In our approach, the ontology includes:

- a formal description O_1 of static aspects of the system;
- a formal description O_2 of the system dynamics as a set of ECA rules;
- a formal representation O_3 of the system behaviour as a finite state machine FSM (according to the static aspects of the system in O_1 and the set of ECA rules in O_2).

The details are given below:

- O_1 describes the declarative part of the system, i.e. concepts like sensors and actuators and their mutual properties and relationships, the deployment context of the devices, etc. For this purpose we can exploit and combine suitable ontologies like SSNO [23] and DogOnt. The values that sensors and actuators can assume, belong to finite sets. For instance the boolean set or a finite integer set. Real numbers are also represented by a finite set of values.
- O_2 conceptualizes the system dynamics expressed by an ECA rule set. It formalizes concepts like event, condition and action. A specific ECA rules-based program is a set of O_2 instances. System invariances are expressed as ECA rules with no actions. The MARS ontology can be used for this purpose [24]. In particular, conditions are conceptualized as predicates including functions operating on finite domains, comparison and boolean operators, while actions are modeled as sets of assignments to labels representing actuators. Assignments are executed in an atomic way. The assigned values are expressions evaluated using the labels of sensors and actuators.
- O_3 conceptualizes the system behaviour (w.r.t. the dynamics described in O_2) expressed as a finite state machine (FSM). The possibility to exploit the FSM formalism follows by the assumption that all domains are finite. In detail, we take into account the State Chart extensible Markup Language (SCXML) [25]. Hence, O_3 provides a suitable conceptualization of SCXML and the system behaviour is represented by all O_3 instances. Such instances are calculated generating all admissible system states (i.e., tuples that verify all system invariances) and all possible state transitions fired by any ECA rule in O_2. SCXML is an XML-based markup language that provides a generic state-machine based execution environment based on Harel statecharts [26]. On the one hand, the conceptualization of SCXML in O_3 gives us directly the conceptualization of a system behaviour as a FSM. On the other hand, the executable nature of SCXML give us a mechanism to simulate and execute the system described in O_1 and programmed in O_2 by executing the corresponding O_3 instances through a SCXML interpreter.

B. Ontology instantiation

For what concern O_1 and O_2 the procedure is simple. The instances of O_1 are exactly the specific devices (sensors and actuators) and context elements of the specific WSAN, while the instances of O_2 are the ECA rule set representing specific functionalities of the given system.

For what concern O_3 we need to generate all possible states and transitions describing the behaviour of the specific set of ECA rules in O_2 related to a specific functionality of the system in a corresponding FSM.

For this purpose, an incremental approach is followed. First of all, we generate the set of all possible *admissible states* that satisfy all *invariances* of the system. Then, the set of all *admissible transitions* is created, i.e., all state transitions obtained applying ECA rules, guaranteeing specific properties (e.g., determinism, consistency, termination).

Details are described below.

1) State generation: The set of all possible admissible states is generated, first calculating the cartesian product of all finite domains of sensors and actuators conceptualized in O_1, then selecting only those states verifying the system invariances in O_2. An admissible state is simply a tuple of values of the form $< I, O >$, where I is a vector of values of all sensors (input components) and O is a vector of values of all actuators (output components). If the vector I has m components, i.e. $I =< i_1, ..., i_m >$ and the vector O has n components, i.e. $O =< o_1, ..., o_n >$, then the number of all possible states is 2^{m+n} . Applying the invariances we obtain the set of admissible states, let say of cardinality z . We denote the set of all admissible states $s_1, ..., s_z$ in S.

2) Transition generation and property verification: The following algorithm is used to generate all the admissible transitions on the set of the admissible states w.r.t. the set of ECA rules in O_2. The algorithm can be described using a matrix $z \times 1$, where z is the cardinality of the set of the all admissible states and where each row corresponds to a specific admissible state.

The generation procedure includes the verification of the following properties of the system: consistency, determinism and termination. The definitions of these properties have been adapted from [4].

A system satisfies the *consistency* propriety if its rules are not *unused*, *incorrect*, *redundant* and *contradicting*.

A rule is called *unused* if it can never be applied. A rule is defined *incorrect* if can lead to a state that is outside the domain (i.e., a state that is not admissible for the system). *Redundancy* is the case where there are rules or chain of rules that are identical. The condition of these rules is always true for the same states and when applied lead always to the same state. A *contradiction* consists of a logical incompatibility between two or more rules.

Given an ECA rule r in O_2 (the following procedure must be repeated for all ECA rules in O_2), we check the condition c_r of r for all states belonging to the matrix.

If the condition c_r can never be applied (for example, if t indicates a sensor temperature with range $(0, 40)$ the condition $t > 50$ is not applicable), then the rule r is labelled as *unused* and is eliminated. Otherwise, i.e. if the set S^r of all states s_i^r satisfying the condition c_r has cardinality $k \neq 0$, then the action of r is applied to s_i^r for $i = 1, ..., k$, and for each s_i^r the target state s_i' is generated. For all s_i' with $i = 1, ..., k$, if exist some $s_i' \notin S$, then the rule r is labelled as *incorrect*. Otherwise, the state s_i' is identified as the target state of the each s_i^r. For what concerns the *redundancy* property, it suffices to check whether for any rule r, w, with $r \neq w$, and for any $s_i = s_i^r = s_i^w$ satisfying condition $c_r \wedge c_w$, the target state of s_i^r coincides with the one of s_i^w. In this case w (or r) is considered redundant. Similarly, a *contradiction* can be put in evidence checking for any rules r, w, with $r \neq w$, whether $c_r = \neg c_w$ holds. Once cut off unused, incorrect, redundant and contradicting rules, an association between the ontological representation of any admissible ECA rule r and the source state s_i^r is created.

Determinism. During the generation phase of the automaton, it could happen that more than one admissible target state is generated for a certain source state s_i. It happens whenever there is at least two rule r, w, with $r \neq w$, such that $s_i = s_i^r = s_i^w$ satisfies condition $c_r \wedge c_w$ and the target state of s_i^r is different from the one of s_i^w. This means that the determinism property of the system is denied.

In order to guarantee the determinism of the finite state machine, this kind of situation must be excluded. To solve this problem, at the end of the automaton generation procedure, all rows having two or more admissible target states are selected through the association with ECA rules in O_2. The selected set is made up of all those ECA rules denying determinism. If the user wants to eliminate an ECA rule in this set, the target states of those source states to which the ECA rule is applicable are eliminated.

Termination. The termination is defined as the property assuring that for all source states, the target state of the automaton becomes stable in a finite time. A stable state is a state whose target state is empty, that is to say the target state is not reached by applying an ECA rule to the source state. It is possible to move from this state only in consequence of an environment change (variation of sensor values).

For each state s_i having target state not empty, the not reflexive transitive closure $cl(s_i)$ on the target state is calculated. For all s_i it must be $s_i \notin cl(s_i)$.

If at least one s_i not satisfying this property exists, then the automaton has a cycle. The set of rules associated to the states involved in the cycle is selected for the user.

IV. Home Automation Case study

Monitoring and automatic control of building environment is a case study considered quite often in the literature [27], [28]. Home automation can include the following functionalities: (i) heating, ventilation, and air conditioning (HVAC) systems; (ii) emergency control systems (home burglar security alarm, fire alarm); (iii) centralised light control; and (iv) other systems to provide comfort, energy efficiency and security. In order to validate our approach we consider the home burglar security alarm system.

A security alarm system is designed to detect intrusion into a building or area. Security alarms are used in residential, commercial, industrial, and military properties for protection against burglary as well as personal protection against intruders. Home security systems work on the simple concept of securing entry points into a home with sensors that communicate with a control panel or command center installed in a convenient location somewhere in the home.

For our case study, we consider a security alarm composed by passive infrared detectors (PIRs). A PIR motion detector is one of the most common sensors found in household and small business environments. PIR sensors do not detect motion; rather, they detect abrupt changes in temperature at a given point. As an intruder walks in front of the sensor, the temperature at that point will rise from room temperature to body temperature, and then back again. This quick change triggers the detection.

We assume that a 360 Degree Passive Infrared Motion Sensor is placed on the ceiling of every room, and each room is completely covered by its sensor. In our example, the house has two rooms, a bathroom and a living room. In addition we assume that only one person lives in the house. The occupancy sensor placed in every room can be used for the intrusion detection but also for the light switch; we also hypothesize that in every room there is a lighting button to turn on and off the light manually (for example, when someone go to sleep).

The set of devices placed in the house is given by $\mathcal{D} = \{L_m, L_l, L_s, L_a, B_m, B_l, B_s, B_a, L_w\}$, where L defines the living room, B defines the bathroom, the letter m defines the PIR detector, a light sensor is denoted by l, a light switch is represented by s, a lamp actuator by a. For example the light sensor in the bathroom is labelled with B_l. The entrance is in the living room and from living room you can enter the bathroom through the door. In addition one warning alarm actuator w is associated to the living room (L_w).

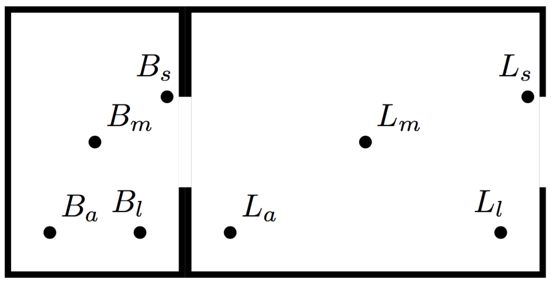

Fig. 1. House plan

The formalism used for ECA rules is the following: $E[C]/A$, where E is a set of labels in $\mathcal{D}$, C is a predicate with inputs in $\mathcal{D}$ and A is a set of actions on actuators in $\mathcal{D}$. This formalism can also be used to represent rules without an action (or a condition, or an event).

Firstly, the invariances of the system, i.e. all those constraints that are independent from the behaviour of the system, are defined. For this example, the following invariances are defined

i1. $L_m, B_m[L_m = false \wedge B_m = true]$
i2. $B_m, L_m[B_m = false \wedge L_m = true]$
i3. $L_m[L_m = false \wedge B_m = false]$
i4. $L_m[L_m = true \wedge B_m = false]$
i5. $L_a, B_a[(L_a = true \oplus B_a = true) \vee (L_a = false \wedge B_a = false)]$.

The first constraint, for instance, states that if a person is inside the living room, and both occupancy sensors B_m, L_m change value, then the person must have moved from the living room to the bathroom. The last axiom excludes the situation in which lights are on in both rooms. Then, the dynamic of the system is defined using the following ECA rules:

d1. $B_m[L_m = false \wedge B_m = true]/L_w \leftarrow true$
d2. $B_m, L_m[B_m = false \wedge L_m = true]/ B_a \leftarrow false, L_a \leftarrow true$
d3. $B_m, L_m[B_m = true \wedge L_m = false]/ B_a \leftarrow true, L_a \leftarrow false$

The following ECA rules are examples of rules that make the system properties not satisfied.

Incorrect rule:

d4. $L_m[B_a = true \wedge L_m = true]/L_a = true$

Unused rule:

d5. $B_m[L_a = true \wedge B_a = true]/L_a \leftarrow false$

Non termination (cycle):

d6. $L_m[L_m = true]/L_a = true$
d7. $L_l[L_l = true]/L_a = false$

The following rules can be introduced to avoid the previous situation:

d8. $L_m[L_l = false \wedge L_a = false \wedge L_m = true]/L_a \leftarrow true$
d9. $L_m[L_l = true \wedge L_a = true \wedge L_m = false]/L_a \leftarrow false$

Redundancy rules:

d10. $B_m[B_a = false]/B_a \leftarrow true, L_a \leftarrow false$
d11. $B_m[B_a = false \wedge L_a = true]/B_a \leftarrow true, L_a \leftarrow false$

V. Conclusion

We proposed a semantic-driven approach for *modeling*, *checking* and *executing* ECA rule-based WSANs. The approach can be considered uniform since any of the previous phases works on a common ontology-based WSAN model integrating and linking three different system views: the static one (i.e., sensor and actuator types in the WSAN, description of its environment, deployment information etc.), the dynamic one (i.e., a set of ECA rules programming the given WSAN) and the behavioural one (i.e., a finite state machine (FSM)-based representation of the WSAN behaviour w.r.t. the given dynamics).

The proposed approach provides an agile verification of properties like consistency, correctness and termination, and these properties can be defined in general for all systems. The SCXML is used during verification and execution phases.

In the future, we would like to allow the definition of new properties in a dynamic way, exploiting only query and reasoner-based methods (which are native in ontologies). For this reason, we are thinking about translating the verification procedure only in terms of SPARQL queries.

Moreover, if the approach is managed only with ontologies, it is possible to define and verify not only the generic properties considered in this paper, but also system-dependent properties (for instance, fault tolerance).

REFERENCES

[1] C. Zhang, M. Li, and Q. Pan, "An eca rules based middleware architecture for wireless sensor networks," in *PDCAT*. IEEE Computer Society, 2005, pp. 586–588. [Online]. Available: http://dblp.uni-trier.de/db/conf/pdcat/pdcat2005.html

[2] A. Casteigts, A. Nayak, and I. Stojmenovic, *Applications, Models, Problems and Solution Strategies*. Wiley, Jan 2010, ch. 1 of Wireless Sensor and Actuator Networks - Algorithms and Protocols for Scalable Coordination and Data Communication, Nayak, A. and Stojmenovic, I. (Eds.).

[3] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges." *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/adhoc/adhoc2.html

[4] F. Corradini, R. Culmone, L. Mostarda, L. Tesei, and F. Raimondi, "A constrained ECA language supporting formal verification of wsns," in *29th IEEE International Conference on Advanced Information Networking and Applications Workshops, AINA 2015 Workshops, Gwangju, South Korea, March 24-27, 2015*, 2015, pp. 187–192.

[5] X. Jin, Y. Lembachar, and G. Ciardo, "Symbolic verification of eca rules." in *PNSE+ModPE*, ser. CEUR Workshop Proceedings, D. Moldt, Ed., vol. 989. CEUR-WS.org, 2013, pp. 41–59. [Online]. Available: http://dblp.uni-trier.de/db/conf/apn/pnse2013.html

[6] I. Ray and I. Ray, "Detecting termination of active database rules using symbolic model checking." in *ADBIS*, ser. Lecture Notes in Computer Science, A. Caplinskas and J. Eder, Eds., vol. 2151. Springer, 2001, pp. 266–279. [Online]. Available: http://dblp.uni-trier.de/db/conf/adbis/adbis2001.html

[7] E.-H. Choi, T. Tsuchiya, and T. Kikuno, "Model checking active database rules under various rule processing strategies," *IPSJ Digital Courier*, vol. 2, pp. 826–839, 2006.

[8] N. D. Rodríguez, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores, "A survey on ontologies for human behavior recognition," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–33, 2014. [Online]. Available: http://doi.acm.org/10.1145/2523819

[9] R. Culmone, M. Falcioni, and M. Quadrini, "An ontology-based framework for semantic data preprocessing aimed at human activity recognition," in *SEMAPRO 2014 : The Eighth International Conference on Advances in Semantic Processing*. Alexey Cheptsov, High Performance Computing Center Stuttgart (HLRS), 2014.

[10] R. Culmone, P. Giuliodori, and M. Quadrini, "Human activity recognition using a semantic ontology-based framework," *International Journal On Advances in Intelligent Systems*, vol. 8, no. 1,2, pp. 159–168, 2015.

[11] D. Bonino, E. Castellina, and F. Corno, "The dog gateway: enabling ontology-based intelligent domotic environments," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1656–1664, November 2008.

[12] L. Sommaruga, A. Perri, and F. Furfari, "Domoml-env: an ontology for human home interaction." in *SWAP*, ser. CEUR Workshop Proceedings, P. Bouquet and G. Tummarello, Eds., vol. 166. CEUR-WS.org, 2005. [Online]. Available: http://dblp.uni-trier.de/db/conf/swap/swap2005.html

[13] D. Bonino and F. Corno, "Dogont - ontology modeling for intelligent domotic environments," in *Proceedings of the 7th International Conference on The Semantic Web*, ser. ISWC '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 790–803.

[14] T. G. Stavropoulos, D. Vrakas, D. Vlachava, and N. Bassiliades, "Bonsai: A smart building ontology for ambient intelligence," in *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics*, ser. WIMS '12. New York, NY, USA: ACM, 2012, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2254129.2254166

[15] J. Cano, E. Rutten, G. Delaval, Y. Benazzouz, and L. Gurgen, "Eca rules for iot environment: A case study in safe design." in *SASO Workshops*. IEEE Computer Society, 2014, pp. 116–121. [Online]. Available: http://dblp.uni-trier.de/db/conf/saso/saso2014w.html

[16] A. Paschke, H. Boley, Z. Zhao, K. Teymourian, and T. Athan, "Reaction ruleml 1.0: Standardized semantic reaction rules." in *RuleML*, ser. Lecture Notes in Computer Science, A. Bikakis and A. Giurca, Eds., vol. 7438. Springer, 2012, pp. 100–119.

[17] A. Paschke and A. Kozlenkov, "Rule-based event processing and reaction rules," in *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, ser. RuleML '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 53–66.

[18] A. Paschke and H. Boley, "Rules capturing events and reactivity," *Emerging Rule-Based Languages and Technologies*, 2009.

[19] A. Paschke, P. Vincent, and F. Springer, "Standards for complex event processing and reaction rules," in *Rule - Based Modeling and Computing on the Semantic Web*, ser. Lecture Notes in Computer Science, F. Olken, M. Palmirani, and D. Sottara, Eds. Springer Berlin Heidelberg, 2011, vol. 7018, pp. 128–139. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24908-2_17

[20] R. A. Kowalski and F. Sadri, "A logic-based framework for reactive systems." in *RuleML*, ser. Lecture Notes in Computer Science, A. Bikakis and A. Giurca, Eds., vol. 7438. Springer, 2012, pp. 1–15. [Online]. Available: http://dblp.uni-trier.de/db/conf/ruleml/ruleml2012.html

[21] T. Eiter, G. Ianni, T. Krennwallner, and A. Polleres, "Reasoning web," in *Rules and Ontologies for the Semantic Web*, C. Baroglio, P. A. Bonatti, J. Maluszynski, M. Marchiori, A. Polleres, and S. Schaffert, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–53.

[22] (2015) Ruleml. [Online]. Available: http://www.ruleml.org/

[23] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W. D. Kelsey, D. L. Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, and K. Taylor, "The {SSN} ontology of the {W3C} semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 17, pp. 25 – 32, 2012.

[24] C. Schlenoff and M. Gruninger, "Towards a formal representation of driving behaviors." in *FAABS*, ser. Lecture Notes in Computer Science, M. G. Hinchey, J. L. Rash, W. Truszkowski, C. Rouff, and D. F. Gordon-Spears, Eds., vol. 2699. Springer, 2002, pp. 290–291. [Online]. Available: http://dblp.uni-trier.de/db/conf/faabs/faabs2002.html

[25] W3C. (2015, 9) State chart xml (scxml): State machine notation for control abstraction. [Online]. Available: http://www.w3.org/TR/scxml/

[26] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, no. 3, pp. 231–274, 1987.

[27] D.-M. Han and J.-H. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1403–1410, Aug 2010.

[28] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 2, pp. 422–430, May 2009.