Department of Electrical and Computer Engineering

SEMIoTICS: Semantically-enhanced IoT-enabled Intelligent Control Systems

George M. Milis

A dissertation submitted to the University of Cyprus in partial fulfillment of the requirements for the degree of Doctor of Philosophy

June, 2018

VALIDATION PAGE

George M. Milis

SEMIoTICS: Semantically-enhanced IoT-enabled Intelligent Control Systems

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy at the Department of Electrical and Computer Engineering, and was approved on June *18,* 2018 *by the members of the Examination Committee.*

Committee Chair _____
Dr. Maria Michael

Research Supervisor _____
Dr. Marios Polycarpou

Research Supervisor _____
Dr. Christos Panayiotou

Committee Member _____
Dr. George Pallis

Committee Member _____
Dr. Riccardo Ferrari

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

_____

_____

Περίληψη

Αυτήν την εποχή, η ανθρωπότητα διανύει την επανάσταση του «Διαδικτύου των Πραγμάτων» (Internet of Things - IoT), . Εκτιμάται ότι μέχρι το 2020 ο αριθμός των συνδεδεμένων στο διαδίκτυο συσκευών με δυνατότητα ανταλλαγής δεδομένων πιθανόν να υπερβεί τα 50 δισεκατομμύρια. Αυτές οι συσκευές, μέσω της συνδεσιμότητας τους, επαυξάνουν την τεχνητή νοημοσύνη των συστημάτων σε διάφορους τομείς εφαρμογών. Η αλληλεπίδραση των συστημάτων με το φυσικό και τεχνητό τους περιβάλλον βελτιώνεται σημαντικά μέσω προηγμένων δυνατοτήτων ανίχνευσης συμβάντων, ανάλυσης και ανάληψης κατάλληλων δράσεων. Βασική πρόκληση από τη σκοπιά του αυτόματου ευφυούς ελέγχου μέσω ανατροφοδότησης, είναι η δυνατότητα εύκολης ενσωμάτωσης νέων στοιχείων IoT ή τροποποίησης/διαμόρφωσης υφιστάμενων συστημάτων ελέγχου χωρίς να χρειάζεται να σταματήσει η λειτουργία του συστήματος και να επανασχεδιαστεί.

Η ερευνητική κοινότητα αναγνώρισε πρόσφατα την πρόκληση του σχεδιασμού ευέλικτων και προσαρμόσιμων συστημάτων ελέγχου που θα επωφελούνται από την σε πραγματικό χρόνο αλλαγή του αριθμού και του τύπου των διαθέσιμων συσκευών σε ένα σύστημα μεγάλης κλίμακας. Στα πλαίσια αυτής της ερευνητικής εργασίας, εκμεταλλευόμαστε τεχνολογίες από την ερευνητική περιοχή του «σημασιολογικού διαδικτύου», προχωρώντας στο σχεδιασμό μιας καινοτόμας αρχιτεκτονικής ευφυών συστημάτων ελέγχου, βασισμένης στις τεχνολογίες σημασιολογικής μοντελοποίησης και το Διαδίκτυο των Πραγμάτων (Semantically-Enhanced, IoT-enabled Intelligent Control Systems (SEMIoTICS)). Η υπό αναφορά αρχιτεκτονική και σύστημα, SEMIoTICS, ενσωματώνει ένα υπό-σύστημα «Επόπτη» που με τη σειρά του βασίζεται σε ένα «Γράφο Αποθήκευσης Γνώσης» για τη διευκόλυνση της σημασιολογικής περιγραφής των συνιστωσών στοιχείων και την υλοποίηση των μηχανισμών εξαγωγής λογικών συμπερασμάτων. Στόχος είναι η επακόλουθη συνεχής ενημέρωση της διαμόρφωσης του συστήματος ελέγχου. Το σύστημα ελέγχου που προκύπτει, επιτρέπει την σε πραγματικό χρόνο προσαρμογή στις τρέχουσες δυνατότητες που διαθέτουν τα επί μέρους διαθέσιμα στοιχεία.

Δείχνουμε τη δυνατότητα εφαρμογής του SEMIoTICS μέσω επεξηγηματικών σεναρίων στην περιοχή των Ευφυών Κτιρίων (Smart Buildings). Συγκεκριμένα, παρουσιάζουμε τον τρόπο με τον οποίο το υπό-σύστημα «Επόπτης» είναι σε θέση να υποστηρίξει τη διαδικτυακή (επανα-)διαμόρφωση του συστήματος ελέγχου, ώστε να επιλύει το πρόβλημα ρύθμισης της θερμοκρασίας χώρου σε κτίρια πολλαπλών ζωνών. Μετά από οποιαδήποτε αλλαγή στις διαθέσιμες δυνατότητες ανίχνευσης, ανάλυσης ή/και ανάληψης δράσης ή ακόμη και στις παραμέτρους του ίδιου του κτιρίου, το υπό-σύστημα «Επόπτης» ξεκινά μια διαδικασία κατά την οποία ανακτά αποθηκευμένες γνώσεις σχετικά με το κτίριο και τα διαθέσιμα στοιχεία του συστήματος ελέγχου και τις χρησιμοποιεί για να διαμορφώσει σε πραγματικό χρόνο ένα σχέδιο κατανομής των στοιχείων για την υλοποίηση του απαιτούμενου ελέγχου, καθώς και ικανές λειτουργίες για τη βέλτιστη διανομή του σήματος ελέγχου σε (εν δυνάμει υπαρκτούς) πολλαπλούς ενεργοποιητές σε κάθε ζώνη του κτιρίου.

Abstract

Humanity is currently experiencing the Internet of Things (IoT) revolution, where by 2020 it is estimated that the number of connected and data exchanging devices may exceed 50 billion. These Internet-enabled devices augment the intelligence of systems within a domain of applications, by significantly improving the interaction through advanced sensing, analysis and actuation capabilities. A key challenge from a feedback control viewpoint is the ability to seamlessly integrate new IoT components or modify existing configurations in feedback control settings without having to halt the operation of the system and redesign the overall feedback control scheme.

The control research community has recently recognised the challenge of designing flexible and adaptable control systems that will take advantage of the online evolution of numbers and types of devices in a large-scale system. The work in the framework of this Dissertation exploits technologies from the semantic web domain, for the design of a novel Semantically-enhanced IoT-enabled Intelligent Control Systems (SEMIoTICS) architecture. SEMIoTICS incorporates a supervisor module that integrates a declarative-logic knowledge graph to facilitate the semantic annotation of IoT components and logic-based deductive inference mechanisms for the subsequent continuous and online informed re-configuration of feedback control systems, adapting to the capabilities of available cyber-physical components.

We demonstrate the applicability of SEMIoTICS through illustrative scenarios from the Smart Buildings domain. More specifically, we present how the supervisor is able to support the online (re-)configuration of feedback controllers for the space heating problem in multi-zone buildings. Following a change in the available sensing, analysis and/or actuation capabilities or even in the building envelope parameters, the semantically-enhanced supervisor initiates a process that retrieves stored knowledge about the building and the available control system cyber-physical

components and uses it to configure online a distributed feedback control scheme. In addition, each controller is given the ability to optimally split and allocate the control signal to multiple actuators in a building zone.

Acknowledgments

My journey towards producing this Doctorate Dissertation has been a long yet interesting and much rewarding experience. Thankfully, throughout this journey I was surrounded by people and organizations who provided support at various levels and of various kinds, the whole of it very important for my achievements to date as a worker, as a researcher, and as a human being.

I hereby express my sincere thanks and gratitude to my supervisors, Prof. Marios M. Polycarpou and Prof. Christos G. Panayiotou, for holding my hand during my first research steps, as well as for providing continuous guidance and advices until the last moment of producing my research results; their patience and understanding have played a catalytic role to my personal development.

Being part of the 'KIOS Research and Innovation Center of Excellence" family for the last eight years has been a real inspiration. The research and technical skills, as well as the personality of members of this family are unique. Thank you all for spending time with me in identifying research questions, in social and philosophical discussions, as well as in having fun. Taking the risk of (unintentionally) skipping some of them and by no means assuming any order, I especially thank the rest of the KIOS members with whom I had the chance to collaborate for the production of my PhD research: Elias Kyriakides, Maria Michael, Markos Asprou, Panayiotis Kolios, Stelios Timotheou, Panayiotis Papadopoulos, Vasso Reppa, Kalina Georgiades, Skevi Chrysanthou, Despina Petrou. In addition, I separately express my sincere thanks to Demetris Stavrou and Demetrios Eliades, who made an extra effort coping with me during my down times and helping me to keep getting up after any difficulty. It is my pleasure and relief knowing that we will continue collaborating and pursuing high targets.

Apart from the KIOS family, my PhD journey was supported by many other people. First of all, I acknowledge the contribution and I thank the rest of the

members of my Examination Committee, namely Dr. George Pallis and Dr. Riccardo Ferrari, for taking the time to review my work and provide valuable expert feedback towards securing the high quality of the research outcomes.

My research work and results would not have been undertaken and completed without the financial support by: i) the University of Cyprus, ii) the European Research Council (ERC) under the project ERC-AdG-291508 "Fault-Adaptive Monitoring and Control of Complex Distributed Dynamical Systems" (FAULT-ADAPTIVE); and iii) the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No 739551 (KIOS CoE). I acknowledge this and I feel the responsibility of returning their investment in the form of technological and societal advancements in the years to come.

This PhD Dissertation is dedicated to my parents, Michalis Milis and Eleni Tofa, for showing me the sky and providing me with all necessary tools to touch it, to my brother Marios and his family who have always been there for me, and, more importantly, to my beloved wife Emiliana and our two angels, Marianna and Michalis, for making my life journey so worthwhile and beautiful. Thank you.

Publications

Dissertation Journal Articles

1. G.M. Milis, C.G. Panayiotou, and M.M. Polycarpou, "SEMIoTICS: Semantically-enhanced IoT-enabled Intelligent Control Systems," *IEEE Internet of Things Journal*, 2017, doi:10.1109/JIOT.2017.2773200,
   Available at: http://ieeexplore.ieee.org/document/8106780/.

2. G.M. Milis, C.G. Panayiotou, and M.M. Polycarpou, "Semantically-Enhanced Online Configuration of Feedback Control Schemes," *IEEE Transactions on Cybernetics*, vol. 48, no. 3, pp. 1081-1094, 2017, doi:10.1109/TCYB.2017.2680740 ,
   Available at: http://ieeexplore.ieee.org/document/7891022/

Dissertation Conference Papers

1. G.M. Milis, D.G. Eliades, C.G. Panayiotou, and M.M. Polycarpou, "A cognitive agent architecture for feedback control scheme design," *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Athens, 2016.
   doi: 10.1109/SSCI.2016.7850187

2. G.M. Milis, D.G. Eliades, C.G. Panayiotou, and M.M. Polycarpou, "A Cognitive Fault-Detection Design Architecture," *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, 2016, pp. 2819-2826. doi: 10.1109/IJCNN.2016.7727555

3. G.M. Milis, D.G. Eliades, C.G. Panayiotou, and M.M. Polycarpou, "Semantic mediation in smart water networks," *2015 IEEE Symposium Series on Computational Intelligence*, Cape Town, 2015, pp. 617-624. doi: 10.1109/SSCI.2015.96

4. G.M. Milis, M. Asprou, E. Kyriakides, C.G. Panayiotou, and M.M. Polycarpou, "Semantically-enhanced configurability in state estimation structures of power

systems," *2015 IEEE Symposium Series on Computational Intelligence*, Cape Town, 2015, pp. 679-686. doi: 10.1109/SSCI.2015.104

5. G.M. Milis, C.G. Panayiotou, and M.M. Polycarpou, "Towards a Semantically Enhanced Control Architecture," in *2012 IEEE International Symposium on Intelligent Control*, Dubrovnik, 2012, pp. 1195-1200. doi: 10.1109/ISIC.2012.6398273

Dissertation Unpublished Journal Articles

1. G.M. Milis, C.G. Panayiotou, and M.M Polycarpou, "IoT-enabled Plug-and-Play Control for the Energy and Comfort Performance in Smart Buildings," *IEEE Internet of Things Journal*, 2018. [to be submitted].

Other Related Publications

1. G. Milis et al., "Integrated modelling of medical emergency response process for improved coordination and decision support," *Healthcare Technology Letters*, vol. 3, no. 3, pp. 197-204, 9 2016, doi: 10.1049/htl.2016.0039,
Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7889137&isnumber=7889097

2. G. M. Milis, E. Kyriakides, and A. M. Hadjiantonis, *Electrical Power Systems Protection and Interdependencies with ICT*. Berlin, Heidelberg: Springer-Verlag GmbH Berlin Heidelberg, 2012, pp. 216–228.

3. G. M. Milis, E. Kyriakides, and A. M. Hadjiantonis, "On the complexities of interdependent infrastructures for wide area monitoring systems," in *2012 Complexity in Engineering (COMPENG). Proceedings*, Aachen, 2012, pp. 1-6, doi: 10.1109/CompEng.2012.6242951

4. P. Kolios, G. Milis, C. Panayiotou, T. Staykova and H. Papadopoulos, "A resource-based decision support tool for emergency response management," in *2nd International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, Rennes, 2015, pp. 159-165, doi: 10.1109/ICT-DM.2015.7402032

5. J. Kantorovitch, A. Giakoumaki, A. Korakis, H. Papadopoulos, G. Milis, P. Kolios and T. Staykova, "Knowledge modelling framework," in 2nd Interna-

tional Conference on Information and Communication Technologies for Disaster Management (ICT-DM), Rennes, 2015, pp. 145-151, doi: 10.1109/ICT-DM.2015.7402037

Contents

List of Figures

List of Tables

Main Notation

x A vector of the states of a plant/system

w A vector of uncontrolled inputs (disturbances) to a plant/system

y A measurement vector associated with a plant/system

y' A vector of processed measurements of plant states

u A vector of control decisions aiming to drive the inputs to a plant/system

u' A vector of processed control decisions

v A vector of controlled inputs to a plant/system

ζ A parameters vector

r A state-reference vector associated with a plant/system

$\mathcal{K}$ The set of discrete time samples of the continuous signals at 1-sec intervals.

$k \in \mathcal{K}$ An index of the discrete time samples of a certain period of time

$k_I, I = 0, 1, 2, ...$ Times at which events happen, which change the availability of IoT components in the plant.

k_I^+ The discrete time samples for $k_I < k < k_{I+1}$

$p, s, y, c, u, a, \zeta, e, \theta$ When used as superscripts, they help differentiating between variables referring to the models of plants, sensors, pre-control functions, controllers, post-control functions, actuators, parameter-functions, state-estimators and approximation functions respectively

$f^p, f^s, f^y, f^c, f^u, f^a, f^\zeta, f^e, f^\theta$ Functions representing the models and/or implementations of plants, sensors, pre-control functions, controllers, post-control functions, actuators, parameter-functions, state-estimators and approximation functions respectively

$\zeta^p, \zeta^s, \zeta^y, \zeta^c, \zeta^u, \zeta^a, \zeta^\zeta, \zeta^e, \zeta^\theta$ Parameter vectors associated with the implementations of plants, sensors, pre-control functions, controllers, post-control functions, actuators, parameter-functions, state-estimators and approximation functions respectively

$\mathcal{F}$ The set of all control system components each time available in the system/-plant

$\mathcal{F}^p, \mathcal{F}^s, \mathcal{F}^y, \mathcal{F}^c, \mathcal{F}^u, \mathcal{F}^a, \mathcal{F}^\zeta, \mathcal{F}^e, \mathcal{F}^\theta$ Subsets of the set $\mathcal{F}$, with elements the control system components of type plants, sensors, pre-control functions, controllers, post-control functions, actuators, parameter-functions, state-estimators and approximation functions respectively respectively

$f_J^I, I = 0, 1, 2, \ldots$ and $J = 1, 2, \ldots, n_I$ The J-th of the n_I control system configuration able to offer the required control service to the plant following the changes at time k_I.

σ_I The configuration selection decision, which activates one of the configurations, following the time k_I

Σ The Semantically-enhanced Supervisor module of SEMIoTICS

$\mathcal{G}_I$ The state of the declarative-language-based "Knowledge Graph" that hosts the available experts' knowledge at time k_I

$\mathcal{S}_I$ The state of the set of feedback-control (semantic) specifications given to the function (e.g., the characteristics of the desired state) at time k_I

C The space of all possibilities for configuration decisions

C_I The set of all valid configurations of the feedback-control system at time k_I - The Configurations Graph at time k_I

$C_{(I,J)}$ A sub-graph of the Configurations Graph C_I, which defines the configuration options $J = 1, \ldots, n_I$

$\mathcal{L}$ The set of plant's features of interest (locations)

$\mathcal{Q}$ The set of physical properties associated with the defined features of interest (locations)

$\mathcal{P}$ The set of components' capabilities and expected effect on plant's states, inputs, parameters, etc.

$\mathcal{M}$ The set of measurement units associated with the defined properties of features of interest

$\mathcal{U}$ The set of all inputs defined by available components

$\mathcal{Y}$ The set of all outputs defined by available components

$\mathcal{Z}$ The set of all parameters defined by available components. It also acts as a "Parameters Registry"

Λ The semantic annotation space, defined by four dimensions: $\Lambda \equiv \mathcal{L}\times\mathcal{Q}\times\mathcal{P}\times\mathcal{M}$

$\lambda(\cdot)$ The Semantic Annotation operation

Chapter 1

Introduction

1.1 Motivation

Today's engineered systems more and more employ cyber and physical components with advanced communication capabilities [6], such as sensors for monitoring system properties, electrical and mechanical actuators, controllers and other software tools implementing algorithms for data and signal processing. This trend is becoming even more prominent due to the emerging level of maturity of the Internet of Things (IoT) paradigm and the subsequent proliferation and high penetration of smart portable and embedded devices connected to cyber-physical systems and networks. Due to this IoT revolution, it is estimated that by 2020 the number of connected and data exchanging devices may exceed 50 billion [34], suggesting a future where machines will interact with each other and the environment in a context-aware framework [67]. IoT components augment the intelligence of systems within a domain of applications through their capability to exploit (wireless) Internet connectivity and use standard communication protocols, e.g. MQTT [83]. They are able to cover a broad range of functionalities, from observing and measuring properties of physical features of interest, to processing collected data and information, to making and/or supporting decisions, to acting and affecting these properties after receiving appropriate instructions.

Large-scale smart buildings are good candidate consumers of IoT components, due to their advanced monitoring and control requirements. Fig. 1.1 shows a top-down view of a large-scale commercial building. We spend a big part of our lives in such buildings and we require them to offer a comfortable and safe environment,

and in addition operated in a cost-effective manner. Therefore, many large-scale buildings are already equipped with Building Automation Systems (BAS) and subsystems that monitor and control air quality, lighting, water distribution, security, as well as heating, ventilation and air-conditioning (HVAC). Building owners/operators typically purchase the systems from certain integrators that use equipment from certain vendors, most times vendor-specific and technology-specific. The integrators are responsible to appoint a team of technical persons to perform the installations of fixed types and numbers of devices in the building, as well as train a team of operators to maintain a set of pre-programmed functions and generated reports.

Current solutions for monitoring and feedback control applications typically assume pre-deployed sensing and actuation devices, as well as pre-designed feedback control intelligence. However, the IoT ecosystem is rapidly evolving and IoT-enabled equipment/components, including mobile devices and virtual ones running through a computer terminal or on the cloud, can take advantage of the readily available wireless internet infrastructure and be deployed and/or removed online, thus modifying the available sensing, actuation and signal processing capabilities in general as shown in Fig. 1.1. The IoT components may either be ordered, installed and maintained by experts within a relevant upgrade plan or they may be installed by occupants following ad-hoc purchases of IoT-enabled devices or they may become available at run-time through the sensing infrastructure on the mobile phones of building occupants.

Assume a broken lighting sensor; the lighting control system could make use of a luminosity sensor from an Android device of an occupant, e.g. to turn on the lights when they are off yet the luminosity level appear to be low for the needs of the occupants in the room. The HVAC could use information from the local weather station or it could make use of a window opening measurement from the security system to improve its decision making. Also, an HVAC system may utilize a newly installed electric heater to supplement its actuation capacity to effectively control the temperature considering cost-optimality as well. A temperature regulation system could be reconfigured automatically so as to utilize the output of a newly installed occupancy sensor to lower the reference value of the zone temperature when the zone is not occupied, thus saving energy. The security system could detect occupants' presence in a room using information from a CO_2 measurement installed for the air-quality system. Such flexibility is not provided by current control systems and

it is clearly not practical or economical to have control engineers on-call, to design new feedback control loops whenever a change in the components occurs. Building operators will expect their control systems to exploit such seamless intelligence; i.e., be able to utilize the new measurements, as well as utilize the processing and actuation capabilities online, towards performing better against certain Quality of Service (QoS) criteria related to occupants' comfort, energy efficiency, quality of the air, robustness of the solution, and so on.

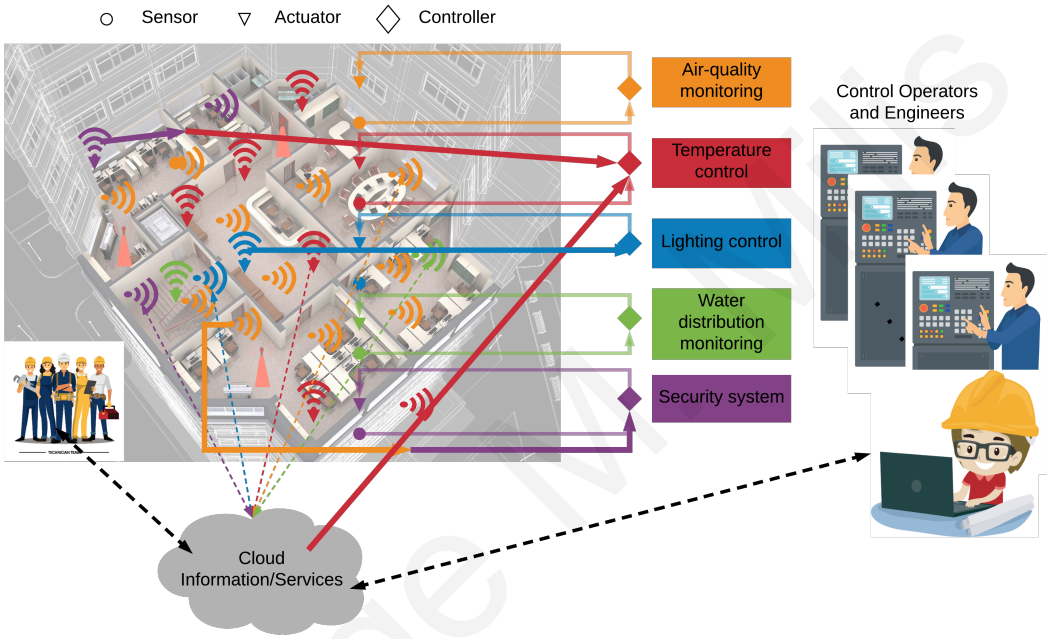


Figure 1.1: A top-down view of a large-scale commercial building, equipped with several control systems for air-quality monitoring, temperature regulation, lighting control, water distribution monitoring, security system. Heterogeneous static and mobile sensors and actuators are deployed in several locations of the building. The systems are maintained by technicians' teams and control operators/engineers. Devices and human operators are able to communicate through cloud services.

The control community has recently identified the above research challenge [107]. The cited report identifies the need to build interoperable components (sensors, models, algorithms, actuators) that will exhibit plug-and-play capabilities, suggesting the exploitation of technologies that deal with online discovery and composition of services, from the Web domain [84]. The objective would be for control applications to deal with dynamic compatibilities between cyber-physical components. To respond to the above described challenges and build the prescribed flexibility, new control

system architectures are required, able to facilitate the online recognition of devices, the acquisition of their capability characteristics and finally the self-reconfiguration of the control system to incorporate the new capabilities. This is an important aspect for the practical adoption of IoT devices in monitoring and control applications of large-scale systems [21], where the topology of the systems is too complex to handle with traditional monolithic architectures. The smaller components will be enabled to work autonomously to accomplish certain tasks in some part of the system and co-operate effectively with the rest of the system, so as to address higher-level monitoring and control challenges [116].

1.2 Dissertation Thesis Statement and Contributions

Thesis Statement: The configuration or reconfiguration needs of feedback control loops in certain classes of large-scale systems, where the availability of sensors, actuators, controllers and other signal processing functions changes dynamically over time, is effectively automated through the use of ontology-based knowledge models and deductive inference techniques (see Section 2.1), which facilitate the online management of information about the components, the storage of knowledge about feedback control engineering, as well as the implementation of necessary reasoning algorithms. This functionality becomes available through the design of a Supervisor system which undertakes to communicate with all deployed components, as well as with human operators and cloud services, "understand" what sensing, actuation, processing and control capability is available in the system and "think" on behalf of the operators/engineers to appropriately re-configure all feedback control loops. In addition, the resulting module facilitates the offering of additional flexibility and intelligence by synthesizing online control schemes for certain applications.

The above "Thesis Statement" is proved in the smart building domain with the design of the "Semantically-Enhanced IoT-enabled Intelligence Control Systems (SEMIoTICS)" module as shown in Fig. 1.2 on the right. SEMIoTICS adds a middle layer between the human operators, e.g. control engineers and the feedback control components deployed in the system.

More specifically, the contributions related to this Dissertation are summarized

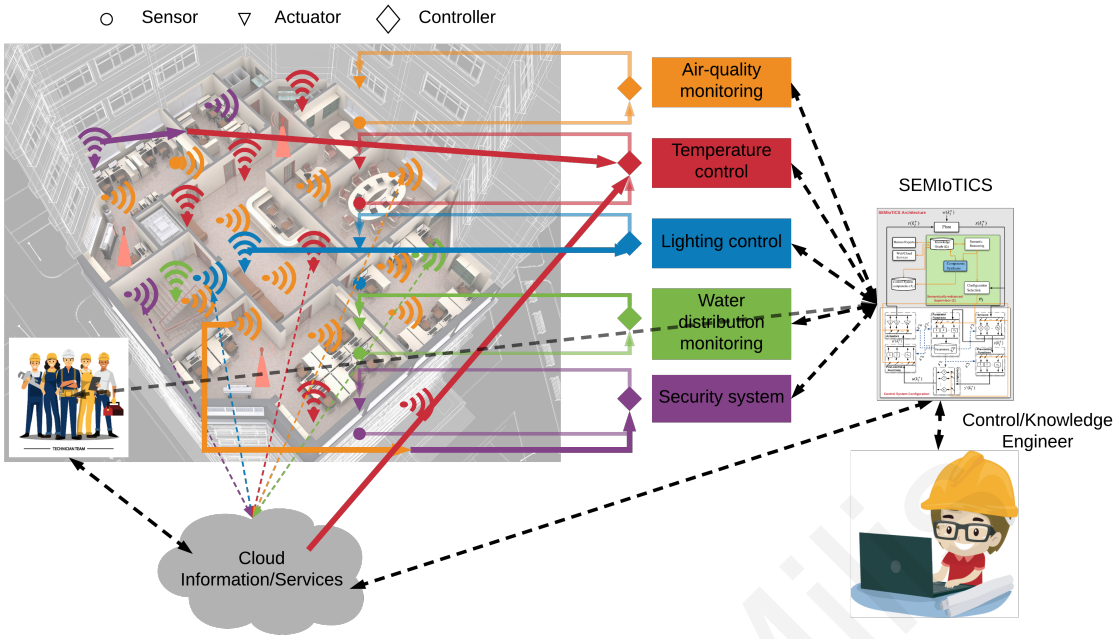


Figure 1.2: The SEMIoTICS module acts as a middle layer between the control operators/engineers shown in Fig. 1.1 and the (physical or cloud-based) feedback control components in the building

as:

- Design and development of the *Semantically-enhanced IoT-enabled Intelligent Control Systems (SEMIoTICS)* reference architecture and system [77, 80]:

  SEMIoTICS is a novel, logical-inference-based system architecture and method, applicable to a class of dynamical systems. SEMIoTICS addresses in general a variable structure control problem, assuming that there is no pre-designed single controller that poses all required adaptation capabilities to learn and accommodate the changes in the controlled system and/or the control system components (e.g., in case of replacing a faulty sensor). It incorporates a supervisor that uses ontological knowledge graphs to model the expert knowledge about individual components in a consistent way, so as to be exploitable by machines. It subsequently performs deductive (semantic) reasoning to decide the online composition/re-configuration of feedback control loops, thus addressing the design-operation continuum challenge, as recognised by the CPSoS EU project with relation to the cyber-physical systems of systems [24]. The control system switches online between different configurations when components are plugged-in. The switching among extracted control system configurations is

decided and enforced through a discrete-state logic-based deductive reasoning process [9]. This mechanism assumes that the available control modes (configurations) are not defined in advance; they are configured online using components from an evolving database of control system components, able to interoperate in a modular architecture but without hard-coded interfaces. At discrete events, the logic-based system exploits modelled knowledge and subsequent (standards' based) semantic characterisations of components' interfaces, as well as deductive inference rules to produce the switching signal. The knowledge is stored in "knowledge graphs" formed by meaningful relations between modelled knowledge entities of pre-defined types [39,104]. We apply SEMIoTICS in the Smart Buildings domain to facilitate the re-configuration of a heating control system where various IoT components are gradually deployed and incorporated online.

- Online synthesis of distributed feedback control schemes in the Smart Buildings domain, with optimal control signal allocation capability to drive multiple heating devices:

  A core benefit from SEMIoTICS is the inherent flexibility to exploit redundancy of "Things" capabilities by deciding and implementing alternative configurations of a specific feedback control system. The latter is a feature that can be deliberately exploited by control system engineers to design components that adhere to relevant technical and QoS specifications. We designed a method that allows the online synthesis of a distributed scheme of model-based controllers (one controller per building zone), which are able to retrieve existing building knowledge online and allow the plug-and-play temperature control. In addition, the controllers are designed so as to be able to split the control signal into multiple signals driving additional heating devices in the zone they control. The control scheme provides functionality which is specific to an application domain, a class of systems and/or a set of control problems. In our work, we demonstrate this feature for the heating control in multi-zone buildings, with the aim to improve the performance against energy savings and occupant's thermal comfort criteria. Similar SEMIoTICS-compatible components can be designed for other types of plants and domains.

- Horizontal and vertical applications of SEMIoTICS in Critical Infrastructure

Systems [76,78,79]:

We show the customisation and transferability of SEMIoTICS to certain use cases in Critical Infrastructures such as the Water Distribution Networks (WDNs) and the Electric Power Systems (EPSs), as well as in vertical research areas such as the fault detection schemes (FDSs).

First, we present the application of SEMIoTICS to the online (re-)configuration of state estimation structures in EPSs; this use case involves changing the domain knowledge to cover the EPS, customizing the semantic reasoning and configuration selection processes to consider the EPS state estimation and using components relevant to the domain and addressed problem. Then, we present the application of SEMIoTICS to the online (re-)configuration of the tank-head feedback control scheme in WDNs; the use case involves changing the domain knowledge to cover the WDN and using feedback control components relevant to the domain and addressed problem. Finally, we present the application of SEMIoTICS to the contamination event detection in WDNs; in addition to the domain knowledge change introduced with the previous use case, this use case involves the re-design of the reasoning and configuration selection processes to consider components from the fault detection area.

1.3 Dissertation Outline

The Dissertation is organized as follows: Chapter 2 discusses first the history of logic theories and how the community ended up with first-order logic and declarative languages used in our work for deductive reasoning. It then presents a component-based analysis of a feedback control system, which is used as the basis for SEMIoTICS semantic models and reasoning. Finally, it offers insights into the current state-of-the-art in the research areas related to the Dissertation work. Chapter 3 details the work related to our first contribution. Specifically, the Chapter starts with a general formulation of the problem being considered, with reference to the proposed novel architecture. It then provides a detailed description of the design of SEMIoTICS modules and processes. It defines the "Knowledge Graph" and the respective semantic annotations of individual control system components, as well as the pairwise composition models in a feedback loop orchestration. It then defines and details the

"Semantic Matching", the "Semantic Reasoning", and the "Configuration Selection" operations after building the "Configurations Graph". It continuous with discussing the scalability aspects of the SEMIoTICS semantic reasoning and configuration selection processes and ends with presenting the implementation technical details of SEMIoTICS. The second contribution of our work is detailed in Chapter 4, which presents the design of SEMIoTICS-compatible controllers that are synthesized and plugged online and assume the heating control in a multi-zone building. The controllers deal also with cost-optimal ways of splitting and allocating the control signal to multiple actuators in each zone. Chapter 5 details the work related to the third contribution, by presenting the exploitation of SEMIoTICS in Critical Infrastructures, such as the Water Distribution Networks and the Electric Power Grid, dealing with state estimation, control and fault-detection problems. Finally, Section 6 concludes the Dissertation and discusses the overall impact of the work and promising future directions.

Chapter 2

Background Knowledge and State-of-the-art Overview

2.1 Logic Theories, Deductive Inference and Declarative Languages

The study of "Logic" has its roots in the Aristotelian theory of syllogism (deduction) [101]. According to Aristotle, a syllogism is "*speech (logos) in which certain things have been supposed, different from those supposed results of necessity because of their being so*". The "things supposed" comprise the premise (protasis), while the "results of necessity" comprise the conclusion (sumperasma), that is, the logical consequence. For instance, a statement C is the consequence (result of necessity) of the fact that A and B being true (they are supposed), if its logical truthfulness cannot be challenged given the truthfulness of the things supposed.

The Aristotelian theory of logic had remained dominant until the mid 19th century, followed by approximately one hundred years of research that led to the conception of "modern logic". During that transformation time, G.E. Moore and B. Russel had put effort in formalising logic by referring to "common sense" [82, 106], then they started adding bounds and discussing the "set theory" as the foundations for mathematics. During early twentieth century, the bounds became narrower following the realisation of the "paradoxes of logic" and the work of B. Russel on "Types theory". These led to the establishment of "First-Order Logic (FOL)", otherwise called the "modern logic", which had not happened before the mid 20th century [39].

2.1.1 The road towards modern logic

During the second half of the 19th century, logic was studied within the framework of the classical philosophy, as well as in relation with the science of mathematics; algebra, axiomatics and analysis. Since the late 19th century the theory of logic started becoming a blend of the propositional and predicate logic (`http://www.iep.utm.edu/prop-log/`) on one hand and the theory of sets and relations on the other hand. At that time, "thinking" was considered the ability of mind to relate between "things", i.e., derive conclusions about properties of certain things based on knowledge about properties of other things. This process of "thinking", performed by the mind, is essentially a mapping between pre-stored knowledge about things. Based on this understanding, B. Russell [105] formed the "symbolic logic" within the areas of "calculus of propositions", "calculus of classes" and "calculus of relations". B. Russel, together with Peano and Frege agreed in their separate works that mathematics were actually "symbolic logic", i.e., the symbols are used to help expressing logical propositions and relations between things.

More specifically, at the beginning of the 20th century, the work of Russell, Peano and Frege considered the logical systems within the bounds of the theory of sets. They had believed in the principle of "comprehension", meaning that a thing/object could be fully described by certain attributes, properties, qualities, etc. However, later this theory faced certain contradictions (or paradoxes or antinomies), with the most significant being the "Russell's paradox", which was formulated in terms of notions of the set theory itself, i.e. the notion of negation and membership. The discovery of the paradoxes, led to the revision of the foundations of set-theory and its effect on logic and mathematics. In order to resolve the issues revealed by the paradoxes, Russell developed his "Theory of Types", which was incorporated into his work on "Principia Mathematica", co-authored with Whitehead [105]. The theory of types [104] introduced the notion of "types" of elements of a set. This way, the notion of "comprehension" was preserved. According to van der Waerden, in his Moderne Algebra [28], the "Theory of Types" was then the most important system of logic. However, in 1931, Godel announced his first "incompleteness theorem" [47].

During that time (1930), the early work of Skolem [110] on the Zermelo system and the work of Von Neumann on his logic system [85], led to the appearance of "First Order Logic (FOL)" as the basic system for logic. Hilbert & Ackermann's

"Grundzuge der theoretischen Logik" [26] shows that FOL was studied as a separate system of logic by around 1928. This book shows also that the use of higher-order logic would be possible and necessary when one would require to analyse meta-concepts of mathematics. FOL was adopted as the natural system of logic by 1950, adhering to: i) The principle of deduction (conditions of arguments to be correct following analysis of their validity); ii) The principle of universality (logical sentences formed independently of specific topic); iii) The "Kant's principle" (study the formulation of arguments and deductions using logical variables and not instantiations within a domain of discourse; iv) The Leibnizian ideal (compute logical deductions with machine algorithms). The latter was first achieved by Boole with his algebra of logic. Then, Frege made an impressive step forward and Godel followed by establishing certain implementation limitations.

The concept "domain of discourse" was defined by G. Boole in 1854 in his work "Laws of Thought", as:

"*In every discourse, whether of the mind conversing with its own thoughts, or of the individual in his intercourse with others, there is an assumed or expressed limit within which the subjects of its operation are confined. The most unfettered discourse is that in which the words we use are understood in the widest possible application, and for them the limits of discourse are co-extensive with those of the universe itself. But more usually we confine ourselves to a less spacious field. Sometimes, in discoursing of men we imply (without expressing the limitation) that it is of men only under certain circumstances and conditions that we speak, as of civilized men, or of men in the vigour of life, or of men under some other condition or relation. Now, whatever may be the extent of the field within which all the objects of our discourse are found, that field may properly be termed the domain of discourse...*"

In summary, it can be deduced that around 1900 logic was conceived as a theory of sentences, sets and relations; almost until 1930 the established logic system was (simple) type theory, while by 1950 FOL became the paradigm logical system.

2.1.2 First Order Logic

In general, a first-order language system assigns variable symbols to logical constants. The language also defines the "domain of discourse", which specifies the range of the variables, in line with the Russel's Theory of Types. Sentences written using such languages have clear semantics. For instance, assume the domain of

discourse $\mathcal{D}$, being a set of objects of a certain type. An example of a first-order statement within this domain could be the $x, a(x)$, stating that some logical sentence x is true and at the same time it's logical transformation through the predicate $a(x)$ is also true. If a domain of discourse is not clearly defined, the logical derivations may not be always feasible. For instance, the sentence $\forall x, x > y$ cannot be logically interpreted without knowing the domains of the logical variables x and y. So, if x and y are real numbers, the statement is false because there are real numbers that can be greater than other real numbers. But, if x is a positive number and y is a negative number, the statement becomes always true since all positive numbers are greater than any negative number. Moreover, assuming $\mathcal{D}$ being the domain of building zones, we can define the adjacency of two building zones as a binary predicate $a(\cdot)$, taking two building zones are arguments and returning $\top$ if the two zones are indeed adjacent or $\perp$ if they are not.

If the validity or not of any given sentence can always be deduced through some language system, then this language is said to be "complete". According to Godel, the FOL language system is complete, although mathematical systems cannot guarantee in general syntactic and semantic completeness [47]. This understanding made FOL a sufficient language for codifying mathematical proofs. Moreover, the tools utilised in mathematical proofs (theorems, sentences, etc.) can be explicitly supported by quantification expressions such as the "given any", "there is", "for any", "for each". For instance, the sentence: $\forall x(a(x) \mapsto b(x))$ may be written.

Summarizing, to guarantee completeness, a FOL language always needs to define a "domain of discourse" $\mathcal{D}$, also defining the classes Ω of all terms used in the language. For instance, an n-ary relation r is essentially a mapping of the form $\Omega(r)$: $\mathcal{D}^n \mapsto \{\text{true}, \text{false}\}$. That is, a statement is true if is can be inferred by some logical deduction within the domain $\mathcal{D}$. In mathematics, a formula is logically valid only if it is true no matter the instantiation of the variables in the defined domain of discourse. Certain "axioms" can be also defined within a domain of discourse, as being true anyway and helping the inference.

2.1.3 Syllogism - Deductive Inference Systems

Basic systems of "Logic" deal with certain logical "statements" or "sentences" being true or false. The logical statements under consideration are typically called

"propositions" (see also the terms "propositional logic"or "sentential logic" or even "zero-order logic"). Logical statements can be combined using "logical connectives". For instance, the "and" (conjunction), the "or" (disjunction), the "not" (negation) and the "if" (denoting condition of existence) are logical connectives of the English language. Moreover, "logical axioms" can be pre-defined, stating things that are believed to be true by design. A third element of logic systems are the "inference rules", which specify given knowledge about the truthfulness of combinations of propositions through logical connectives. In logic, an "inference rule" is a logical n-ary function that takes as input certain statements, analyzes their syntax and returns a conclusion. For example, in the "modus ponens" inference system, an inference rule takes two inputs, one statement in the form "if a then b" and another in the form "a", and returns the conclusion "b". A fourth element are the "quantifiers", which are operators that map propositions (their symbolic representation) to a specific domain of discourse.

It has been mentioned earlier, that according to the Kant's principle, only the form of the statements should matter and not the instances of the logical variables. We take an example of a syllogism from Aristotle: *All men are speakers*. *No oyster is a speaker*. Therefore, *no oyster is a man*. This example can be transformed to the first-order sentence: $\forall a, b(a)$. $/b(c)$. Therefore, $b \neq c$. Another example would be: **Statement 1**: IF <high temperature> THEN <feeling hot>. **Statement 2**: <feeling hot>. **Deduction**: <high temperature>. This example can be also transformed to FOL as: **Statement 1**: $a \rightarrow b$. **Statement 2**: b . **Deduction**: a. The given statements are what we know is true (knowledge facts). Other knowledge can be inferred by the given statements after applying some inference mechanism. The knowledge facts and the inference rules are considered known a-priori in deductive inference, either because they are part of the expect knowledge or because they can be derived analytically using a-priori knowledge of the system.

Therefore, the concept "Deductive Inference" is concerned with checking whether a statement is true given the truthfulness of another statement or a combination of statements through logical connectives. There are many such systems for first-order logic, including the "Hilbert-style deductive systems", the "natural deduction", the "sequent calculus", the "tableaux method", and the "resolution". It is noted that derivations of proofs in "proof theory" are essentially deductions . A deductive inference system is considered "sound" if any statement that can be derived in

the system as true is logically valid. Conversely, a deductive inference system is "complete" if every logically valid statement can be derived through the system. All above mentioned deductive inference systems are both sound and complete. The conclusions in these deductive inference systems do not typically consider the semantic interpretations of the statements in a domain of discourse. On the other hand, in FOL, deductive inference is only semi-decidable, that is, if a logically implies b then this can be deduced by a deductive inference system. However, if a does not logically imply b, this does not mean that a logically implies $\not b$.

In general, the deductive inference systems use several logical and non-logical symbols from the alphabet, such as:

- The quantifier symbols $\forall$ and $\exists$
- The logical connectives: $\wedge$ for conjunction, $\vee$ for disjunction, $\Rightarrow$ for implication, $\Leftrightarrow$ or $\equiv$ for biconditional, $\neg$ or˜for negation, etc.
- Parentheses, brackets, and other punctuation symbols.
- An infinite set of variables, often denoted by lower-case letters at the end of the alphabet $x, y, z, \cdot$. Variables are often distinguished by sub-scripts: $x_0, x_1, x_2, \cdot$.
- An equality symbol, e.g., =
- The truth constants are included, e.g., $\top$ for "true" and $\bot$ for "false".
- Additional logical connectives that may be required, such as "NAND" and "exclusive OR".

On the other hand, non-logical symbols represent semantic relations, i.e. mappings of knowledge objects within a domain of discourse. In the past, FOL considered a single fixed set of non-logical symbols to apply deductive inference. The current practice, however, is to define a different set of non-logical symbols within the bounds of an application. Such definition of logical symbols can be made through "ontology languages". Ontology languages are essentially collections of a finite number of n-ary predicates/symbols, representing relations between n objects. For example, man(x) is an example of a predicate of arity 1, which defines that "x is a man"; father(x, y) is a predicate of arity 2, interpreted as "x is the father of y".

For completeness, we note here that there are also other types of inference: i) "Inductive inference", where the rules are not considered known a-priori; possible rules are derived from the propositions and the observations/experiences, as if the logic system is "identified" from inputs and outputs; ii) "Abductive inference", where the observations/experiences as well as the rules are considered known, and the logically valid truths are inferred accordingly. E.g., I observe something (fault detection) and knowing how the system works I try to infer what might have caused it (fault isolation).

2.1.4 Ontology languages and Description Logic

Ontology languages allow the encoding of pre-existing or acquired knowledge about a specific domain of discourse. They typically include also inference (or "reasoning") rules that help with making logical conclusions above the modelled knowledge. Ontology languages are usually declarative languages and are commonly based either on FOL or on "Description Logic" (DL) [9]. DL is a family of formal knowledge representation languages, which are typically subsets of FOL (less expressive than FOL). They are used to represent a domain of discourse in a structured way, offering clear decidability with no gaps due to adding constraints. DLs usually use binary predicated (two-variable logic). There are enough good-quality reasoning mechanisms (decision procedures) designed and implemented for DLs.

DLs have been used as the logical formalism for "ontologies" and the "Semantic Web". For instance, the Web Ontology Language (OWL) and its profile is based on DLs [60]. In DLs usually a "unary predicate" of FOL, is called a "class", a "binary predicate" is called a "property" and a "constant" is called an "individual". DLs pose certain relationships with other logics as well. For instance, Fuzzy description logic combines fuzzy logic with DLs. Fuzzy logic deals with the notions of vagueness and uncertainty about the classes of certain logical variables. These properties are common in intelligent systems, where concepts do not have clear boundaries. Fuzzy logic therefore generalises the description logic to deal with vague concepts. In addition, the "Temporal Description Logic" allows reasoning about time dependent concepts and can be the combination of DL with a modal temporal logic such as "Linear Temporal Logic". Concluding, DLs are a very good tool for representing knowledge and inference rules, and subsequently inferring mappings and values

from known facts. Examples are given in Section 3.3.1 where the SEMIoTICS knowledge graph is discussed.

2.2 Background on Feedback Control System Components

In general, a feedback control system decides about the action to be applied on a plant via its inputs. In a discrete-time implementation, where $\mathcal{K}$ is the set of discrete time samples of the continuous signals at a certain sampling rate and k is an index of this set, the plant's input $v(k)$ is a vector of signals produced by the actuators at time $k \in \mathcal{K}$. A feedback control system implementation, can be considered as consisting of sub-components, some of which are basic (mandatory) for all implementations of feedback control systems while others are required only in certain cases. These components are reviewed in the sequel.

2.2.1 Plant dynamics

First of all, a control system is always implemented to offer a service to a specific plant. The dynamics of the plant, adopting a discrete time formulation, are generally described by (2.1).

$$x(k+1) = f^p\left(x(k), v(k), w(k), \phi(k), h(k); \zeta^p(k)\right) \tag{2.1}$$

where $x(k) \in \mathcal{R}^{n_x}$ is the vector of state-variables of dimension n_x, describing the plant, $f^p(\cdot)$ is a function representing the plant's dynamics, $x(k+1)$ is the value of the state at next time sample, $v(k) \in \mathcal{R}^{n_v}$ is the input signal of dimension n_v produced by controlled actuators, $w(k) \in \mathcal{R}^{n_w}$ is the input signal of dimension n_w produced by uncontrolled sources, $\phi(k)$ is a signal modelling faults introduced in the plant's dynamics, $h(k)$ is the input signal produced by third interdependent systems and ζ^p is a vector of other parameters related to the plant dynamics.

2.2.2 Actuator

Then, a first component of a feedback control system is the "Actuator", given by (2.2).

$$v(k) = f^a\left(u(k); \zeta^a(k)\right) \tag{2.2}$$

where $v(k)$ is the plant input signal discussed earlier, $f^a(\cdot)$ is the implementation of a function that produces the signal acting on the plant, $u(k)$ is a computed signal that drives the action and ζ^a is a vector of other parameters required by the available actuation implementation.

2.2.3 Controller

A second component in a feedback control system is the "Controller" given by (2.3).

$$u(k) = f^c\left(y(k), r(k), \hat{x}(k), \hat{f^p}(k); \zeta^c\right) \tag{2.3}$$

where $u(k)$ is the control decision signal defined earlier, $f^c(\cdot)$ is the control algorithm that derives the signal, $y(k)$ is the signal representing the plant's output as given to the controller, $r(k)$ is the plant's state reference trajectory, $\hat{x}(k)$ is the estimated plant state (optionally used), $\hat{f^p}(k)$ is the estimated value of unknown plant dynamics (optionally used) and ζ^c is a vector of parameters required by the adopted controller implementation.

In practice, the controllers can range in complexity from a simple ON/OFF control to PID control to adaptive control algorithms, etc. For instance, the control function $f^c(\cdot)$ may be a *bang-bang* controller that compares the measured states $y(k)$ with the desired states $r(k)$ and returns a vector of binary signals indicating whether the measurements are greater than the desired values or not. Another example may be a fuzzy control implementation, where the control decision is the defuzzification of a linguistic value, e.g., "fast", which was the output of the triggering of a set of fuzzy rules on the fuzzified plant output [35].

2.2.4 Sensor

A third component of a feedback control system is the "Sensor", which undertakes the task of measuring the state of the plant and is given by (2.4).

$$y(k) = f^s\left(x(k), v(k), w(k), \phi(k), h(k); \zeta^s\right) \tag{2.4}$$

where $y(k)$ is the signal produced by the installed sensing devices, $f^s(\cdot)$ is the model of a sensor device, $x(k)$ is a vector of the plant states, $v(k), w(k), \phi(k), h(k)$ are the various types of plant's input vectors discussed earlier and ζ^s is a vector of parameters required by the adopted implementation of the sensing. For instance, the sensing parameters may correspond to measurement accuracy given by the manufacturer or the location of the device derived from expert knowledge about the system operation, etc.

In summary, at a minimum, the feedback control system is composed of the components specified above, forming a composite function $f^a \circ f^c \circ f^s$. That is, the input to the plant is a function of the control decision which in turn is a function of the plant's (state) measurements.

2.2.5 State Estimator

In addition to the above described components, additional ones may be required by certain feedback-control systems. For instance, the estimation signal of the plant states, $\hat{x}(k)$, may be computed by a separate component. In that case, a "State-Estimation" component can be considered, given by (2.5).

$$\hat{x}(k) = f^e\left(\hat{x}(k-1), \hat{f^p}(k), y(k), u(k); \zeta^e\right) \tag{2.5}$$

where $\hat{x}(k)$ is the estimated plant states signal at the current discrete time step, $f^e(\cdot)$ is a state-estimation function, $y(k)$ and $u(k)$ are the vectors of plant's measured output and known control signal respectively, ζ^e is a vector of other parameters required by the adopted implementation of the state estimator, $\hat{x}(k-1)$ is the vector of estimated previous values of the plant state, and $\hat{f^p}(k)$ is the estimated value for unknown plant dynamics (if required). For instance, the State-Estimation module may correspond to a "Kalman filter" which produces estimates based on some prior knowledge about the plant states, a measurement vector and certain parameters of measurement and state's uncertainty; it can also be a "Luenberger observer" which, based on a known model of the plant dynamics and the available measurements, produces estimates of the plant state.

2.2.6 Learning Component - Approximator

Furthermore, in the case of having a plant model with unknown dynamics $f^p(\cdot)$ (or part of the $f^p(\cdot)$), a "Learning Component" can be utilized, to learn the unknown function using a suitable approximation structure (e.g., neural network, polynomial function, radial-basis functions, wavelets, etc.), such that $\hat{f}^p(\cdot)$ approximates $f^p(\cdot)$. This component can be described in general by (2.6).

$$\hat{f}^p(k) = f^\theta\left(y(k), u(k); \zeta^\theta\right) \tag{2.6}$$

where $\hat{f}^p(k)$ is the approximated value of the unknown function, $f^\theta(\cdot)$ is the adopted online learning implementation and ζ^θ are any other parameters required by the adopted implementation (e.g., the convergence rate, knowledge about the structure of the function) [35].

2.2.7 Pre-Control Processing Function

In some cases, the measured plant output needs to be processed by a separate component before being fed to the controller. For instance, if a plant state is measured by more than one sensors, we may want to fuse the measurements and use the computed signal in the controller; alternatively this could correspond to data validation/reconstruction. This step is undertaken by a "Pre-Control Function", defined as in (2.7).

$$y'(k) = f^y\left(y(k); \zeta^y\right) \tag{2.7}$$

where $y'(k)$ is the processed plant output, $f^y(\cdot)$ is the adopted measurement processing implementation, $y(k)$ is the actual sensor measurements vector and ζ^y are any other parameters required by the adopted processing implementation (e.g., knowledge about the proximity of devices to the state location). Then, the controller implementation $f^c(\cdot)$ receives as input the signal produced by the function $f^y(\cdot)$ instead of the actual measurement.

2.2.8 Post-Control Processing Function

Similarly, the output of the controller, $u(k)$ may need to be processed before fed to the actuators. For instance, consider the case where a single control signal needs to

drive two actuators. This can be implemented by a "Post-Control Function", given in (2.8).

$$u'(k) = f^u\left(u(k); \zeta^u\right) \tag{2.8}$$

where $u'(k)$ is the processed control decision, $f^u(\cdot)$ is the adopted control signal processing implementation, $u(k)$ is the actual control signal and ζ^u are any other parameters required by the adopted processing implementation (e.g., knowledge about the type of actuation devices). Then, the actuators receive the signal $u'(k)$ instead of the control signal $u(k)$.

2.3 State-of-the-Art

A lot of work has been undertaken to date by the control community towards designing control algorithms with online adaptation capabilities that aim to facilitate the flexibility of the control system to accommodate system uncertainties and/or time variations. For instance, approaches to designing fault-tolerant control systems are presented in [13], while the textbooks [8] and [63] comprise comprehensive information sources on approaches to design adaptive controllers. Combining adaptation capabilities with online learning of unknown system dynamics, has been also addressed in research; the textbook [35] discusses general methodologies for designing adaptive approximation-based control systems. The authors in [69,112] and more recently in [11], have worked towards adapting to and accommodating online changes in the system's dynamics and order (e.g., when new sensors or actuators are plugged in a closed-loop system), for linear systems. A more recent approach to the control reconfiguration and fault detection in non-linear systems can be seen in [100].

The aforementioned work deals with the design of controllers with pre-defined structure, which however have the required adaptation capabilities to maintain satisfactory performance in the presence of certain system uncertainty and changes. Nevertheless, there are cases where the design of a single-structure controller is not practical or it cannot satisfy the control objectives. Such cases are addressed by another research area, the "variable structure control" [58]. This area assumes that the control system switches among several pre-defined control structures that take

over when certain criteria are met (e.g., when a regulated state crosses the bounds of certain pre-defined areas in the state-space). This type of systems are a special case of "hybrid dynamical systems" that have been studied extensively in the literature [74].

Beyond the dynamic changes that necessitate the adaptation and/or switching of control structures, a complementary challenge present in large-scale systems is the size of the control problem in combination with the complexity of the plants in terms of modelling, as well as the heterogeneity of control system modules that need to be employed. These challenges led the research community to exploit the "divide and conquer" principle, by breaking the large-scale systems into parts that can be modelled and controlled separately and by adopting modular architectures of control systems. An early effort towards introducing the need for a modular architecture for the control system design and the concept of structuring the data exchanged based on standardised descriptions of their characteristics, has been discussed in [14]. More recent efforts [57], though not focusing specifically on the control problem, address the modelling of cyber-physical systems as collections of services offered by specific physical devices and cyber tools within certain time and domain context. These services are then invoked to perform specific tasks. Moreover, the authors in [45] are taking advantage of the cloud computing paradigm to offer middleware solutions that allow abstraction of services of physical units for online invocation. A more recent work [18] focuses specifically in the Building Automation Systems (BAS) domain and creates a library of models capturing the dynamics and input-output relations of cyber-physical modules present in real BAS setups. The library contains algebraic and differential models, with discrete and/or continuous states, linear and non-linear, accounting for a large number of possible configurations between the underline modules. The objective is to facilitate BAS engineers to design specific solutions by choosing and configuring models from the subject library. To facilitate the automation of the process, the instantiation of the parameters used in the models can be performed online, by retrieving the information from existing building designs [23]. The authors in [27, 71], address the modular architecture problem by a software programming and operating system perspective. They develop a *Building Application Stack* and a *Building Operating System Services* architecture, which allows developing applications based on existing software system modules and subsequently developing control applications for buildings that are decoupled from hardware and building specificity. The work further automates the interoperability

of the system modules/applications, by implementing metadata models for their description and fuzzy queries for abstraction of their configuration.

A big challenge accompanying the modules (component-based) architectures discussed above was the interoperability of the modules and how to facilitate it and automate it. Responding to this challenge, the research community had investigated the use of pre-defined knowledge models that capture the characteristics of the components and the knowledge about them, in machine-readable format. These models are also called "semantic models" since they focus on defining the meaning and capabilities of components and not their implementations. The use of semantic and knowledge models in control systems had been proposed earlier, as can be seen in [102]. In this article, the semantic knowledge models are considered as an artificial intelligence tool, required to be used for the modelling of knowledge about plants, control system components and control goals. An architecture is then proposed, which allows the control system to reason upon the modelled knowledge and select appropriate control systems from a pre-existing bank of control systems. The idea behind this approach is very close to what we develop in our work, however, we were not able to locate information in the literature about continuation of those early efforts. The concept of expert knowledge modelling in control systems is also used in [72], mainly focusing on the validation of simulation software at the semantic level, i.e., in terms of the meaning and capabilities of software functions, rather than the facilitation of their online interoperability. A more system theoretic approach to presenting the concepts of semantic control systems can be found in [66], which discussed the notion of semantic rules as mappings between physical phenomena and explained that control is in general rule-following, be it functional or semantic rules. Furthermore, concrete examples of using semantic knowledge models in the smart buildings domain, are observed in the literature. For instance, the "DOGont" ontology [15] deals with the current issues of vendors' and technologies' heterogeneity in domotic environments. [Note: an ontology is a standard format (XML-like text) representing knowledge facts in the form of triplets ⟨ *subject - property - object* ⟩ [49]. More details can be found in Section 3.3.1]. Also, the authors in [87] describe an ontology-based expert system that is able to transparently control the home automation processes by learning from the human users' behaviour. A more recent and complete approach in the Building Automation Systems (BAS) domain is the one in [92], where the authors design the BASont ontology. This ontology

explicitly combines the current Building Information Models [23] with abstract and concrete knowledge from the building automation systems' devices and end-points. This work offers a good basis for building control applications that need to receive online information about the devices' location and capabilities. Combining the information with the reasoning capabilities of ontologies, makes it possible to support effective online configuration and decision making use cases. Knowledge modelling efforts have been undertaken also in the Computational Intelligence domain, where a significant milestone was the approval of the IEEE Standard for Fuzzy Markup Language (IEEE Std 1855-2016), which specifies an interoperability framework for fuzzy logic controllers [4, 61].

More recently, the emergence of the IoT paradigm and the growing ecosystem of IoT-enabled devices that exploit (wireless) Internet connectivity, have created new opportunities in-line with the modular architecture discussed above, as well as new challenges for large-scale cyber-physical systems and subsequently for their effective monitoring and control. The IoT vision is exactly for the capabilities of components (e.g., sensors, actuators, signal processing functions, decision making functions) to be consumed as "services" by cyber implementations of control algorithms thus offering more advance and composite services [107]. The term "composition" refers to utilising a variety of different services/things in a system that explores their individual functionalities, so that it can be observed from a higher level as offering one single (composite) service. This paradigm has emerged from the "Web services", a key technology of Web 2.0 that in late 1990's enabled web resources to exchange messages and offer enhanced interaction between cyber sources and humans [94]. The technologies behind these cyber services are still emerging [90], while in parallel the focus has been on the semantics of the content and artificial intelligence (Web 3.0) to enable knowledge structuring and meaning-aware interaction between Web services. The IoT has already become part of this technological evolution [25, 53], bringing the physical world into play, with smart devices connected to each other and speaking to each other via machine-to-machine communication [83] to perform composite tasks without human intervention. Innovation in this area is currently happening since knowledge and technology are now becoming mature enough to facilitate full automation of control component-based architectures with online configuration capabilities.

The aforementioned vision, in order to be successfully pursued, requires knowl-

edge from a broad range of ICT-related fields, as well as from the systems theory and automatic control domains. The W^3C Semantic Sensor Network Incubator Group (SSN-XG), as well as the Open Geospatial Consortium (OGC) have been among the first who identified the challenge. They created the "Semantic Sensor Network (SSN)" [55] and the "SensorML" standard [109] respectively, for the semantic characterisation of sensors' operation. The standards have recently published updates that take into consideration also the modelling of actuation and other processing operations. During the last few years, several groups have started working on top of the above standards, to address the interoperability, evolve-ability and online reconfiguration challenges of IoT-enabled systems. The authors in [73] have worked on the conceptual matching of heterogeneous lexical sources and ontologies in the IoT and Smart Cities domains. The authors in [42] have used rule-based systems for the automatic construction of topical ontologies from real-world IoT sensor measurements. They also developed a distributed mechanism to index IoT resources and their data, based on their location, thus enabling online grouping by similarity [36]. Another group used knowledge models to process large amount of IoT data and produce added-value knowledge for Smart City applications [118]. Other works [25, 53] model IoT components as Web resources, thus enabling their online discovery and utilisation of their capabilities. An attempt to design self-configurable systems composed of IoT components has been also presented in [21]. There is also work from the "smart factory" domain and the Industry 4.0 concept, with teams working on modelling the information about factory objects (hardware or software) using semantic knowledge formalisms [50] and also modelling the IoT resources and processes towards optimal resource management [113]. Other researchers have also acknowledged the need for additional flexibility in the IoT paradigm and started exploiting the Building Information Model [89, 95]. Another such example is the work in [92], which has been discussed earlier and which resulted in an ontological model able to facilitate advance online decision making use cases. The same work has been used and extended with additional Resource Description Framework (RDF)-based information concepts to facilitate the fault diagnosis use cases in the Building Automation domain [37]. More specifically, this work uses ontological models of the buildings domain, so as to enable the encoding of building information in machine-readable format and subsequently the online passing of parameters' values required by a certain fault diagnosis method. As a result, it enables the online synthesis of the

fault diagnosis scheme, where building parameters are not known apriori. The idea and the technologies involved in this work are very relevant to our work on online synthesis of distributed feedback control schemes, discussed in Chapter 4.

The above presented work addresses spherically and extensively the services' and data semantic interoperability challenge in the IoT domain. It mainly focuses on the sensing and data collection and manipulation capabilities, which support IoT applications in general. **In an IoT perspective, the key challenge still remains the implementation of the feedback control systems as compositions of components acting as providers and/or consumers of "services" thus allowing flexibility in adapting to components' changes**. IoT-enabled applications will not be able to offer effective monitoring and control services, if the semantic models do not explicitly consider the systems' modelling and control design details. The need to extend the semantic models towards conceptualising control and other processing entities as well, has been partially addressed in [70]. The author identified the need to model sensors, actuators and controllers, presenting a "Semantic Smart Gateway Framework" that acts as an interoperability service and mediator between IoT application providers and consumers. Although this work effectively models the semantics of control from an input-output perspective, it does not take a control-theoretic view, thus it is not opening the way to deal with the internal dynamics and online (context-aware) re-configuration of IoT-enabled control systems. The incorporation of feedback control capability, requires on one hand the ICT experts with their understanding of the concepts of services' online composition and on the other hand the control experts with their understanding of systems' dynamics and control.

The work related to this Dissertation is within the above framework. We advance the state of art by combining existing tools from the two different aforementioned areas and addressing certain IoT-related challenges. We address in general a variable structure control problem, since we assume that there is no pre-designed single controller that poses all required adaptation capabilities to learn and accommodate the changes in the controlled system and/or the control system components (e.g., in case of replacing a faulty sensor). We consider a plug-and-play capability in our proposed solution, in terms of having a mechanism that enables the control system to switch online between different configurations when components are plugged-in. The switching among extracted control system configurations is decided and enforced through a discrete-state logic-based deductive reasoning process [9]. The

explicit design of this mechanism is a key contribution. This mechanism assumes that the available control modes (configurations) are not defined in advance; they are configured online using components from an evolving library of control system components as discussed in aforementioned literature review, able to interoperate in a modular architecture but without hard-coded interfaces. At discrete events, the logic-based system exploits modelled knowledge and subsequent (standards' based) semantic characterisations of components' interfaces, as well as deductive inference rules to produce the switching signal. The knowledge is stored in "knowledge graphs" formed by meaningful relations between modelled knowledge entities of pre-defined types [39,104]. These are further clarified in the context of the Dissertation.

Our work moves one step further, by exploiting IoT-enabled re-configuration capabilities for the comfort- and cost- effective online evolution of feedback control schemes in smart building automation systmes. Currently, the building automation industry is dominated by big players [119] who provide solutions for HVAC, lighting, security and other applications, as well as integration capabilities explored by local integrating companies through network equipment and communication protocols. However, the flexibility of the solutions remains within the bounds of a pre-defined set of components that are compatible with the overall system by design. The bounds become narrower when it comes to feedback control options, since the need for portability of the solutions leads to using basic model-free control structures (e.g., on-off control, tunable PID control, IF-THEN-ELSE control) [40]. In this work we use model-based feedback linearization controllers [35] and we show their portability through the online passing of building topology and parameters. We also apply the work from the cost-effective load demand generation in electric power systems [46], to perform cost-effective heating control in buildings using multiple heating devices deployed online.

To the best of our knowledge these challenges have not yet been addressed. Our work aims to turn advanced intelligent feedback control and decision making methods accessible to emerging IoT-enabled building automation systems.

Chapter 3

SEMIoTICS Architecture and System

3.1 Problem Formulation

Large-scale complex systems are typically equipped with a large number of components and subsystems and they evolve over time as new technology becomes available, leading to the addition of new or replacement of existing components. For example, in the context of a smart building, new IoT devices (e.g., occupancy sensors, CO and CO_2 sensors, heaters, humidifiers) may be plugged-in and become available. On the other hand, feedback control systems are typically designed for certain plants, taking into account certain measurable variables, known dynamics, certain actuation capabilities, as well as other pre-assumed relevant information. With the current technological advancements, it becomes desirable for control systems to be equipped with self-reconfiguration mechanisms, so as to incorporate any new components online, thus ensuring continuous operation and satisfaction of control specifications (e.g., user safety, comfort, energy efficiency).

Consider the plant in Fig. 3.1, characterised by a vector of state variables $x \in \mathcal{R}^{n_x}$, with n_x being the dimension of the state vector. In a Smart Building context, these can represent room temperature, humidity, lighting, air quality, etc.. We assume that the state variables need to be controlled to satisfy certain control objectives. The satisfaction of the objectives is achieved through the design of a control system able to generate an appropriate plant input vector $v \in \mathcal{R}^{n_v}$ of dimension n_v and provide the expected control service to the plant. For generality, the vector $w \in \mathcal{R}^{n_w}$ represents n_w uncontrolled inputs to the plant. In an IoT context, the plant may be equipped with multiple components from the types discussed in Section 2.2; sensors able to

measure its states, actuators able to act on and affect its states, controllers able to consume the measurements and drive the actuators, other functions able to process and transform the produced signals when required, etc.. Instances of sensors may represent thermistors, occupancy sensors, limit switches, smoke detectors and so on. Instances of actuators may be the heating elements used to adjust the temperature, a lamp to adjust the lighting, a motor to open a window, and so on.

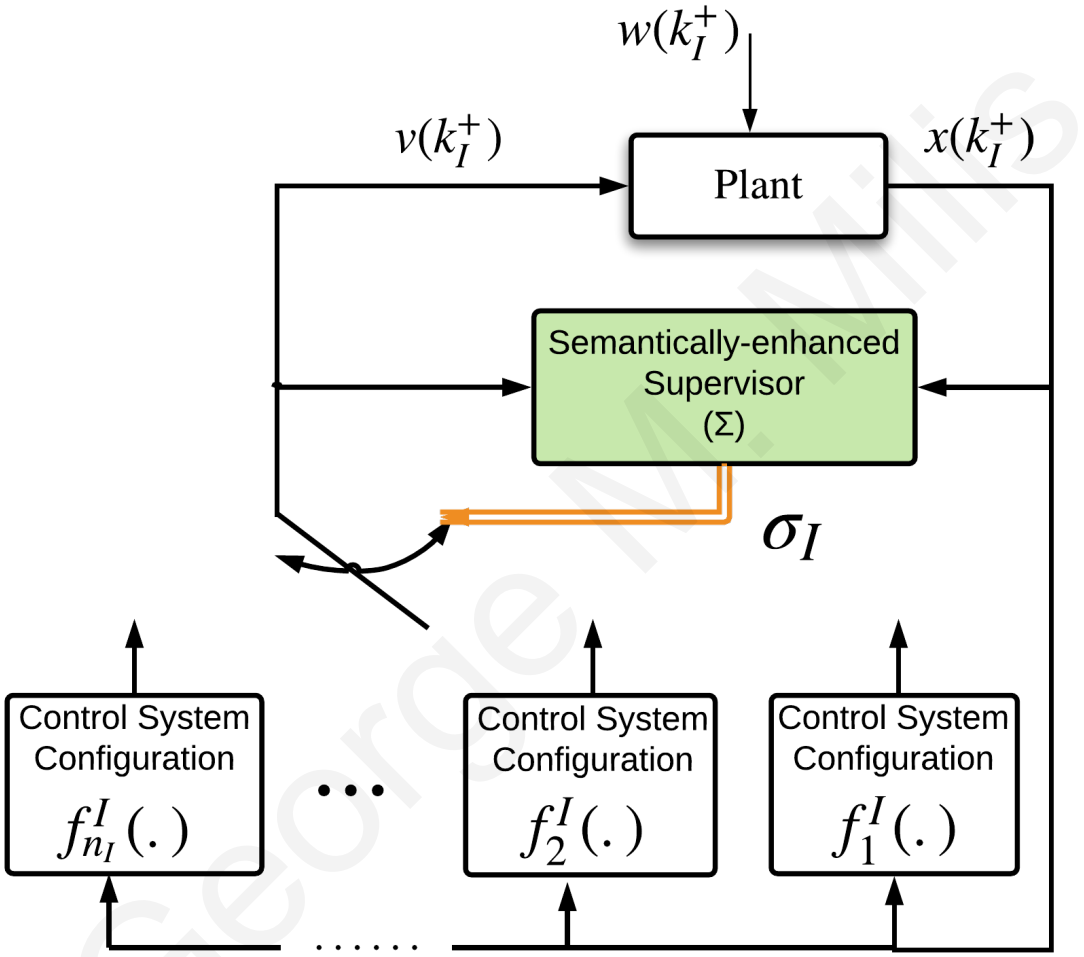


Figure 3.1: A plant with multiple IoT components available at time step k_I, forming n_I possible "Control System Configurations". A supervisor generates the respective selection signal σ_I

As mentioned also in Section 2.2, we consider a discrete-time formulation, where $\mathcal{K}$ is the set of discrete time samples of the continuous signals and $k \in \mathcal{K}$ is an index of the discrete samples set. We assume that at certain times $k_I \in \mathcal{K}$, $I = 0, 1, 2, ...$ events happen that change the availability of IoT components in the plant. With k_I^+ we denote the index of the discrete time steps for $k_I \leq k < k_{I+1}$ (the time between

two consecutive events where available components change). Within each such period without changes, we assume a finite set $\mathcal{F}_I$ of available components, which correspond to a set of valid control system configurations $C_I = \{f_J^I | J = 1, 2, ..., n_I\}$, able to offer the required control service to the plant. A control system configuration in the time window $k_I \leq k < k_{I+1}$ is formulated as in (3.1),

$$v(k_I^+) = f_J^I(x(k_I^+), k_I^+; \sigma_I) \tag{3.1}$$

where σ_I represents the configuration selection decision, which activates, or not, a configuration option. Depending on the process and the given specifications, an engineer (human) would have selected and designed a feedback control system using specific instances of the components in $\mathcal{F}_I$, as well as specific domain knowledge expertise. In other words, to make a decision, the engineer relies on thinking and reasoning which considers the available knowledge about the domain and the feedback control engineering, including the associated semantics of each control system component.

We address the challenge of designing a reference architecture and a semantically-enhanced Supervisor system Σ (light-green-highlighted module), with the ability to utilize pre-modelled knowledge and a set of feedback-control specifications, implement inference mechanisms and (re-)configure a suitable feedback-control system online, for a large class of plants. The decision about the configuration of the control system, can be formulated as in (3.2).

$$\sigma_I = f_\sigma(\mathcal{G}_I, \mathcal{S}_I, \mathcal{F}_I) \tag{3.2}$$

where σ_I is the decision signal that models the selection of one specific control system configuration from the set C_I, denoted as $f_{J^*}^I$; $f_\sigma(\cdot)$ is a function implementing the reasoning and the decision about the configuration; $\mathcal{G}_I$ is the state of the declarative-language-based "Knowledge Graph" that hosts the available experts' knowledge at time k_I; $\mathcal{S}_I$ is the state of the set of feedback-control (semantic) specifications given to the function at time k_I (e.g., the characteristics of the desired state, the weights to pre-defined criteria, etc.); $\mathcal{F}_I$ is the set of all available control system components at time k_I, presented earlier.

The challenges discussed above, are addressed through the design and development of the "Semantically-Enhanced IoT-enabled Intelligent Control Systems (SEMI-

oTICS)" reference architecture and its core processes. SEMIoTICS processes are built around a semantically-enhanced supervisor Σ. The next sections, zoom into the design of Σ and provide details about the implementation of the function f_σ and the subsequent enforcement of the configuration. It is noted that the implementation of the function f_σ is two-fold. It first involves a "Semantic Reasoning" process that relies on logical queries in the "Knowledge Graph" $\mathcal{G}$, detailed in Section 3.3.4. Second, it involves a "Configuration Selection" process which minimizes an objective function that considers pre-defined quality-of-service (QoS) criteria as detailed in Section 3.3.7.

3.2 Reference Architecture

SEMIoTICS and the respective reference architecture are presented in Fig. 3.2. We consider each selected "Control System Configuration" $f^I(\cdot)$ (the pointer J^* is omitted for notation clarity) as a composition of instances of specific types of available components as a result of a change at time k_I. Each composition consists of zero or more sensors producing the signal-vector $y(k_I^+)$, one or more controllers producing the control decision signal $u(k_I^+)$, one or more actuators producing the plant input vector $v(k_I^+)$ and zero or more of other signal processing functions. The latter are further classified into: i) "Pre-Control Functions" that transform plant state measurements to the signal-vector $y'(k_I^+)$; ii) "Post-Control Functions" that transform the control decision signal to the signal-vector $u'(k_I^+)$; and iii) "Parameter Functions" that utilize measurements of plant properties and produce/update values of general plant and control system parameters stored in a "Parameters Registry" set $\mathcal{Z}$. The parameter vectors required by the respective types of participating components contain elements from this set, i.e., $\zeta^s, \zeta^y, \zeta^c, \zeta^u, \zeta^a, \zeta^\zeta \in \mathcal{Z}$. As discussed in Section 2.2, designs of feedback-control systems may require more types of components, e.g., "State Estimators" and "Online Learning" structures; however, we do not consider these types of components in the current version of SEMIoTICS.

The inputs and outputs of the components comprise the signals involved in the feedback loop continuous information flow (black continuous lines). On the other hand, the values of parameters (e.g., openings of windows in a building, set-points of heating elements, reference plant state trajectories, etc.) are considered retrieved on-demand and/or on-availability, from dedicated "Parameter Functions"

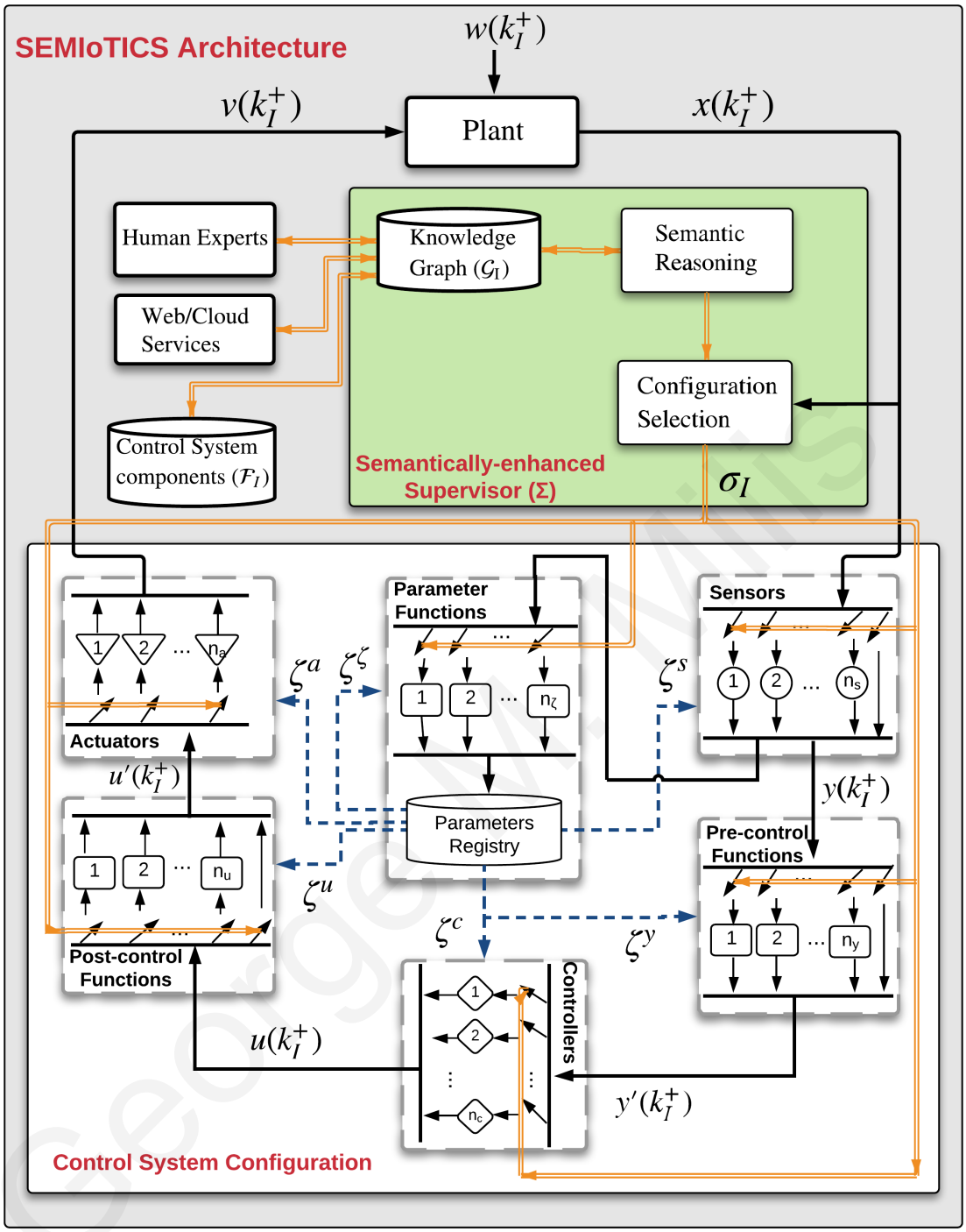


Figure 3.2: The SEMIoTICS architecture

(blue dashed lines). The components are considered implementing internally certain functionality relevant to their type. For instance, a "Post-Control Function" may implement an amplifier of a control signal so as to properly drive an available actuator; a "Pre-Control Function" may act on the measured signals to perform data validation/reconstruction or state-estimation, e.g., Kalman filter, Luenberger observer, etc.; a controller may be a simple On/Off function or a PID implementation or a more

advanced non-linear and adaptive control algorithm. SEMIoTICS considers all components as syntactically modelled by the general formulas in Section 2.2. Whether a specific component will take part in a control system configuration $f^I(\cdot)$, is defined by the Semantically-enhanced Supervisor module Σ through the selection decision σ_I (double orange line), as detailed in the sequel.

3.3 Semantically-enhanced Supervisor

In this section we go into the design of the sub-modules and operations of the supervisor Σ. All operations of Σ are built on top of the "Knowledge Graph" $\mathcal{G}$, which is a super-graph of many (bipartite) graphs formed by semantic relations between modelled knowledge objects of pre-defined types (see Section 3.3.1 in the sequel). To model the knowledge we use declarative language, i.e. *ontologies*, building on top of: i) the W^3C OWL-S standardisation effort that deals with the semantic composition of (Web) services/components [75]; and ii) the Semantic Sensor Network (SSN) ontology which is a joint effort of the W^3C and the Open Geospatial Consortium (OGC) and describes sensors and actuators, as well as the features-of-interest and their specific properties, which they observe/affect [55].

Referring back to Fig. 3.2, it can be seen that the modelled knowledge can be updated/inserted online by human experts. For instance, an engineer may provide information about a new sensor, the topology of the plant, and so on. In addition, updates of the knowledge can be performed automatically through Web/Cloud services, e.g., retrieving information about a component from online sources, assuming that such information is kept in a remote database in the form of "semantic drivers" (see the Building Information Model [23] in the smart buildings domain). Finally, the "Knowledge Graph" also retrieves information directly from the available IoT control system components, so as to become aware of their characteristics and capabilities, location, etc.. The communication requirements are technically addressed by adopting established message formatting standards and communication protocols, like MQTT [83]. The operation of modelling the knowledge about the characteristics and capabilities of control system components is called "semantic annotation" and is presented in Section 3.3.2 in the sequel.

3.3.1 The knowledge Graph

This section uses an illustrative scenario aiming at clarifying the way the "Knowledge Graph" is built and how the semantic (deductive) inference is performed over the stored knowledge, to implement the decision mechanism for the switching and subsequent re-configuration of the feedback control system.

Consider the scenario of an office as in Fig. 3.3, where a digital sensor device f_1^s is deployed, measuring the temperature of the office x_1 in degrees Celsius. The office is also equipped with a heating device f_1^a which introduces heat to the office at some energy rate in kW. In addition, a controller f_1^c is deployed to regulate the temperature of the office by controlling the operation of the heating device (actuator). There is also a door on the south wall of the office, through which energy losses E_{loss} can potentially appear, however, it is initially considered fully closed and made of material with zero heat transfer coefficient.

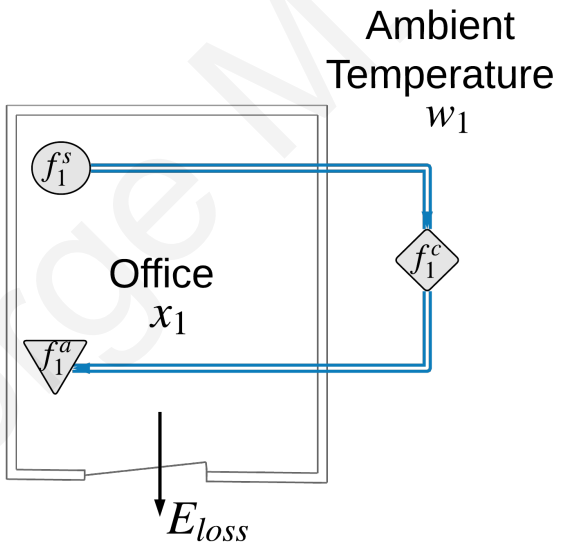


Figure 3.3: An open-plan office with inner temperature x_1, ambient temperature w_1, a deployed temperature sensor f_1^s, one heating actuator f_1^a and a temperature controller f_1^c

Considering the office with the dynamics of its temperature state, we identify the terms that a human would use in order to describe the plant. As shown also in Fig. 3.4, these are the features "office" and "ambient" that are of interest to the specific description, their physical properties "temperature" and "energy", the measurement units "Celsius (^oC)" and "kW", as well as the temperature sensor f_1^s, the

heating actuator f_1^a and the controller f_1^c. These linguistic terms describe "things" which comprise either cyber-physical entities or other types of entities and real-world phenomena and concepts. From a knowledge (logic) perspective, any plant can be considered as a set of "things" (knowledge objects), defined as follows:

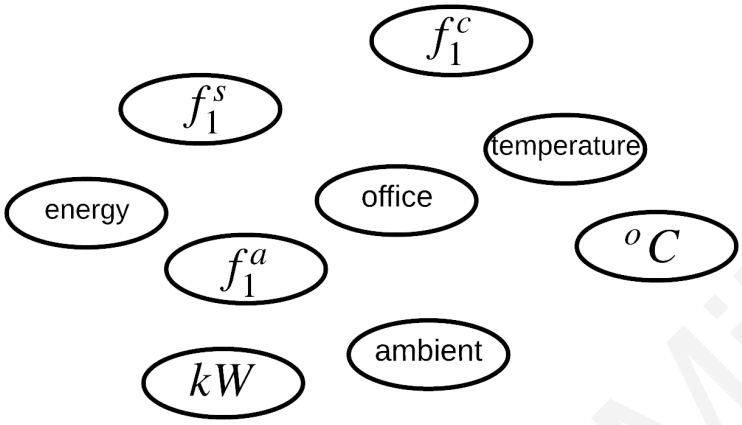


Figure 3.4: A set of terms that a human would use in order to describe the plant in Fig. 3.3

Definition 1. *Things*: $O = \{o_i | i = 1, \cdots, n_O\}$ is the finite set of all "things" which an expert would use as a convention, to describe what exists in a subject plant. (The terms "individuals" and "instances" are also used to refer to these "things", in the ontology engineering domain.)

The "things" are not all of the same type. The type of each "thing" is an important part of the knowledge which experts assume in order to share the same meaning (refer also to the Theory of Types by Russel, Section 2.1). For instance, a non-expert would not be able to understand the description without knowing that the linguistic term "Celsius" refers to a unit of measurement, while the linguistic term "office" denotes a location feature of the plant. This is further formalized through a convention about the types of "things", which is defined, for the purposes of our work, as follows:

Definition 2. *Classes*: $\Omega = \{\omega_i | i = 1, \cdots, n_\Omega; \omega_i \subseteq O\}$ is the set of all types/classes to which the "things" belong. Note that each element of Ω is a subset of the set O.

According to the current example, elements of Ω are the set of sensors ($\mathcal{F}^s \in \mathcal{F}$), the set of actuators ($\mathcal{F}^a \in \mathcal{F}$), the set of controllers ($\mathcal{F}^c \in \mathcal{F}$), the set of plant features of

interest ($\mathcal{L}$), the set of physical properties ($\mathcal{Q}$), as well as the set of measurement units ($\mathcal{M}$). In addition, it can be seen from the analysis in Section 3.1 that all components are essentially functional mappings between certain inputs to certain outputs. The inputs to a function describing the operation of component, are distinguished to inputs that correspond to core feedback control signals (e.g., sensor measurements, control decisions, actions to plant, etc.) defined as "inputs" and to inputs that represent other "parameters" required by the components to perform their duty. Therefore, three additional classes of "things" are defined here as $\mathcal{U}$ for the "inputs", $\mathcal{Y}$ for the outputs and $\mathcal{Z}$ for the "parameters". The subset of the inputs of a specific sensor f_1^s is then defined as $\mathcal{U}^{(f_1^s)}$. For completeness of the description we add also the set of components' capabilities ($\mathcal{P}$), which helps to describe the role of an input, output or parameter signal. Fig. 3.5 presents the grouping of all aforementioned "things" in the pre-defined set of classes. It is noted that in general, each "thing" may belong to more than one classes, however, for simplicity we consider in our work that each one belongs to only one class.

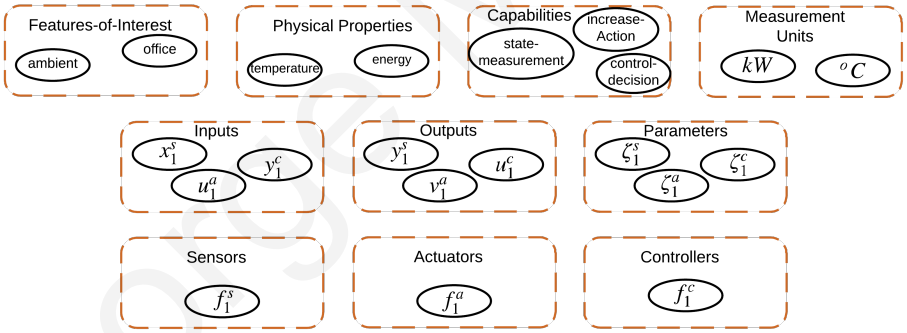


Figure 3.5: The types/classes to which the "things" belong

Further to the classes of "things", the full understanding of the meaning is facilitated by the (semantic) relations between them. That is, "Celsius" ($m_1 \in \mathcal{M}$) is a measurement unit of "temperature" ($q_1 \in \mathcal{Q}$), while $f_1^s \in \mathcal{F}^s$ is a "sensor" that has output $y_1^s \in \mathcal{Y}$ which represents the physical property "temperature". Such logical relations, comprise a third convention assumed by humans when sharing knowledge. In general, relations comprise mappings between "things" of one class/type to "things" of another and their definition is given below:

Definition 3. *Relation-graph*: $G(\mathcal{V}^o, \mathcal{V}^d, E_n^{(\mathcal{V}^o,\mathcal{V}^d)})$ defines a non-balanced graph with vertices being the elements of the sets $\mathcal{V}^o, \mathcal{V}^d \in \Omega$ and edges being the elements of the

set $E_n^{(\mathcal{V}^o,\mathcal{V}^d)} = \{(v_i^o, v_j^d) | v_i^o \in \mathcal{V}^o, v_j^d \in \mathcal{V}^d, i \in \{1, ..., n_{\mathcal{V}^o}\}, j \in \{1, ..., n_{\mathcal{V}^d}\}\}$, which represent the arcs connecting elements of the origin set $\mathcal{V}^o$ to elements of the destination set $\mathcal{V}^d$. (The term "properties" and "relations" are used in the ontology engineering domain).

Typically, a "Relation-graph" is bipartite, i.e., the sets $\mathcal{V}^o$ and $\mathcal{V}^d$ comprise distinct classes of "things", as shown with the coloured continuous-line arrows in Fig 3.6. However, in the case when the "Relation-graph" describes mapping between "things" of the same class (e.g., two distinct features-of-interest), then the origin and destination sets of vertices converge to one. Furthermore, in the case where multiple relations are required and defined between the same pair of classes, a subscript $n = 1, 2, \cdots$ is used with the set of edges E in the above definitions, in order to differentiate between the relations. For example, a building consists of several rooms (features of interest). Within each room, one or more other features of interest may be identified. Then, one relation may represent that a feature-of-interest "is part of" another feature-of-interest (or "contains" for the inverse relation), as show with the purple line in Fig. 3.6. This relation is represented by $G(\mathcal{L}, \mathcal{L}, E_1^{(\mathcal{L},\mathcal{L})})$, using the notation of "Definition 3". Another relation is that a feature-of-interest "is adjacent to" some other feature-of-interest which is also bi-directional and is represented by $G(\mathcal{L}, \mathcal{L}, E_2^{(\mathcal{L},\mathcal{L})})$. See for instance the relation shown with the pink arrow in Fig. 3.6, where it is assumed that a second office exist and that it is adjacent to the first one. Other relations between features-of-interest may also be defined as needed. In the case where the features-of-interest correspond to spatial (relative or absolute) locations, relevant relations comprise a very important property of cyber-physical components, as the operation of the latter is often highly associated with their spatial properties.

Using the notation introduced with "Definition 3", Fig. 3.7 shows another view of the graph with the sub-graphs ("Relation-graphs"), which represent various relations between "things" in the subject plant. As before, the nodes of the graphs represent "things" while their classes are shown with dashed-line rectangle containers. An edge that connects two "things" represents the relation that exists between them. The visualised edges represent ten different "Relation-graphs", however, for readability purposes, only five of them are highlighted with different colours.

Multi-edge paths between "things" represent "composite" relations, i.e., relations

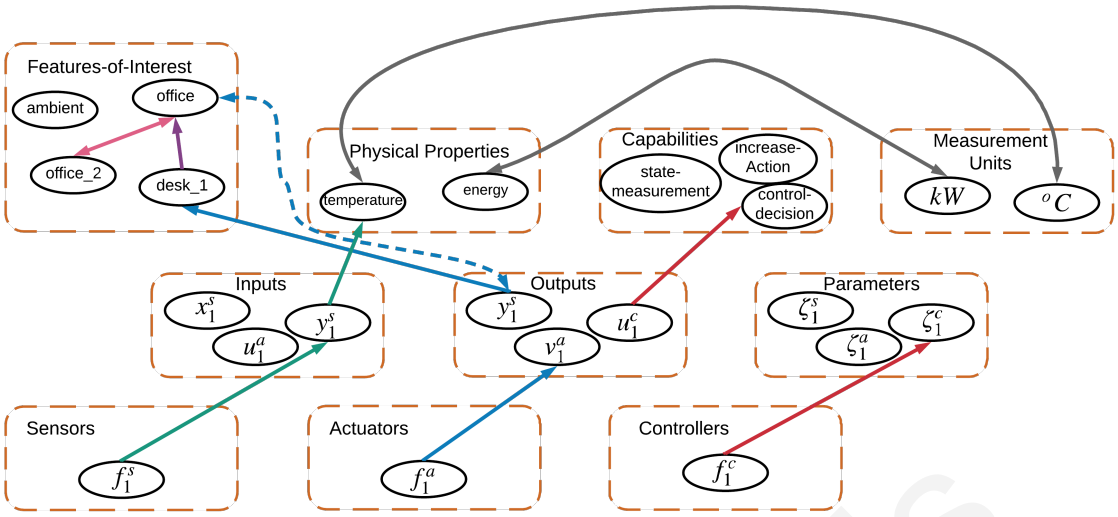


Figure 3.6: Relations between "things" of the same or of different classes. Coloured, continuous-line arrows illustrate explicit relations between things of certain types, whereas dashed-line arrows illustrate indirect relations between things of certain types

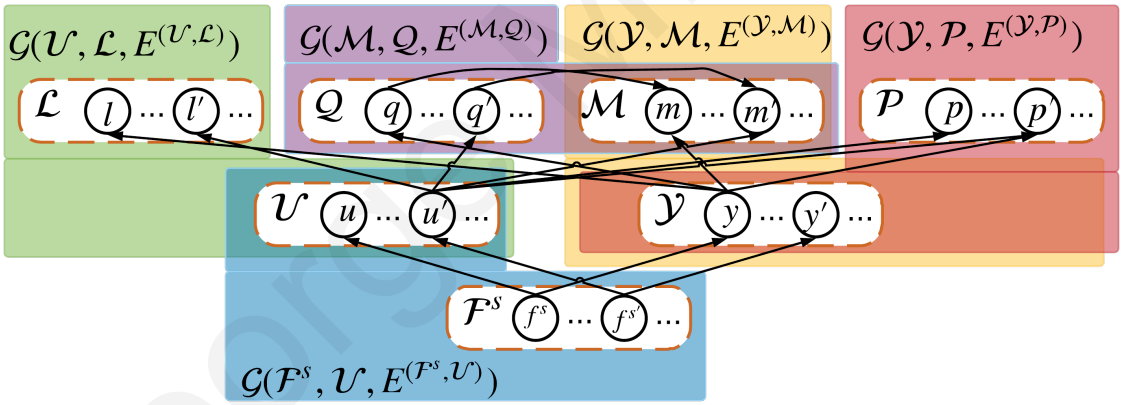


Figure 3.7: Representation of relation graphs: The graph $G(\mathcal{F}^s,\mathcal{U},E^{(\mathcal{F}^s,\mathcal{U})})$ is highlighted with light blue colour, the graph $G(\mathcal{U},\mathcal{L},E^{(\mathcal{U},\mathcal{L})})$ is highlighted with light green, the graph $G(\mathcal{Y},\mathcal{M},E^{(\mathcal{Y},\mathcal{M})})$ is highlighted with light orange, the graph $G(\mathcal{Y},\mathcal{P},E^{(\mathcal{Y},\mathcal{P})})$ is highlighted with light red, and the graph $G(\mathcal{M},\mathcal{Q},E^{(\mathcal{M},\mathcal{Q})})$ is highlighted with light purple. The "prime" superscript is used to illustrate the potential of having multiple "things" in each class.

not explicitly defined in advance. For example, assuming that sensor f_1^s is located on the feature of interest "desk 1" and that the latter is located within the office, as shown with the respective arrows in Fig. 3.6, we can deduce that the output of the sensor, y_1^s actually represents the temperature of the office, further to the temperature on desk

1. This is captured with the blue dashed line in the same figure, which represents the respective composite relation. The same concept is illustrated in Fig. 3.7, where the bipartite graph $G(\mathcal{Y}, \mathcal{Q}, E^{(\mathcal{Y},\mathcal{Q})})$ can be considered as formed by vertices from the sets $\mathcal{Y}$ and $\mathcal{Q}$ and edges the paths of length 2 from nodes in $\mathcal{Y}$ to nodes in $\mathcal{Q}$ (e.g., the path (y, q, m). This is a relation that is derived from the composition of two other relations, using an inference rule (see Section 2.1).

Since the relations are defined from specific origin class to specific destination class, the respective graphs are directed in general. However, in most cases the converse relation also exists, which results in bidirectional (or non-directional) graphs. For instance, considering the illustrative example introduced earlier, saying that sensor f_1^s is "located in" l_1 : office, would have the same meaning as saying l_1 "contains" sensor f_1^s. Wherever the converse of relations utilized in our work are meaningful, the edges are shown as non-directional or bi-directional.

Summarizing, the Knowledge Graph $\mathcal{G}$ is a super-graph of many "relation graphs" (sub-graphs). The classes and the nodes ("things") in these sub-graphs define the size of the overall Knowledge Graph $\mathcal{G}$, while the number and types of relations that are defined between different classes of "things", represent the connectivity of $\mathcal{G}$. The following section describes how the "Knowledge Graph" is used in practice for the semantic annotation of the feedback control system components and the subsequent implementation of the semantic reasoning (deductive inference) process.

3.3.2 Semantic Annotation

The SEMIoTICS Knowledge Graph, $\mathcal{G}$, provides explicit support for all considered types of control system components: Plant, Sensors, Actuators, Controllers, Processing Functions (Pre-Control, Post-Control and Parameter Functions). For its design we adopt certain parts of the OWL-S "Service Profile" model [75], which facilitate the modelling of the service offered by each type of component. I.e., each component has inputs, outputs, parameters, as well as some additional information for its categorisation. For simplicity, at this stage we do not use the OWL-S concepts of "pre-conditions" (certain state conditions which need to hold for a service to be invocable) and "effects" (certain state values, beyond outputs, that are the result of the execution of a service).

The semantic characterisation of the control system components is mainly based on the SSN ontology [55]. The SSN ontology defines sensors and actuators as "Systems" that "observe"/"act-on" a certain "property" of a "feature-of-interest" of the environment/plant in which they are deployed. For instance, a sensor may measure the property "temperature" of the feature-of-interest "room 1" in a given building. The same ontology defines that such a "System", in order to provide its intended service, implements a "Procedure" that has certain "Inputs" and "Outputs". In our work we extend this by modelling sensors, actuators, controllers, other processing functions and the plant itself as components that transform properties of the plant's features-of-interest, towards offering a collaborative control service. To avoid complexity and without loss of generality, we do not use the SSN ontology in its full detail but we adopt the parts that help us define our semantic models. Moreover, we adopt the concept "feature-of-interest" to refer to specific objects in a physical plant, which correspond to specific "locations" in the plant. "Locations" here do not refer to a representation of coordinates in a geographic map; they refer to parts of the plant and objects in the plant that correspond to certain relative positions; e.g., "heater 1", "room 1" "window 1" are locations and subsequently "features-of-interest" in the plant. In order to model relations between locations, we adopt concepts of the GeoSPARQL model [43], e.g. "touches", "inside", "contains".

We model explicitly the services offered by control system components and facilitate their online invocation, by combining the "Procedure" concept of *SSN* with the "Service" concept of *OWL-S*. That is, beyond inputs and outputs we consider additional "parameters" required by each component. Also, we incorporate categorisation of the services in terms of their "capability" on the specific properties of "features-of-interest" that are represented by inputs, outputs and parameters. For the convenience of the reader, we may refer to the inputs, outputs and parameters associated with a component/service, collectively as "end-points" of that component/service. The adopted way of annotating/describing the components, allows us to model the knowledge about all produced/consumed signals using the "Five Ws and one H" method [52], which has been proposed for capturing and communicating the correct information about an entity in a reporting or decision making context.

As an example, Fig. 3.8 shows the semantic annotation of an input, an output and a parameter of a component. It can be seen that the semantic annotation space Λ is defined by four dimensions: $\Lambda \equiv \mathcal{L} \times \mathcal{Q} \times \mathcal{P} \times \mathcal{M}$, as also introduced earlier in

Section 3.3.1. That is, an element of the space Λ is represented by the specific values in a quadruple of respective variables:

- Variable l represents the plant's "feature-of-interest" and answers to the question "WHERE", taking values from the set $\mathcal{L}$ = {office, zone 1, zone 2, door, window, ambient, wall 1, ceiling 1, heater 1, ...}. The set can be the output of the building design using a CAD software. e.g. an extract of a BIM [23].

- Variable q represents the studied property of the feature-of-interest and answers to the question "WHAT", taking values from the set $\mathcal{Q}$ = {temperature, energy, opening, flow rate, filtration rate, fan speed, time, ...}. The values of this set, as well as of the measurement unit below, can be retrieved from existing models (e.g., the current version or future extensions of the Building Information Model [23]).

- Variable p represents the role of the signal/variable in the control system configuration and answers to the question "WHY", taking values from the set $\mathcal{P}$ = {state, stateMeasurement, controlDecision, disturb, referenceValue, plantTopology, regulate, increase, decrease, ...}. These values are given at the time of annotating the component, either manually selected by the engineer/technician or automatically by downloading the information from the Internet.

- Variable m represents the measurement unit of the property, where applicable, and answers to the question "HOW", taking values from the set $\mathcal{M}$ = {Celsius, Fahrenheit, kWatt, kilogramsPerSecond, percentage, ...}

Note that the question "WHO" is explicitly answered through the link of end-points to specific components, whereas the question "WHEN" is out of the scope of the decision making performed by the supervisor Σ. The size of the above sets can change online, adding or removing elements, without affecting the operation of the system.

The "Semantic Annotation" operation is defined as in 3.3 below:

$$\lambda(\cdot) : \mathcal{A} \mapsto \Lambda \tag{3.3}$$

where $\mathcal{A} \supset \{\mathcal{U}, \mathcal{Y}, \mathcal{Z}\}$ denotes the set of all end-points of components. For instance, the annotated input in Fig. 3.8 may represent the measured temperature state of

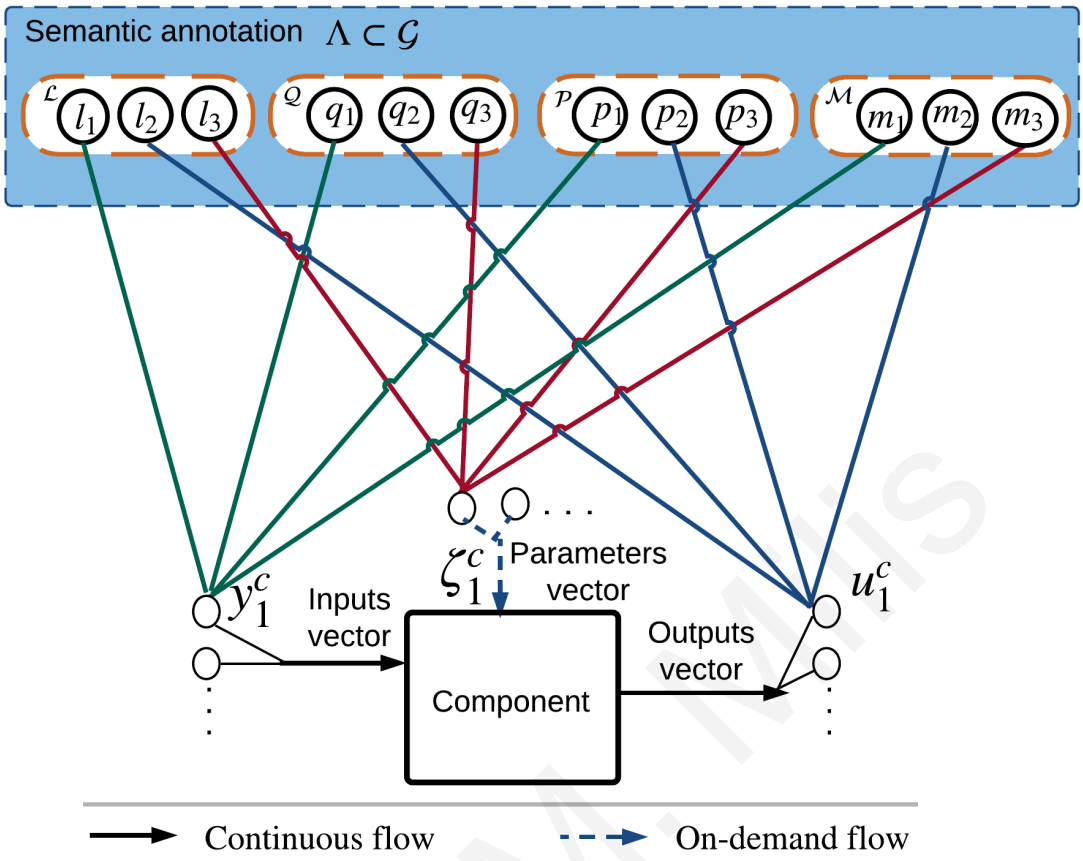


Figure 3.8: A control system component with an example semantic model of an input, an output and a parameter

the "office" in degrees Celsius and denotes a point in the space Λ, as: $\lambda(y_1^c) = \{l : \text{office}, q : \text{temperature}, p : \text{stateMeasurement}, m : \text{Celsius}\}$. In the same way, the semantic annotations of the example output and parameter are: $\lambda(u_1^c) = \{l : \text{heater}, q : \text{flow rate}, p : \text{controlDecision}, m : \text{kilogramsPerSecond}\}$ and $\lambda(\zeta_1^c) = \{l : \text{door}, q : \text{opening}, p : \text{plantTopology}, m : \text{percentage}\}$.

There are cases where a semantic annotation does not define a specific value for one or more of the annotation space variables. In such cases, the annotation is considered as covering an area of the semantic annotation space instead of a single point. In these cases, instead of a value we use the symbol of the respective set. For instance, if no specific measurement unit was defined for the above door opening, the semantic annotation would be: $\lambda(\zeta_1^c) = \{l : \text{door}, q : \text{opening}, p : \text{topologyParameter}, m : \mathcal{M}\}$.

We clarify that a component with multiple end-points is associated with a vector of semantic annotations, elements of Λ. For instance, in the case of a PID controller

with its coefficients exposed as a parameter vector in SEMIoTICS, these will correspond to a (equal size) vector of semantic annotations of parameters. We further clarify that the modelled control system components do not necessarily coincide with single physical devices or software tools; i.e., a single device may implement more than one of these components, each one with its own syntactic profile and semantic annotations. For instance, a device that measures the occupancy and uses that value to estimate the CO_2 concentration in a room, offering both values as output, will be modelled as two components: i) a "Sensor" receiving a "state" signal (e.g., occupancy of room) and producing a signal of capability "stateMeasurement"; ii) A "Pre-control Function" that uses a "stateMeasurement" signal and produces a processed "stateMeasurement" signal. This helps towards breaking the dynamics of a control system into smaller components which cooperate to provide the overall control service.

3.3.3 Semantic annotation models of components

The semantic annotation operation is clarified in the sequel through the illustrative example of Fig. 3.3. For convenience, we repeat here that the scenario involves an office with its temperature state and three deployed components: the sensor f_1^s, the actuator f_1^a and the controller f_1^c. The respective plant dynamics are represented by f_1^p. The sensor has an inputs-set $\mathcal{U}^{(f_1^s)}$ and an outputs-set $\mathcal{Y}^{(f_1^s)}$. The respective sets for the other components are defined in the same way. In addition, the following domain knowledge is assumed stored in the "Knowledge Graph":

- $\mathcal{L} = \{l_1 : \text{office}, l_2 : \text{ambient}, l_3 : \text{door}\}$
- $\mathcal{Q} = \{q_1 : \text{temperature}, q_2 : \text{heat-energy}, q_3 : \text{U-value}, q_4 : \text{valveOpening}\}$
- $\mathcal{P} = \{p_1 : \text{increase}, p_2 : \text{state}, p_3 : \text{plantTopology}, p_4 : \text{stateMeasurement}, p_5 : \text{controlDecision}\}$
- $\mathcal{M} = \{m_1 : \text{Celsius}, m_2 : \text{Fahrenheit}, m_3 : \text{kW}, m_4 : \text{W}/m^2\text{K}, m_5 : \text{percentage}\}$

In order to further visualise the semantic annotations and facilitate understanding, we formalise the encoding of the knowledge about the components in a layered hierarchical view, as informally introduced in Section 3.3.1. The following three layers are used: i) the "Feedback Control Components" layer which contains the

"things" that represent the actual implementations of the respective functions (i.e., plant, sensor, controller, actuator, etc.); ii) the "Profile" layer which contains the end-points of all components; iii) the "Semantic Annotation" layer which contains the sub-graphs that define the semantic annotation space Λ, including the domain specific knowledge, i.e., physical properties, measurement units and relative or direct features-of-interest / locations. The semantic models of the components are then graphically introduced, so as to enable subsequent understanding of the semantic reasoning towards extraction of control systems configurations.

Plant

For presentation clarity, we consider plants with one input-variable and one parameter, which are mapped to an output-variable. This is represented by the model shown in Fig. 3.9. In general, multiple end-points can be considered, simply by introducing more nodes of the classes $\mathcal{U}$, $\mathcal{Y}$ and $\mathcal{Z}$ respectively. The full semantic model of a plant consists of a *tree* that has as *root* the plant node. Then the *tree* has as many *branches* as the number of end-points of the component. For visualisation purposes, these branches are marked by green, blue and red edges respectively. It can be seen that the semantic annotations of the presented end-points are given by (3.4).

$$\lambda(v^p) = \{l_1 : \text{office}, q_2 : \text{heat-energy}, p_1 : \text{increase}, \mathcal{M}\}$$
$$\lambda(x^p) = \{l_1 : \text{office}, q_1 : \text{temperature}, p_2 : \text{state}, \mathcal{M}\}$$
$$\lambda(\zeta^p) = \{l_3 : \text{door}, q_3 : \text{U-value}, p_3 : \text{plantTopology}, \mathcal{M}\}$$

In the example, the input v^p corresponds to the heat energy introduced in the office, the output x^p corresponds to the office temperature state, while the parameter ζ^p corresponds to the thermal performance of the door (U-value). The grey edges in Fig. 3.9 represent the semantic relation between the measurement units ($\mathcal{M}$) and the physical properties ($\mathcal{Q}$), which is not used for the annotation of the plant itself.

Sensor

Adopting the same profile view of components, a sensor can be viewed as a component that receives as input some physical property and produces as output a signal or a time series measured in some measurement units. Fig. 3.10 illustrates

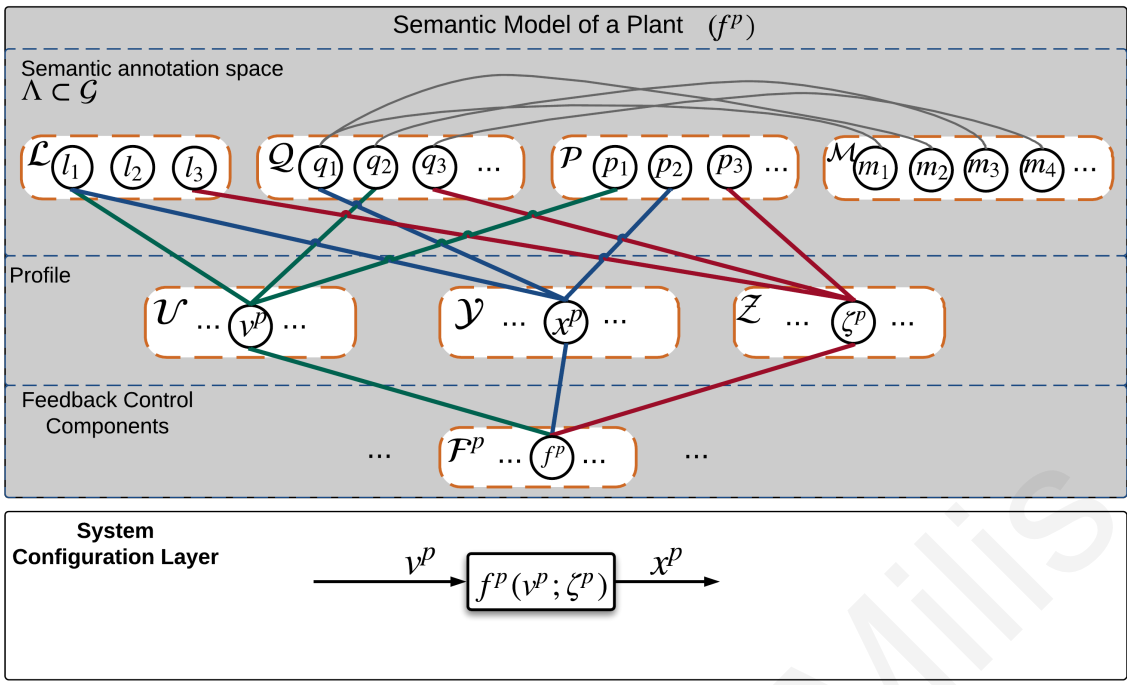


Figure 3.9: The semantic model of a certain plant f^p, with one input v^p, one output x^p and one parameter ζ^p

the semantic model of a sensor f^s. Similarly to the "plant" the semantic model tree associated with the sensor has "thing" f^s as a root node and three branches: one for the plant state x^s as input, one for the sensor measurement y^s as output and one for a parameter ζ^s. The semantic annotations related to the sensor's semantic model are given in (3.4). For presentation clarity, the annotations of parameters are omitted in this case and in the sequel.

$$\lambda(x^s) = \{l_3 : \text{door}, q_1 : \text{temperature}, p_2 : \text{state}, \mathcal{M}\}$$
$$\lambda(y^s) = \{l_3 : \text{door}, q_1 : \text{temperature}, p_4 : \text{stateMeasurement}, m_1 : \text{Celsius}\}$$

where the input x^s corresponds to the temperature at the window and the output y^s corresponds to the same signal measured in degrees Celsius.

Actuator

An actuator receives an input command or an input signal by a controller and produces an output signal that affects a physical property. The semantic model of an actuator f^a is given in Fig. 3.11. It can be seen that the semantic annotations of the actuator's input u^a and output v^a are given by (3.4).

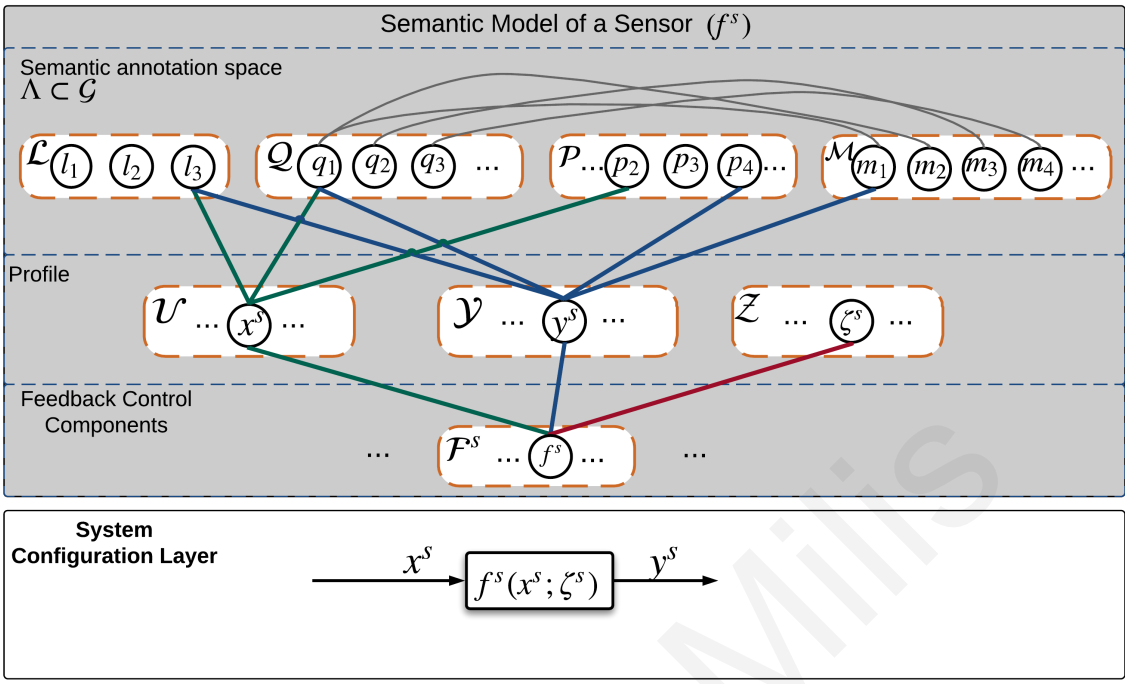


Figure 3.10: The semantic model of a certain sensor f^s, with one input x^s, one output y^s and one parameter ζ^s

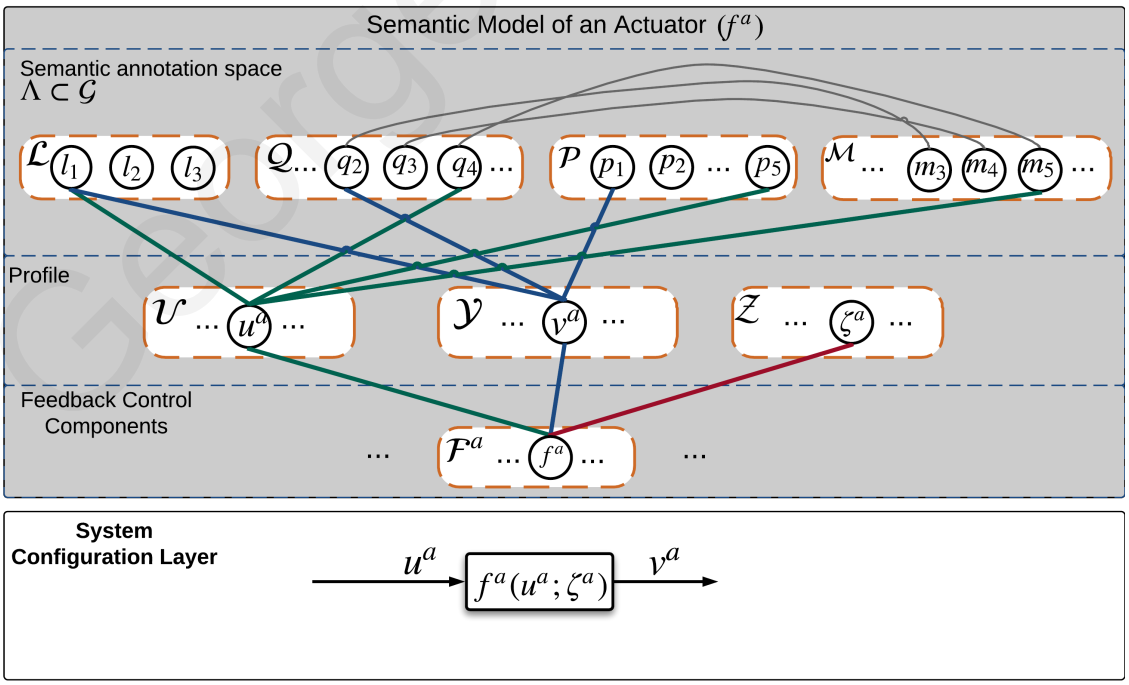


Figure 3.11: The semantic model of a certain actuator f^a, with one input u^a, one output y^a and one parameter ζ^a

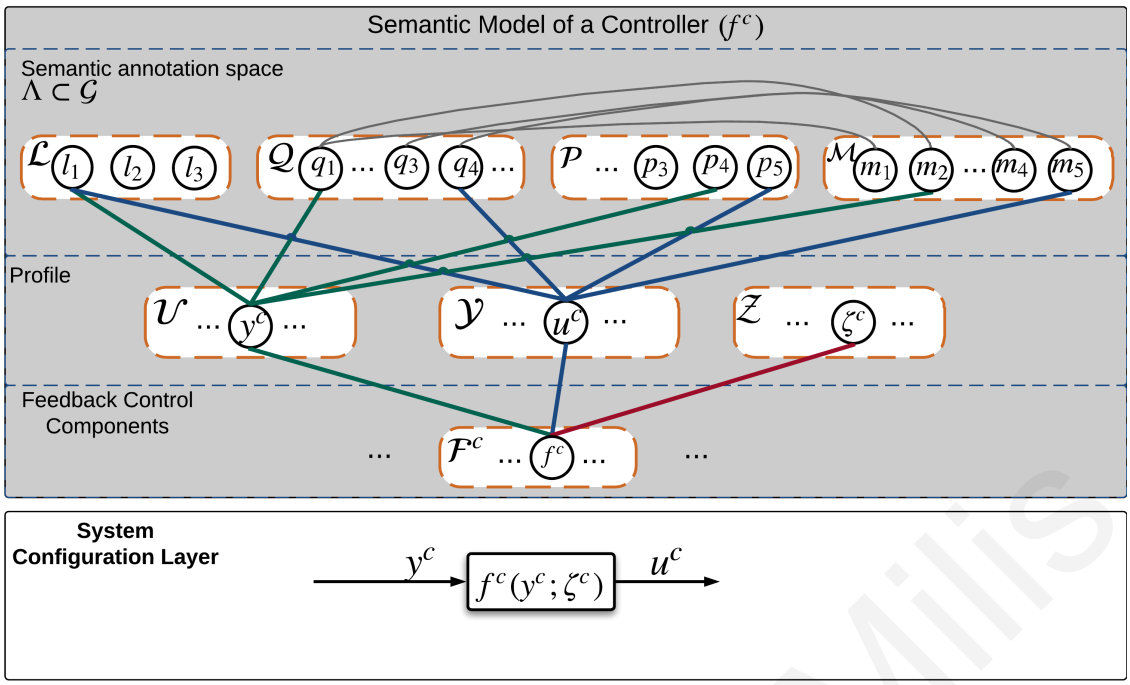


Figure 3.12: The semantic model of a certain controller f^c, with one input y^c, one output u^c and one parameter ζ^c

$$\lambda(u^a) = \{l_1 : \text{office}, q_4 : \text{valveOpening}, p_5 : \text{controlDecision}, m_5 : \text{percentage}\}$$

$$\lambda(v^a) = \{l_1 : \text{office}, q_2 : \text{heat-energy}, p_1 : \text{increase}, \mathcal{M}\}$$

that is, the actuator receives a valve opening signal in $[0, 1]$ and introduces respective heat energy in the office through hot liquid flow-rate in the pipes of the heating device and subsequently heat radiation or hot air blowing.

Controller

The semantic model of a controller is given in Fig. 3.12. The controller object f^c is the root node and has one input y^c, one output u^c and one parameter ζ^c. It is noted that the controller is typically considered in relation with a specific plant (either due to its design or due to its potential deployment) and, as such, it inherits the feature-of-interest properties from its input signal. The respective semantic annotations are given by (3.4).

$$\lambda(y^c) = \{l_1 : \text{office}, q_1 : \text{temperature}, p_4 : \text{stateMeasurement}, m_2 : \text{Fahrenheit}\}$$

$$\lambda(u^c) = \{l_1 : \text{office}, q_4 : \text{valveOpemning}, p_5 : \text{controlDecision}, m_5 : \text{percentage}\}$$

that is, the controller receives a temperature signal from the office in Fahrenheit and outputs a valve opening signal in $[0, 1]$.

The semantic models of the key components have been presented above, while the models for the Pre- and Post-Control Functions are not visualised but can be derived in a similar way, annotating their end-points. The use of the models will be made clear through more application use-cases in following Sections.

It is emphasised that the semantic models of the components, which include the semantic annotations of all their end-points, can be either imported manually by human experts or retrieved from pre-defined Internet sources hosting the structured information in the form of "semantic drivers" of components (e.g., through a Building Information Model [23] prepared during the design phase of a building). What is achieved by the proposed modelling, is the encoding of the required knowledge in machine readable format. The semantic matchings of the components and the subsequent semantic reasoning mechanism are discussed in the next sequel.

3.3.4 Semantic Reasoning

At this state, the semantic annotations of all end-points of components are considered pre-defined in the "Knowledge Graph" $\mathcal{G}$. Subsequently, the Supervisor Σ executes the "Semantic Reasoning" process to extract the n_I possible control system configurations at time k_I and subsequently executes the "Configuration Selection" process (see Section 3.3.7) to generate the decision signal σ_I (double orange line in Fig. 3.2) and select one of the options to be applied as $f^I(\cdot)$, based on pre-defined criteria. First, the "Semantic Reasoning" process considers these "Semantic Annotations" and tries to evaluate their "Semantic Matching". The latter is defined as the operator (3.4).

$$\rho : \Lambda \times \Lambda \mapsto \{\top, \bot\} \tag{3.4}$$

This operator takes as input a pair of semantic annotations and returns '$\top$' (true) if the two annotations share at least one point in the space Λ, otherwise it returns '$\bot$' (false). The Semantic Matching operator is used in two different cases:

1. Output-Input semantic matching: it checks all individual outputs of components (elements of the set $\mathcal{Y}$) to all individual inputs (elements of the set

$\mathcal{U}$), considering the type of components and their role in the feedback control loop. For instance, a sensor output signal annotated with the quadruple {office, temperature, stateMeasurement, Celsius} semantically matches with the input of a controller annotated with the quadruple {office, temperature, stateMeasurement, $\mathcal{M}$}. This can be a simple "proportional controller" whose gain is not affected by the specific measurement unit but it only depends on the difference from a respective reference value. Note that the controller's input annotation is a sub-space containing the single-point annotation of the sensor's output. Note also that the outputs of "Parameter Functions" and their semantic matching to existing parameters in the "Parameters Registry" (set $\mathcal{Z}$) are also included in this case.

2. Parameters semantic matching: it checks whether any of the parameters stored in the "Parameters Registry" $\mathcal{Z}$ semantically match with parameters required by components. For instance, a parameter required by a controller and annotated with the quadruple {office, temperature, referenceValue, Fahrenheit}, will match with a stored parameter annotated with the same quadruple.

Further to the direct matching between semantic annotations as described above, the "Semantic Reasoning" process explores also transformations of these annotations within the semantic annotation space Λ. Typically, the control system signals are assumed in spaces of real numbers and the transformations happen between such spaces of different dimensions. This is convenient during the design of fixed-configuration control loops, where the knowledge about the involved components and their variables is implicitly passed in the implementations by the human expert. However, if we want a machine to perform configurations of closed loops online, then we have to model explicitly the representation of the signals, and subsequently the transformations, in a higher-dimension space. That is, for example, the states vector can be denoted as $x \in \mathcal{R}^{n_x} \times \Lambda^{n_x}$ and the controlled inputs vector as $v \in \mathcal{R}^{n_v} \times \Lambda^{n_v}$ respectively. This way, each component not only transforms the real value of signals but it also affects their semantic annotations. For instance, a "state" signal with a semantic annotation given by {office, temperature, state, $\mathcal{M}$}, when measured by a temperature sensor with output in Celsius, will be transformed to a signal with semantic annotation {office, temperature, state, Celsius}.

Beyond the semantic transformations that happen when signals pass through

certain control system components, the semantic annotation of a signal can be transformed also using "semantic rules" [66], which implement deductive inference. Such rules comprise the encoding of expert knowledge about whether we can move from one point of the semantic annotation space Λ to another, without affecting the "true meaning" of the variable in the subject domain usage. For instance, the semantic annotation of a sensor's output {door 1, temperature, state, Celsius}, can be transformed to {office, temperature, state, Celsius} provided that the things "door 1" and "office" are linked through the relation "within", defined using the concepts of the GeoSPARQL ontology mentioned earlier. As defined in Section 3.3.1, these rules comprise composite relations as compositions of "relation graphs"; the edges of these graphs are not explicitly defined but they are implemented as paths of length > 2. The currently defined relations and semantic rules are exploited by the supervisor Σ in the reasoning process, to evaluate whether certain semantic transformations can be applied for enabling the semantic matching.

At each execution of the "Semantic Reasoning" process, following an event at time k_I, the supervisor iterates within the controllers and tries to match their inputs and outputs considering that a component can be used only if all its inputs match with outputs of other components and also all its required parameters match with parameters in the "Parameters Registry". The operation detects all semantically valid matchings between control system components and extracts the n_I candidate control system configuration options. From these valid options, one needs to be selected to operate the control system. The selection mechanism is described in Section 3.3.7, where also examples are given.

The following sub-sections go into further details on the semantic matching process between end-points of components.

Actuator-Plant-Sensor Matching

The first task of the Supervisor Σ when executing the "Semantic Reasoning" process is to find actuators able to act on the inputs of the plant subject to control, as well as sensors that can measure the outputs of the plant. The actuator-plant matching is shown in Fig. 3.13. For visualisation clarity, only the output branch of the actuator and the input branch of the plant are shown. Moreover, the parameters' matching is again omitted for simplicity and visualisation clarity. It can be seen that the semantic

annotation of the actuator's output ($\lambda(v^a) = \{l_1, q_2, p_1, \mathcal{M}\}$), is equal to the semantic annotation of the plant's input ($\lambda(v^p) = \{l_1, q_2, p_1, \mathcal{M}\}$).

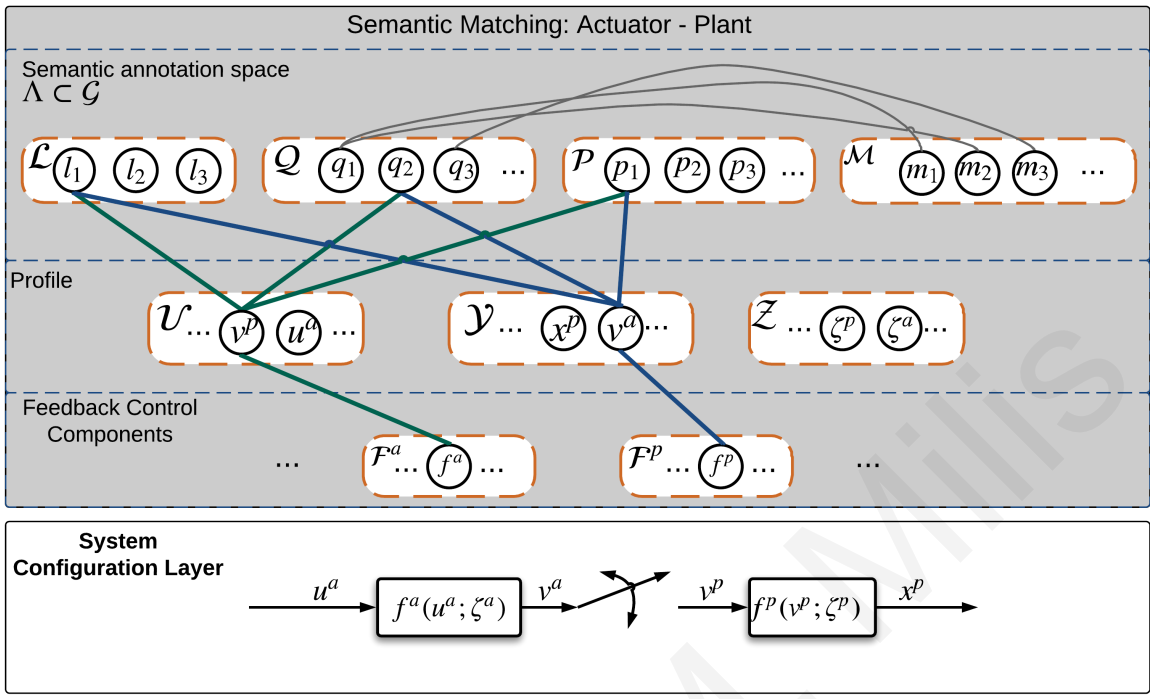


Figure 3.13: Semantic Matching: Actuator - Plant

The matching of a plant output to a sensor's input is not visualised, however, it is very similar to the one discussed above.

Semantic matching of a controller

Once the "Semantic Reasoning" process has found appropriate actuators and sensors, it moves on to find appropriate controller(s). The visualisation of the semantic matchings between a controller output and an actuator input, as well as between a sensor output and a controller input, is omitted as these can be derived from understanding the previous examples. However, we visualise the case where the output signals of certain components need to be processed before being fed as inputs to other components. For instance, there can be a function available and semantically annotated in the knowledge graph, which is able to transform a signal from one measurement unit to another. For example, the function $f^y(y^s) = \frac{9}{5}y^s + 32$ can be used to transform degrees Celsius (^oC) to degrees Fahrenheit (^oF), when required for temperature measurements. Fig. 3.14 illustrates the intervention of such function to transform the measurement unit of the signal produced by the sensor ('Celsius')

before this is fed to the controller that expects a signal in 'Fahrenheit'. The semantic annotation of the sensor's output is given by the set $\lambda(y^s) = \{l_3, q_1, p_4, m_1\}$, while the semantic annotation of the controller's input is given by the set $\lambda(y^c) = \{l_1, q_1, p_4, m_2\}$. No semantic matching can be confirmed with these two annotations passing through the respective operator. However, if the signal is processed first by the function $f^y(.)$, the output signal will have a transformed semantic annotation represented by the set $\lambda(y'_s) = \{l_3, q_1, p_4, m_1\}$. Furthermore, according to the defined relation graphs, the location l_1 : office "contains" the location l_1 : door, which means that the two locations exploit a semantic relation. This triggers a semantic rule which defines that: *a signal from an end-point of a component is associated with a subject feature-of-interest if it has a direct semantic relation with is or if it is associated with some other feature-of-interest that is "within" the subject feature-of-interest*. As a result, the semantic matching of the sensor and the controller becomes feasible through the function $f^y(.)$, the semantic relation "within" defined between the two locations (features-of-interest) and the aforementioned semantic rule that builds a composite semantic relation (see Section 3.3.1).

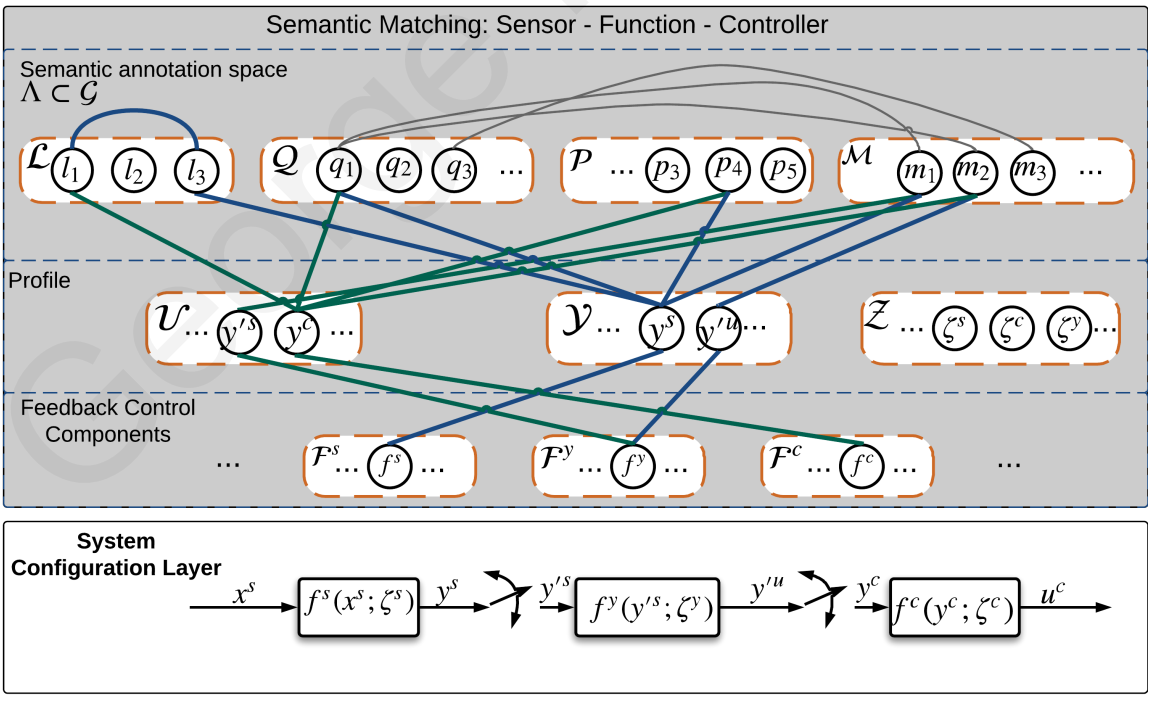


Figure 3.14: Semantic Matching: Sensor - Function - Controller

The transformation of control signals before being fed to an actuator can be modelled in a very similar way and is not presented here.

3.3.5 Use Case

Consider now the two-zone building of Fig. 3.15 consisting of two rooms with certain openings and a number of deployed monitoring and control components. Circles represent sensors (six in total), triangles represent actuators (one in total), diamond-shapes represent controllers (three in total) and rectangles represent processing functions (five in total). Assuming that we give this information to a control engineer and we ask him/her to manually design a heat regulation system for "Room 1", he/she will ask for additional information about the components, their characteristics, their capabilities according to manufacturer, etc.

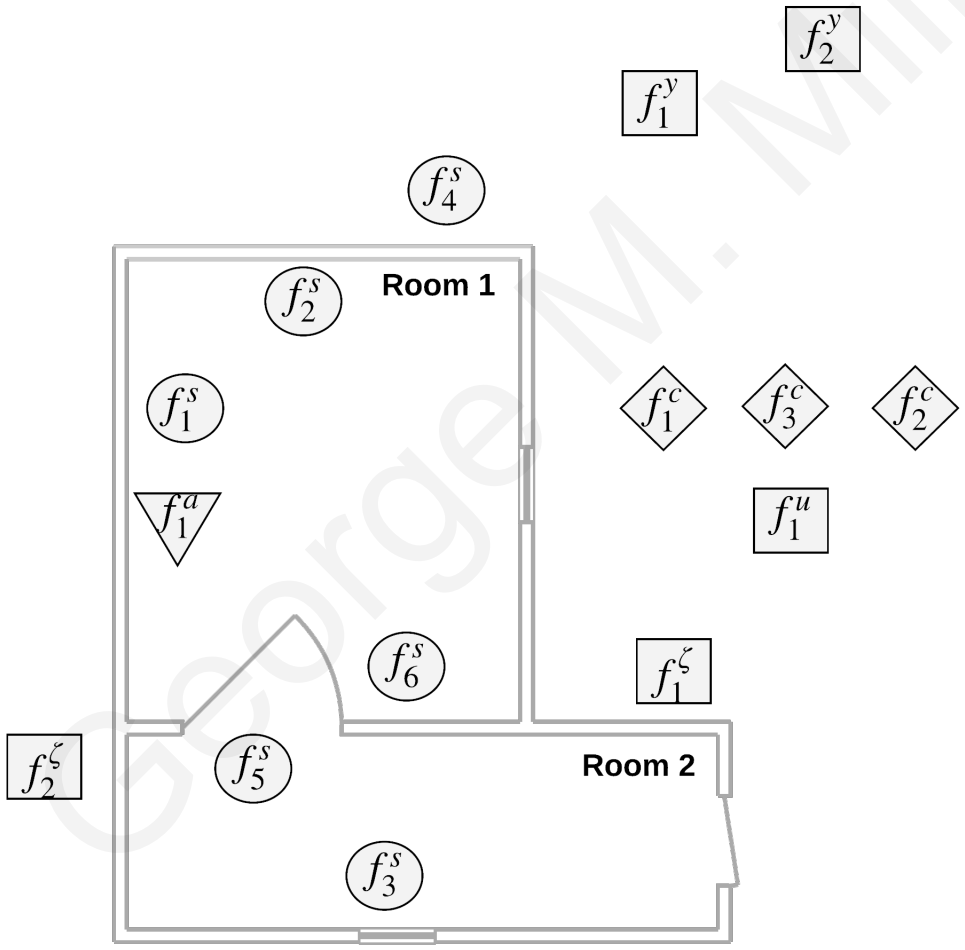


Figure 3.15: A building with two adjacent rooms. Circles represent sensors, triangles represent actuators, diamond-shapes represent controllers and rectangles represent processing functions. Details about the components are given in the context of the examples.

We can then provide the control engineer with all information about the deployed

components, in the form discussed in Sections 3.3.1 and 3.3.2 for the storing of knowledge about the domain, the components and their semantic annotations. First, the following list summarizes the domain knowledge currently available in the Knowledge Graph:

- $\mathcal{M}$ = {m_1 : Celsius, m_2 : Fahrenheit, m_3 : Joule, m_4 : W/m^2K, m_5 : on/off, m_6 : [0,100]}
- $\mathcal{Q}$ = {q_1 : temperature, q_2 : heat-energy, q_3 : U-value, q_4 : flowRate}
- $\mathcal{L}$ = {l_1 : room 1, l_2 : ambient, l_3 : door 1, l_4 : west wall 1}
- $\mathcal{P}$ = {p_1 : increase, p_2 : state, p_3 : plantTopology, p_4 : stateMeasurement, p_5 : controlDecision}

Second, Table 3.1 presents the semantic annotations of all end-points of the control system components shown in Fig. 3.15.

Table 3.1: The components' database with the respective semantic annotations of end-points

		Plant
f_1^p	Inputs:	$\lambda(v_1^p)$ = {room 1, heat energy, increase, $\mathcal{M}$}
		$\lambda(w_1^p)$ = {ambient, temperature, $\mathcal{P}$, $\mathcal{M}$}
	Outputs:	$\lambda(x_1^p)$ = {room 1, temperature, state, $\mathcal{M}$}
		$\lambda(x_2^p)$ = {room 1, occupancy, state, $\mathcal{M}$}}
		$\lambda(x_3^p)$ = {window, opening, state, $\mathcal{M}$}}
		$\lambda(x_4^p)$ = {door 1, opening, state, $\mathcal{M}$}
		$\lambda(x_5^p)$ = {room 2, temperature, state, $\mathcal{M}$}
		$\lambda(x_6^p)$ = {ambient, temperature, state, $\mathcal{M}$}
	Parameters:	$\lambda(\zeta_1^p)$ = {window, opening, topologyParameter, percentage}
		$\lambda(\zeta_2^p)$ = {door 1, opening, topologyParameter, percentage}
		Sensors
f_1^s	Inputs:	$\lambda(x_1^s)$ = {room 1, temperature, state, $\mathcal{M}$}
	Outputs:	$\lambda(y_1^s)$ = {room 1, temperature, stateMeasurement, Celsius}

f_2^s	Inputs:	$\lambda(x_2^s) = \{\text{ceiling 1, temperature, state, } \mathcal{M}\}$
	Outputs:	$\lambda(y_2^s) = \{\text{ceiling 1, temperature, stateMeasurement, Fahrenheit}\}$
f_3^s	Inputs:	$\lambda(x_3^s) = \{\text{room 2, temperature, state, } \mathcal{M}\}$
	Outputs:	$\lambda(y_3^s) = \{\text{room 2, temperature, stateMeasurement, Celsius}\}$
f_4^s	Inputs:	$\lambda(x_4^s) = \{\text{ambient, temperature, state, } \mathcal{M}\}$
	Outputs:	$\lambda(y_4^s) = \{\text{ambient, temperature, stateMeasurement, Celsius}\}$
f_5^s	Inputs:	$\lambda(x_5^s) = \{\text{door 1, opening, state, } \mathcal{M}\}$
	Outputs:	$\lambda(y_5^s) = \{\text{door 1, opening, stateMeasurement, percentage}\}$
f_6^s	Inputs:	$\lambda(x_6^s) = \{\text{room 1, occupancy, state, } \mathcal{M}\}$
	Outputs:	$\lambda(y_6^s) = \{\text{room 1, occupancy, stateMeasurement, } \{\top, \bot\}\}$
		Pre-control processing functions
f_1^y	Inputs:	$\lambda(^y_1) = \{\text{room 1, temperature, stateMeasurement, Fahrenheit}\}$
	Outputs:	$\lambda(y_1'^y) = \{\text{room 1, temperature, stateMeasurement, Celsius}\}$
f_2^y	Inputs:	$\lambda(y_2^y) = \{\text{room 1, 'temperature', stateMeasurement, Celsius}\}$
		$\lambda(y_3^y) = \{\text{room 1, 'temperature', stateMeasurement, Celsius}\}$
		$\lambda(y_4^y) = \{\text{room 1, 'temperature', stateMeasurement, Celsius}\}$
	Outputs:	$\lambda(y_2'^y) = \{\text{room 1, temperature, stateMeasurement, Celsius}\}$
		Controllers
f_1^c	Inputs:	$\lambda(y_1^c) = \{\text{room 1, temperature, stateMeasurement, Celsius}\}$
	Outputs:	$\lambda(u_1^c) = \{\text{room 1, flow rate, controlDecision, on-off}\}$
	Parameters:	$\lambda(\zeta_1^c) = \{\text{room 1, temperature, referenceValue, Celsius}\}$
f_2^c	Inputs:	$\lambda(y_2^c) = \{\mathcal{L}, \text{temperature, stateMeasurement, Celsius}\}$
	Outputs:	$\lambda(u_2^c) = \{\mathcal{L}, \text{flow rate, controlDecision, percentage}\}$
	Parameters:	$\lambda(\zeta_2^c) = \{\text{window 1, opening, plantTopology, percentage}\}$
		$\lambda(\zeta_3^c) = \{\text{door 1, opening, plantTopology, percentage}\}$
f_3^c	Inputs:	$\lambda(y_3^c) = \{\text{ambient, temperature, stateMeasurement, Celsius}\}$
	Outputs:	$\lambda(u_3^c) = \{\text{room 1, flow rate, controlDecision, percentage}\}$
		Post-control processing functions
f_1^u	Inputs:	$\lambda(u_1^u) = \{\text{room 1, flow rate, controlDecision, percentage}\}\}$
	Outputs:	$\lambda(u_1'^u) = \{\text{room 1, flow rate, controlDecision, on-off}\}$

	Actuators	
f_1^a	Inputs:	$\lambda(u_1^a)$ = {west wall 1, flow rate, controlDecision, percentage}}
	Outputs:	$\lambda(v_1^a)$ = {west wall 1, heat energy, increase, Joule}
	Parameter functions	
f_1^ζ	Inputs:	$\lambda(y_1^\zeta)$ = {room 1, occupancy, stateMeasurement, {⊤, ⊥}}
	Outputs:	$\lambda(\zeta_1)$ = {room 1, temperature, referenceValue, Celsius}
f_2^ζ	Inputs:	$\lambda(y_2^\zeta)$ = {ambient, temperature, state, Celsius}
	Outputs:	$\lambda(\zeta_2)$ = {ambient, temperature, uncontrolledInput, Celsius}
f_3^ζ	Inputs:	$\lambda(y_3^\zeta)$ = {door 1, opening, stateMeasurement, percentage}

According to the available information, two temperature sensors, f_1^s and f_2^s are deployed in "room 1", measuring the temperature of the office x_1^p, the first one in degrees Celsius (modelled by $m_1 \in \mathcal{M}$) and the second one in degrees Fahrenheit (modelled by $m_2 \in \mathcal{M}$). A third sensor, f_3^s measures the temperature of the "Room 2" in degrees Celsius, while sensor f_4^s which measures the ambient temperature in degrees Celsius. Sensor f_5^s measures the opening of the door on the wall that is shared between the two rooms. Finally, f_6^s measures the occupancy of "Room 1", indicating whether the room is occupant or not. The room is also equipped with a heating device f_1^a which introduces heat at some energy rate in kW. The heating device expects an input signal in the range $[0, 1]$ and produces an action proportional to those values. In practice, the input commands may be represented by different actual values, e.g., 0/1 meaning '1' for ON and '0' for OFF, or a percentage value in the range $[0, 1]$ or in the range $[0, 100]$. All these are stored as different knowledge objects of the type measurement units, $\mathcal{M}$.

In addition, appropriate controllers are deployed to regulate the temperature of the office by controlling the operation of the actuator. It is noted that we focus here on the knowledge modelling of control system components and the subsequent reasoning towards configuration selection and not on the specific implementation of the controllers. Therefore, for the sake of the example, it has been assumed that the Supervisor Σ has at its disposal, in the database $\mathcal{F}$, only the following three controllers; a threshold controller, a proportional controller and a periodic open-loop controller. These are presented below:

[Reminding note: the use of the variables' superscripts from the set $\{p, s, y, c, u, a, \zeta\}$ facilitates the differentiation between variables associated with the implementation of specific types of components from the set { plant, sensor, pre-control function, controller, post-control function, actuator, parameter-function }].

i. *Threshold Controller:* A bang-bang controller which produces an "on" signal when the measured value of a physical property is less than its desired value and an "off" signal when the value is greater. This controller receives a measurement value of some physical property as an input. The controller is given by (3.5), where a dead-zone is also considered to avoid reactions to measurement noise or other high-frequency oscillations introduced by the controller.

$$u_1^c(y_1^c(t), r_1^c(t), t) = \begin{cases} 1 & \text{if } (r_1^c(t) - y_1^c(t)) > \epsilon \\ 0 & otherwise \end{cases} \tag{3.5}$$

where $u_1^c(.)$ is the control decision produced by controller f_1^c, $y_1^c(t) > 0$ is the measurement of the state assumed by the controller, $r_1^c(t) > 0$ is the reference value of the state and ϵ is a small positive value that defines the size of the dead-zone. Such controller may be used to regulate the temperature of a room close to a reference value, during periods where the outside temperature is lower than the reference value.

ii. *Proportional Controller:* A controller that produces a signal proportional to the tracking error $(y_2^c(t) - r_2^c(t))$, which defines the level of operation of an actuator. The controller receives as input the measured value of a physical property, while it is also given a reference value and, optionally, a vector of other parameters of the plant (e.g., opening of the window), which help in calculating the gain for maintaining certain response properties, e.g. rise time. By default, the parameters are considered constant and are given to the controller upon deployment. This controller is given by:

$$u_2^c(y_2^c(t), r_2^c(t), t) = \Gamma(\theta)(y_2^c(t) - r_2^c(t)), \tag{3.6}$$

where $u_2^c(.)$ is the control decision produced by the controller, $y_2^c(t)$ is the measurement of a plant's state, as assumed by the controller, $r_2^c(t)$ is the reference value of that state of the plant, Γ is the proportional gain function and θ is a

vector of parameters which affect the gain calculation for stable control. Such controller may be used for the regulation of the temperature in a room at specific value.

iii. *Periodic Open Loop Controller:* An open-loop controller which produces an "on/off" signal to an actuator based on internal cycle configuration. The controller operates with a default cycle, exchanging between an "off" time-period followed by an "on" time-period. It also offers an input end-point through which the ambient temperature can be consumed to configure the "on" and "off" periods. For instance, the "on" period may be increasing from 50% to 100% proportional to the distance of the ambient temperature from an upper bound. The exact implementation of the function is out of the scope of this example. The controller is given by (3.7).

$$u_3^c(t) = \begin{cases} 1 & \text{if } nT \leq t \leq (n+a)T \\ 0 & \text{if } (n+a)T \leq t \leq (n+1)T \end{cases} \tag{3.7}$$

where T defines one operation cycle, i.e. the time between two consecutive turn-on events of the controller, $a \in (0, 1)$ is the parameter that configures the "on" and "off" periods in the cycle and it depends on the ambient temperature as explained above, $u_3^c(t)$ is the control decision produced by the controller f_3^c and $n \in \{0, 1, 2, ...\}$. Such controller might be utilised for the control of a ventilation fan, to clean the inside air at fixed cycles.

The components' database contains also a set of functions: i) the pre-control function $f_1^y(y) = \frac{5}{9}(y - 32)$, which transforms the temperature measurement unit Fahrenheit to the temperature measurement unit Celsius; ii) the pre-control function f_2^y that performs an averaged fusion of input signals; iii) the post-control function f_1^u that transforms a control signal given in percentage, to the control value expected by the heating device; iv) the parameter function f_1^ζ that lowers the reference value of the temperature to 18^oC instead of the default 25^oC, when the room is not occupied; and v) the parameter function f_2^ζ that reads the 'door 1' opening measurement and updates the Knowledge Graph by connecting the two rooms through the relation 'within' when the opening exceeds 80%. The reasoning behind this, is that the rooms' temperatures become almost equal if there is enough air flow across the opening.

Given the available components and semantic annotations, the control engineer performs the required reasoning and comes up with three possible control system configuration options, that are able to implement the heat regulation of "Room 1". These are shown in Fig. 3.16, with the blue double line, the yellow dashed line and the red line respectively. Note that the semantic annotations of parameters, and the subsequent semantic matchings, are omitted in the descriptions below, to reduce complexity.

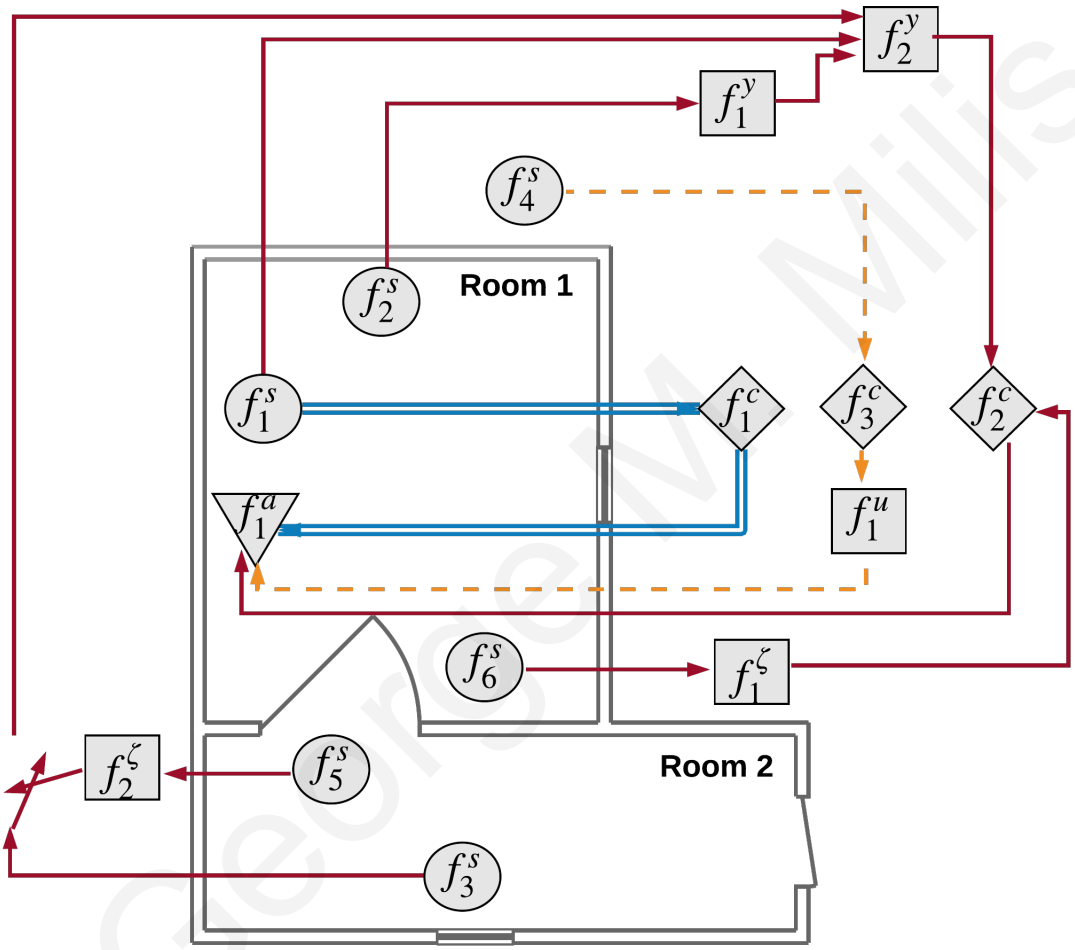


Figure 3.16: The building with two adjacent rooms as shown in Fig. 3.15, marking also the three detected configuration options; one with double blue line, another with dashed orange line and a third with red line

The configuration option marked with the double blue line considers the control system operating with a small set of components, comprising the temperature sensor f_1^s, the heating device f_1^a and the controller f_1^c that reads the room temperature and drives the heating device directly. Applying the semantic reasoning process using the information in Table 3.1, it can be seen that the first output of the plant f_1^p

semantically matches with the input of the sensor f_1^s, since:

$\lambda(x_1^p) = \lambda(x_1^s) = \{\text{room 1}, \text{temperature}, \text{state}, \mathcal{M}\}$.

Then, the output of f_1^s semantically matches the input of f_1^c, since:

$\lambda(y_1^s) = \lambda(y_1^c) = \{\text{room 1}, \text{temperature}, \text{stateMeasurement}, \text{Celsius}\}$.

Following the same logic, the control engineer checks the semantic matching of the output of the controller f_1^c to the input of the actuator f_1^a. It can be seen that the output of the controller does not define a specific feature-of-interest, whereas the actuator's input instantiates the respective variable as l : west wall 1. That is, the input's annotation comprises a point within the area formed by the output's annotation, thus the respective semantic matching is confirmed. Then, checking the potential matching of f_1^a output to the (controlled) input of the plant, we see that the first defines a point that does not lie within the area formed by the second. However, the feature/location *west wall 1* is *within* the feature/location *room 1* and this relation is assumed captured in the Knowledge Graph through a dedicated "Relations Graph". This enables the semantic transformation of the annotation of the actuators' output to a point that lies within the area formed by the annotation of the plant's input.

The above described semantic matchings are also illustrated as part of Fig. 3.17, using the graphical convention introduced in Section 3.3.2. The colouring convention, however, has been adjusted here to facilitate the illustration of the semantic matching for a complete closed-loop. For instance, all edges participating in the matching of the actuator to the plant are marked with a green line and similarly different colour is utilised to mark the semantic matchings between other types of components.

The second configuration option, marked with the yellow/orange dashed line, considers the use of the open loop controller f_3^c (this would be necessary if f_1^s stopped transmitting), which does not expect a state measurement, however uses the ambient temperature measurement from sensor f_4^s in order to set its operation cycle parameter α. The default cycle configuration parameter that defined the "on" period of the controller is assumed $a = 0.4$. Then, the "on" period of the controller becomes longer within the a pre-defined on-off cycle when the ambient temperature is lower than a pre-defined threshold. The output of the controller is processed through the post-control function f_1^u, which transforms the percentage value produced by the controller, to the on-off signal required by the heating device f_1^a. This configuration option is composed of the components $\{f_1^p, f_4^s, f_3^c, f_1^u, f_1^a\}$

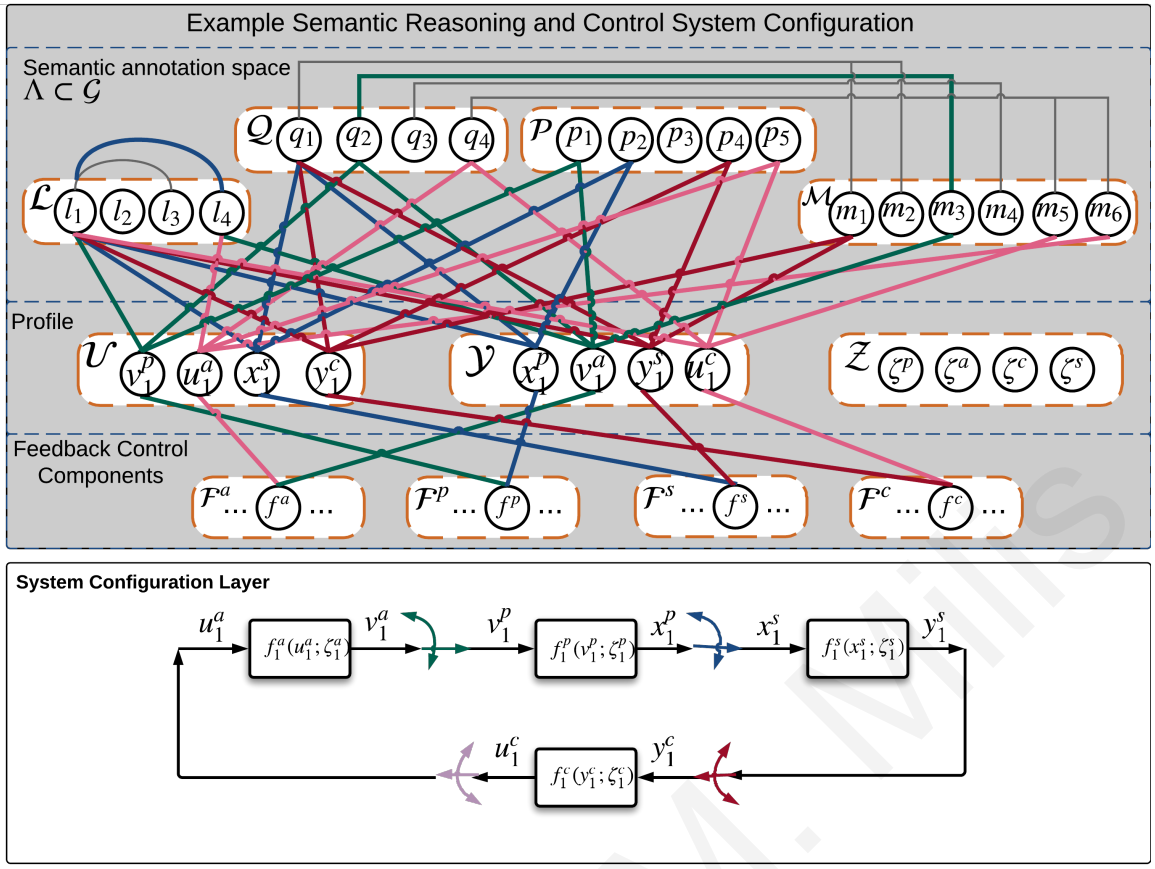


Figure 3.17: The configuration of a feedback control system, comprising the plant f_1^p, the actuator f_1^a, the sensor f_1^s and the controller f_1^c

The third configuration option, marked with the red line, still considers the heating device f_1^a as producing the signal for the plant's input v_1^p. However, it uses the controller f_2^c, which takes as input the output of f_2^y, which fuses the measurements from f_1^s and f_2^s (the latter's output is also processed by f_1^y for the transformation of the Fahrenheit signal m_2 : Fahrenheit to a signal in Celsius, m_1 : Celsius). In addition, function f_1^ζ utilizes the occupancy measurement and updates the temperature reference value, allowing the controller f_2^c to save energy. Moreover, the temperature of 'room 2', output of sensor f_3^s, can be also fused together with the other measurements when the opening of the door causes the change in the relation between the two rooms. The configuration option consists of the components: $\{f_1^p, \{f_1^s, f_2^s, f_3^s, f_4^s\}, \{f_1^y, f_2^y\}, f_2^c, f_1^u, f_1^a, \{f_1^\zeta, f_2^\zeta\}\}$.

Beyond the three configuration options identified above by applying the semantic reasoning process manually, when we execute the automatic implementation of the semantic reasoning process of the Supervisor Σ, we end up with nine (9) different

configuration options, as shown also in Table 3.2. The same table shows, in addition, the evaluation cost of each configuration option, assuming two scenarios in relation with the selection criteria defined later in Section 3.3.7; one with the two criteria given equal significance and another with the energy saving criterion given 80% significance comparing to 20% of the thermal comfort criterion. The time needed for the execution of the reasoning process and the configuration decision making is also given. It can be seen that, considering the reasoning as well as the testing of the configurations through simulation, the time to complete the process and select a single configuration option is about 6 minutes. The time-to-decision is considered an important factor when it comes to the adoption of this solution in real environments, since it may affect the stability properties of plants with fast dynamics. The latter is further discussed in Section 3.4.

Table 3.2: Results of configuration options and selection

Conf.	**Conf. Options**	**Cost (50-50)**	**Cost (80-20)**	**Time (min)**
$f_I(\cdot)$	$C_{(I,1)} = \{f_1^a, f_1^c, f_1^s\}$	0.2075	0.0830	
	$C_{(I,2)} = \{f_1^a, f_1^u, f_3^c, f_4^s\}$	0.1787	0.2716	
	$C_{(I,3)} = \{f_1^a, f_1^c, \{f_3^s, f_5^s\} f_1^\zeta\}$	0.2089	0.0829	
	$C_{(I,4)} = \{f_1^a, f_1^c, f_1^y, f_2^s\}$	0.2083	0.0831	
	$C_{(I,5)} = \{f_1^a, f_1^c, f_2^y, \{f_1^s, f_2^s, f_3^s, f_4^s\}\}$	0.2085	**0.0828**	5.94
	$C_{(I,6)} = \{f_1^a, f_2^c, f_1^s, f_1^\zeta\}$	0.1601	0.2071	
	$C_{(I,7)} = \{f_1^a, f_2^c, \{f_3^s, f_5^s, f_4^s\}\}$	0.1757	0.2627	
	$C_{(I,8)} = \{f_1^a, f_2^c, f_1^y, f_2^s\}$	0.1594	0.1887	
	$C_{(I,9)} = \{f_1^a, f_2^c, f_1^y, f_2^y, \{f_1^s, f_2^s, f_3^s, f_4^s\}, f_1^\zeta, f_2^\zeta\}$	**0.1561**	0.1808	6.58

Exploring more complex knowledge and semantic relations

It is emphasised that further to the above examples, the Semantic Reasoning process can be used for more advance transformation paths, e.g., given the occupancy of a room, derive implicit knowledge about the increase in temperature and/or Carbon

dioxide concentration. With the SEMIoTICS architecture, such knowledge is not necessarily available in advance, but it can be incorporated in the control system online through the update of the Knowledge Graph $\mathcal{G}$.

Consider that, at some time during the operation of the system, another sensor f_7^s is also deployed and measures the physical property 'opening' of the window (defined as $q_5 \in \mathcal{Q}$) in the measurement unit 'percentage', represented by the object $m_6 \in \mathcal{M}$. The semantic model of the plant is also enriched with a new output that represents the measurable window opening. The execution of the Semantic Reasoning process will lead to another control system configuration, where the sensor f_7^s will be confirmed as measuring the window opening, while the controller f_3^c may be adjusted to consume that measurement through an optional input for parameters, to update its gain using the new knowledge about the plant. Visual presentation of the resulted configurations are not given due to high complexity of the graphs.

3.3.6 The Configurations Graph

The next task of the "Semantic Reasoning" process is to build the "Configurations Graph" associated with the event at time k_I. This graph is modelled as: $\mathcal{C}_I = (\mathcal{N}_I, \mathcal{E}_I)$, where the elements of $\mathcal{N}_I$ are the end-points of components that have been confirmed as semantically matching with each other and the elements of $\mathcal{E}_I$ are the respective representations of potential signal flows in a control system configuration. Each configuration option $J = 1, ..., n_I$ is essentially a sub-graph $\mathcal{C}_{(I,J)}$, formed by parts of the above-mentioned graph.

Fig. 3.18 illustrates in diagrammatic form the resulted Configurations Graph from the semantic reasoning process in Section 3.3.5. It can be seen that the Configurations Graph is defined on a diagram showing the actual control system components. In order to show how to read the diagram, the configuration option $\mathcal{C}_{(I,1)}$ is marked again with double blue line.

3.3.7 Configuration option selection

As a final task, the Supervisor Σ explores the "Configurations Graph" (at each time k_I) and applies a pre-defined logic to select one of the detected configuration options and produce the decision signal σ_I, using pre-defined QoS criteria. The "Configuration

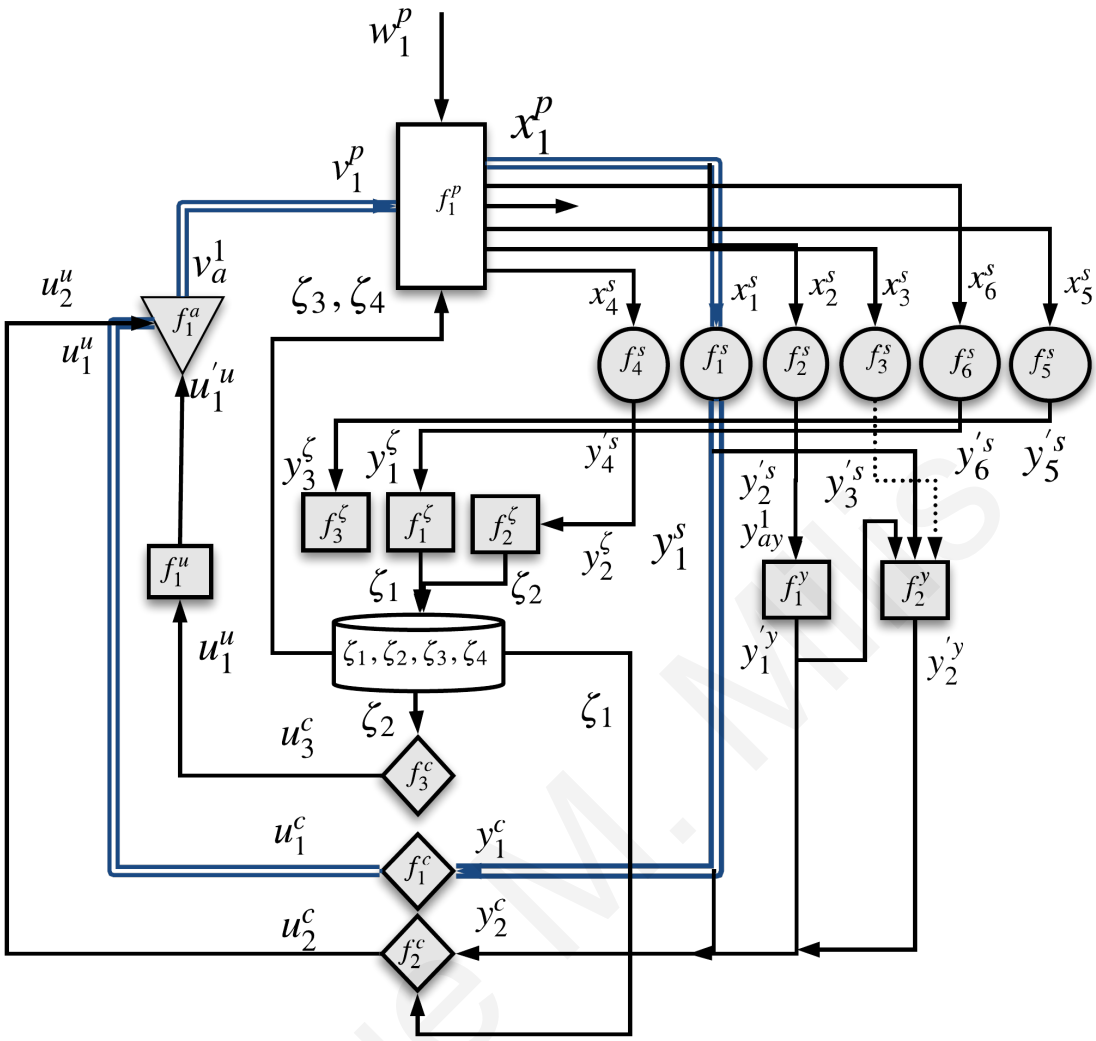


Figure 3.18: A diagrammatic representation of the Configurations Graph resulted from the Semantic Reasoning process

Selection" operation is formulated as in (3.8).

$$\underset{f_J^I(\cdot)}{\operatorname{argmin}} \sum_{j=1}^{n_\psi} \omega_j \psi_j(\cdot) \tag{3.8}$$

where the process iterates through all implementations $f_J^I(\cdot)$ of the n_I valid control system configurations at time k_I; we want to keep the one that minimizes the above formula. In this formula, $\psi_j, j = 1, ..., n_\psi$ define the pre-defined QoS criteria utilised for the selection process and $\omega_j, j = 1, ..., n_\psi$ define the weights assumed for each criterion in the current implementation. The weights take values in the range $[0, 1]$ and model the significance given to each criterion by the building operator.

To implement the logic of the configuration selection operation, a simple solution

would be to use the first one of the detected configuration options. Another option would be to associate the control system configurations with a rating mechanism, such as for the components participating in "well-performing" configurations to have their rating score increased by a certain value. A third option would be to test each configuration option for a certain amount of time on the real system and choose the one with higher performance against the pre-defined QoS criteria. A fourth option would be to test each configuration in a simulation, using pre-defined models for the plant and the actuators. Each option has its own advantages and drawbacks; the first two fail to provide high confidence about the real system operation and performance, while the third requires that untested configuration options are put in operation, with unpredicted behaviour. The fourth option makes the assumption that the installation of a SEMIoTICS architecture is accompanied by test-models of the plant and actuating components. These are not trivial to obtain, especially for the plant. However, recent work [100] already considers online generation of models for a fault diagnosis application in buildings.

In our work, we chose and implemented the fourth option. That is, each of the valid configuration options is passed through a fixed-time simulation test and is evaluated against the QoS criteria. The simulation test uses a pre-defined (test) model of the plant, instantiated with certain parameters so as to offer evaluation on equal basis. In addition, the actuators are also assumed accompanied by their model implementations for testing purposes. We define two cost-criteria ($n_\psi = 2$), as follows:

1. *Occupants' Thermal Comfort:* $\psi_1 = 1 - \frac{\mathcal{K}_i^c}{\mathcal{K}_i^{hc}}$

2. *Energy efficiency:* $\psi_2 = 1 - \sum_{k \in \tilde{\mathcal{K}}} (\frac{c_i(v_i(k))}{c_{max}})$

where $\tilde{\mathcal{K}} \subseteq \mathcal{K}$ defines the period of time during which the performance is evaluated, $\mathcal{K}_i^{hc} \subseteq \tilde{\mathcal{K}}$ is the period of time during which the zone i was occupant, therefore making the occupants' thermal comfort criterion applicable, $\mathcal{K}_i^c \subseteq \mathcal{K}_i^{hc}$ is the time period during which the zone i temperature remained within the comfort zone; without loss of generality, the latter is defined as the operation with zone i temperatures ($x_i(k)$) deviating from the desired values ($r_i(k)$) by more than ϵ. I.e., $|x_i(k) - r_i(k)| > \epsilon$. Moreover, $v_i(k)$ is the vector of inputs in zone i, $c_i(\cdot)$ is a function-vector representing the cost of producing the zone i inputs (it is noted that each heating device is

associated with its own cost function), c_{max} is a theoretical maximum cost, acting as normalization factor. The first criterion is based on the thermal comfort standards of ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) [86] where a metric of thermal comfort is defined as "the occupied time within a defined time period in which the environmental conditions in an occupied space remain inside the comfort zone". The ASHRAE defines also more complex thermal comfort models, taking multiple variables and factors into account, which can be incorporated if an application requires it. The second criterion considers the distance from the maximum normalised accumulated cost of producing the zone i inputs.

The above described method presents certain limitations. Specifically, large multi-zone buildings may be equipped with high numbers of IoT components, especially if we consider also the sensing infrastructure of modern smart phones. In such cases, the "Semantic Reasoning" process may end up with high numbers of configuration options, particularly due to the multiple alternatives of measurement signals. Therefore, passing all detected configuration options through simulation tests, becomes impractical. An immediate solution to the problem would be to test a random subset of n_I^* configurations from the list of detected options following time k_I (e.g., $n_I^* \in [2, 10]$). The actual number of options to be considered, may depend on the application and the reasonable re-configuration time-window. A drawback of this solution would be that it does not take into consideration any quality characteristics of configuration options. To address this, we define a "similarity measure" to make a more informed decision about the configuration options to consider for the control system. The "similarity measure" $|J_1 - J_2|_I$ is implemented as a logic-programming function, which calculates the difference between two configuration options $C_{(I,J_1)}$ and $C_{(I,J_2)}$ in terms of number and types of edges in $C_{(I,J_2)}$ that need to be added/removed in order to end up with $C_{(I,J_1)}$. We use this measure to select n_I^* configuration options to pass through the simulation test, by explicitly excluding the ones with similarity between them lower than a pre-defined threshold. That is, we require that $|J_1 - J_2|_I \geq \delta$ with δ being the lower similarity allowed between two exploited configuration options. This way, the configuration selection process guarantees that the tested options will not be very similar to each other, thus allowing wider exploration of the "Configurations Graph".

Section 3.4 discusses the scalability characteristics of the solution, triggered also

by the findings of Section 3.3.5 as presented in Table 3.2 in terms of the time required for the selection process to be completed.

3.3.8 Illustrative Scenarios Execution

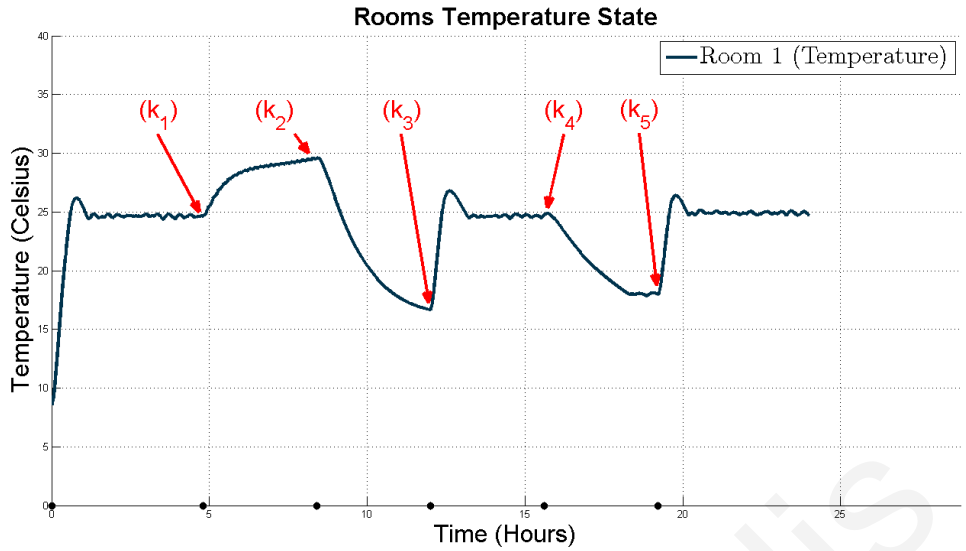
Here we go through an execution of a scenario where several components' changing events are introduced, leading to SEMIoTICS re-configuring the control system in-line with the use case presented earlier in Section 3.3.5. As expected, SEMIoTICS utilises the each time available knowledge and makes decisions for the reconfiguration of the system so as to continue offering the required control service without downtime and at the extent possible.

We repeat that the objective of the control system is the heat-regulation of the temperature of Room 1 at 25 ^oC. Fig. 3.19 shows on the top sub-figure the temperature of Room 1 through a 24-hour simulation where five changes of components are introduced. The bottom sub-figure shows the respective heat energy input introduced by the heating devices. The scenario is described below:

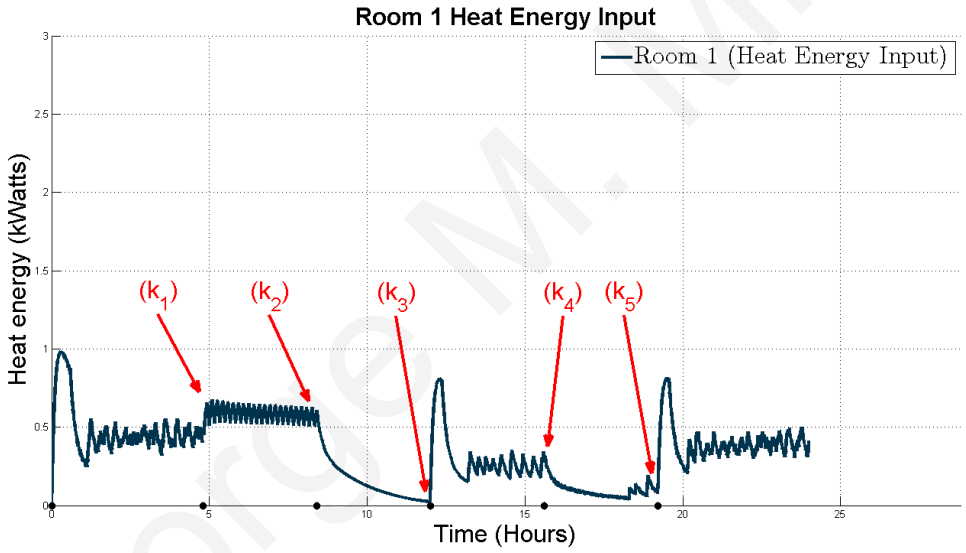
At $k_0 = 0$ (i.e., the initial deployment and configuration of components is assumed happening at $k = 0$), the control system is operating with the configuration marked with double blue line in Fig, 3.18, that is, with a small set of components comprising the temperature sensor f_1^s, the heating device f_1^a and the controller f_1^c. It can be seen that the controller drives the actuator and increases the temperature to reach the reference value and maintains it there according to its capabilities. Note that the ambient temperature is modelled as varying between 6 and 15 degrees Celsius in a 24-hour cycle.

Later during the operation of the system, at time $k_1 > k_0$, the sensor f_1^s stops transmitting. SEMIoTICS executes the Semantic Reasoning and the Configuration Selection processes, so as to re-configure the feedback control system. Since no measurement is available, the result is to select the open loop controller to drive the heating device. We assume also that the ambient temperature measurement becomes available. This configuration is the one marked with orange dashed line in Fig. 3.18. It can be seen that the temperature regulation performance degrades due to the different capabilities of the open loop controller.

At time $k_2 > k_1$, the building operator installs a new sensor, f_4^s, which measures the temperature of Room 1, however, in degrees Fahrenheit. This results in re-



(a) Room 1 Temperature State over the simulation time



(b) Room 1 Heat Energy Input over the simulation time

Figure 3.19: A 24-hour simulation of the plant's control operation. At k_1: The sensor f_1^s stops transmitting and SEMIoTICS re-configures the control system to operate with the available open loop controller; At k_2: Sensor f_4^s is installed, however, measuring in degrees Fahrenheit; At k_3: A pre-control function becomes available, which transforms degrees Fahrenheit to Celsius; At k_4: Additional sensors become available, measuring Room 1 temperature, occupancy of room, opening of door connecting the two rooms. Several pre-control and parameter functions are available as well and Room 1 is not occupant; At k_5: Room 1 is occupant again, while the opening of the door causes the Room 2 temperature to be considered a valid measurement for Room 1 temperature, with certain weight.

configuring the system which operates again with the controller f_1^c. It can be seen that the wrong interpretation of the signal by the controller results in turning into the "off" state and letting the temperature to drop. Then, at time k_3, the pre-control function that transforms the degrees Fahrenheit to degrees Celsius becomes available. SEMIoTICS re-configures the control system to use that functionality and the control system operates again as required.

At time k_4, more sensing capabilities become available: sensor f_3^s measures the temperature of Room 2, sensor f_5^s measures the opening of the door in the wall separating the two rooms and sensor f_6^s measures the occupancy of Room 1. During that time, a user of the first room has an activated temperature measurement as well on her mobile device (f_2^s). In parallel, the pre-control temperature fusion function becomes available, which provides a weighted average of the outputs of f_1^s and f_2^s. Moreover, a parameter function becomes available, which uses the occupancy measurement to lower the reference temperature to 18^oC. The scenario assumes at this event that Room 1 is not occupant. SEMIoTICS re-configures the system and chooses the controller f_2^c to drive the heating device. It can be seen that the temperature of the room is left to drop to the new reference value and then it is regulated there with lower heating power, thus also saving energy.

Finally, at time k_5, Room 1 becomes occupant again. Moreover, a pre-control function becomes available, which monitors the opening of the door and changes the knowledge in the "Knowledge Graph" marking the Room 2 as practically being part of Room 1. As a result, the controller makes use of the new knowledge and the fusion function considers also the temperature of Room 2 (f_3^s) in its weighted average. Since the room is occupant, it can also be seen that the controller brings the temperature back to 25^oc.

It is emphasised that the re-configurations of the system happen online, adopting the design of the processes presented in the previous sections.

3.4 Complexity and Scalability

The online reconfigurability features of SEMIoTICS do not come without a cost. The Semantic Reasoning process is inherently a decision problem of combinatorial nature, since it involves searching through the "Knowledge Graph" aiming to build the "Configurations Graph" and subsequently searching through the "Configura-

tions Graph" to detect the sub-graphs that meet certain criteria so as to comprise feedback control system configuration options. The process is executed every time SEMIoTICS is informed of a change in the "Knowledge Graph", i.e. annotation of new components or changes of plant topology, etc.. Depending on the application, the time required for the process to complete and return the list of configuration options may be an important factor. This time is associated with the computational complexity of the process execution, which is typically a challenge in combinatorial problems. We investigate here the computational complexity of the semantic reasoning process and how this associates with the scale of the problem, i.e. the size of the building and the number and types of deployed control system components. The findings provide useful information related to the scalability of the solution.

Computational complexity typically concerns how algorithms scale as their input size increases, and is measured either in terms of the time needed to solve the problem or in terms of the size of the input vector. The semantic reasoning process is a decision problem, which takes as input the state of the "Knowledge Graph" at time k_I (following an event of change in components), denoted as $\mathcal{G}_I$. The state of the "Knowledge Graph" comprises a set of stored knowledge facts, i.e. set of triplets formed by edges and their adjacent nodes (see Section 3.3.1).

The size of the input to the process is defined by the cardinality of the set of stored knowledge facts. The cardinality of this set is associated with the cardinalities of the sets in Table 3.3. The sets refer to the execution of the semantic reasoning process at time k_I, however, pointer I is omitted in the Table for clarity of the notation. In this work we do not emphasize on designing the decision algorithm with the lowest computational complexity to execute the semantic reasoning process; we are rather concerned with gaining insight to the scalability of the solution, given our existing algorithm and implementation.

The first five rows refer to sets that may evolve more frequently during the operation of SEMIoTICS and can become of big size, the subsequent five rows refer to sets with rather few elements that are expected to evolve much less frequently (e.g., by annotating new physical properties or locations in a building), and the last row refers to a set that depends on the semantic connectivity between the available components in the "Configurations Graph" and it cannot be pre-estimated. Although the actual execution time of the semantic reasoning process is an important factor, it highly depends on the hardware infrastructure on which SEMIoTICS runs.

Table 3.3: Sets participating in Semantic Reasoning

Set	Description
$\mathcal{F}$	The set of annotated control system components, with cardinality n_f.
$\mathcal{U}$	The set of annotated inputs of control system components, with cardinality n_u.
$\mathcal{Y}$	The set of annotated outputs of control system components, with cardinality n_y.
$\mathcal{Z}$	The set of annotated parameters of control system components, with cardinality n_ζ.
Θ	The set of registered parameters in the Parameters Registry, with cardinality n_θ
Ω	The set of different types of control system components (i.e., sensors, actuators, controllers, pre-control functions, post-control functions, parameter functions), with cardinality n_ω.
$\mathcal{Q}$	The set of capabilities that the signals can associate with, with cardinality n_q
$\mathcal{L}$	The set of features-of-interest defined for the underline system topology, with cardinality n_l.
$\mathcal{P}$	The set of properties of features-of-interest the signals can associate with, with cardinality n_p.
$\mathcal{M}$	The set of measurement units the components can associate with, with cardinality n_m.
$\mathcal{E}$	The set of edges, i.e. matching pairs of outputs and inputs, in the respective Configurations Graph.

Therefore, the findings presented here cannot be translated to time-complexity; they rather comprise indications about the scalability of SEMIoTICS.

The semantic reasoning process is implemented as a set of logical queries, which essentially query the Knowledge Graph and subsequently the Configurations Graph to extract certain sub-graphs. Assuming that we want to retrieve all knowledge facts in the graph, then the respective query will search through all different types of edges and all their respective pairs of adjacent nodes. That is, the query would require a total of $n_N n_E n_N$ executions of the basic operation of a triplet matching (the latter is considered covering also the search within pre-indexed lists). This number roughly defines the complexity of the query, which is polynomial in the space of input size $n_N^2 N_E$ ($PSPACE(n_N^2 N_E)$). In practice, the search in the semantic reasoning process is performed on pre-defined relation sub-graphs, which reduces the complexity to $PSPACE(n_N^2)$.

The core logical query of the semantic reasoning process, is the "Output-to-Input Semantic Matching". The query takes as input an output of a component and an input of another component or a parameter of a component and a parameter in the registry $\mathcal{Z}$. Since the semantic annotation space is defined by a quadruple of graph nodes in different relation sub-graphs, the query involves searching within the four different relation graphs. Taking for example the relation graph that connects end-point nodes to a "physical property" node, the worst case will appear when the output node is linked to a different "physical property" node than the input node, with the two "physical properties" associated through a third relation graph. This will require n_q^2 instantiations of the triplet matching operation. The same holds for the other three pairs of relation graphs and a matching is only confirmed if the operation succeeds in all cases. Therefore, the output-input semantic matching query requires $n_q^2 n_l^2 n_p^2 n_m^2$ instantiations of the triplet matching operation, which gives $PSPACE(n_q^2 n_l^2 n_p^2 n_m^2)$ complexity. This core query is used extensively throughout the semantic reasoning process, therefore we refer to its complexity as $O(sm)$ for simplicity. The input space of this query comprises very rarely updated and rather small numbers.

Building on the above core query, we analyse the rest of the queries executed by the semantic reasoning process and we present their resulted complexity in Table 3.4. Query sq_1 concerns the composition of pairs of component types, e.g. plants to sensors, controllers to actuators, and so on, query sq_2 concerns the building of

the Configurations Graph and depends on multiple executions of sq_1, and query sq_3 navigates through the already created Configurations Graph and extracts all valid configuration options. It can be seen that the theoretical results reveal complexity figures which can become challenging for very large input size.

Table 3.4: Complexity of Semantic Reasoning Queries

ID	Query	Complexity
sq_1	Pairwise component-types composition	$PSPACE(n_f^2 n_y n_u O(sm))$
sq_2	Build Configuration Graph	$PSPACE(O^8(sq_1)) = PSPACE(n_f^{16} n_y^8 n_u^8 O^8(sm))$
sq_3	Extract configuration options	$PSPACE(m_I^{(ps)} m_I^{(sy)} m_I^{(syc)} m_I^{(cau)} m_I^{(ua)} m_I^{(ap)} m_I^{(s\zeta)} m_I^{(\zeta\theta)})$

It can be seen that the complexity of the semantic reasoning process depends on the number of components and subsequently the number of end-points (inputs, outputs and parameters) these components have, as well as on the connectivity between the components. That is, bigger number of components that do not increase connectivity (e.g., a building to which we add zones where each zone is separately equipped with certain sensors and actuators) is not expected to have high impact on the complexity of the extraction of configuration options. On the other hand, a small number of components with high connectivity (e.g., adding redundant sensors and actuators in a zone) will affect all three queries. As mentioned, we do not attempt here a thorough theoretical analysis of the complexity, since this would highly depend on the specific design and implementation of the semantic reasoning process. We rather study the scalability of the solution in practical terms. We experiment with a maximum of 1000 components, assuming SEMIoTICS could run on a 20-floor building with 10 zones per floor and 5 components per zone on average. Bigger buildings or groups of buildings can make use of multiple instances of SEMIoTICS with certain information exchange between them, however, we do not consider such architectures in the framework of this Dissertation. The simulation results help us derive useful conclusions about the scalability characteristics of the solution.

Table 3.5 shows indicative experiment results from executions on an average

laptop with Intel Core i5 7th Gen processor. The columns of the table present the time (in seconds) for the following tasks:

Task 1: Build the Configurations Graph

Task 2: Extract all control system configuration options

Task 3: Test the performance of a single control system configuration option, towards Configuration Selection

Task 4: Put the selected control system configuration option in operation

Table 3.5: Scalability experimental results

Experiment	**Task 1**	**Task 2**	**Task 3**	**Task 4**
5 components, 1 configuration option	1.6	1.76	1.77	1.23
20 components, 1 configuration option	2.26	2.55	3.59	2.90
100 components, 1 configuration option	1.82	2.05	4.85	3.24
100 components, more than 500 configuration options	1.81	∞	-	-
100 components, 50 configuration options	1.53	10.2	5.5	4.14
1000 components, 100 configuration options	4.4	194.4	7.5	6

Note that the table reports the time for a single evaluation of a configuration option; this would be roughly multiplied by the number n_l of configurations considered for the selection of the best performing one. The results help us derive useful conclusions about the scalability characteristics of the solution. It can be seen that

the process of building the Configurations Graph takes around 1.5 – 4 seconds for the whole range of the experiments input size and as such it is not expected to be a scalability bottleneck for using SEMIoTICS in large buildings. Then, the process of extracting the configuration options from the "Configurations Graph" highly depends on the configurations' redundancy and it is not affected by the number of components in the building. We observe that a very large number of configuration options causes the system to stop responding, which creates a significant scalability issue. We overcome this issue by cutting the number of considered configuration options to a manageable level ($n_I < n_I^*$), depending on the application (see Section 3.3.7). Finally, the time required to test a configuration option, as well as the time required to put one in operation, increase with the number and heterogeneity of components, however, it is not expected to negatively affect the scalability for the size of the considered buildings and use cases with the same time-response characteristics, like the heating control in multi-zone buildings.

3.5 Implementation Details

Fig. 3.20 presents the tools and technologies adopted for the implementation of SEMIoTICS modules, as well as the interfacing and communication technologies used for the interaction between modules. The subject diagram is positioned on top of a faded version of the SEMIoTICS architecture, in order to facilitate the understanding of the direct relation between tools/technologies on one hand and specific modules and communication links on the other hand.

More specifically, the following dynamics of the plant, as well as the orchestration of the simulation scenarios are implemented as *Matlab* scripts [115]. The control system components are implemented as individual Matlab functions, with certain inputs and outputs, in-line with the syntax presented in Section 2.2. It is noted that the functions implementing the components of type "Sensor" and "Actuator" comprise virtual simulations of their expected physical operation; the functions implementing the rest of the components which are on the cyber part, i.e., "Controllers", "Pre-Control Functions", "Post-Control Functions", "Parameter Functions" and "Parameters Registry" may serve as real implementations as well. The invocation of the components' functions is performed in Matlab, with direct function calls using the information from the configuration selection decision.

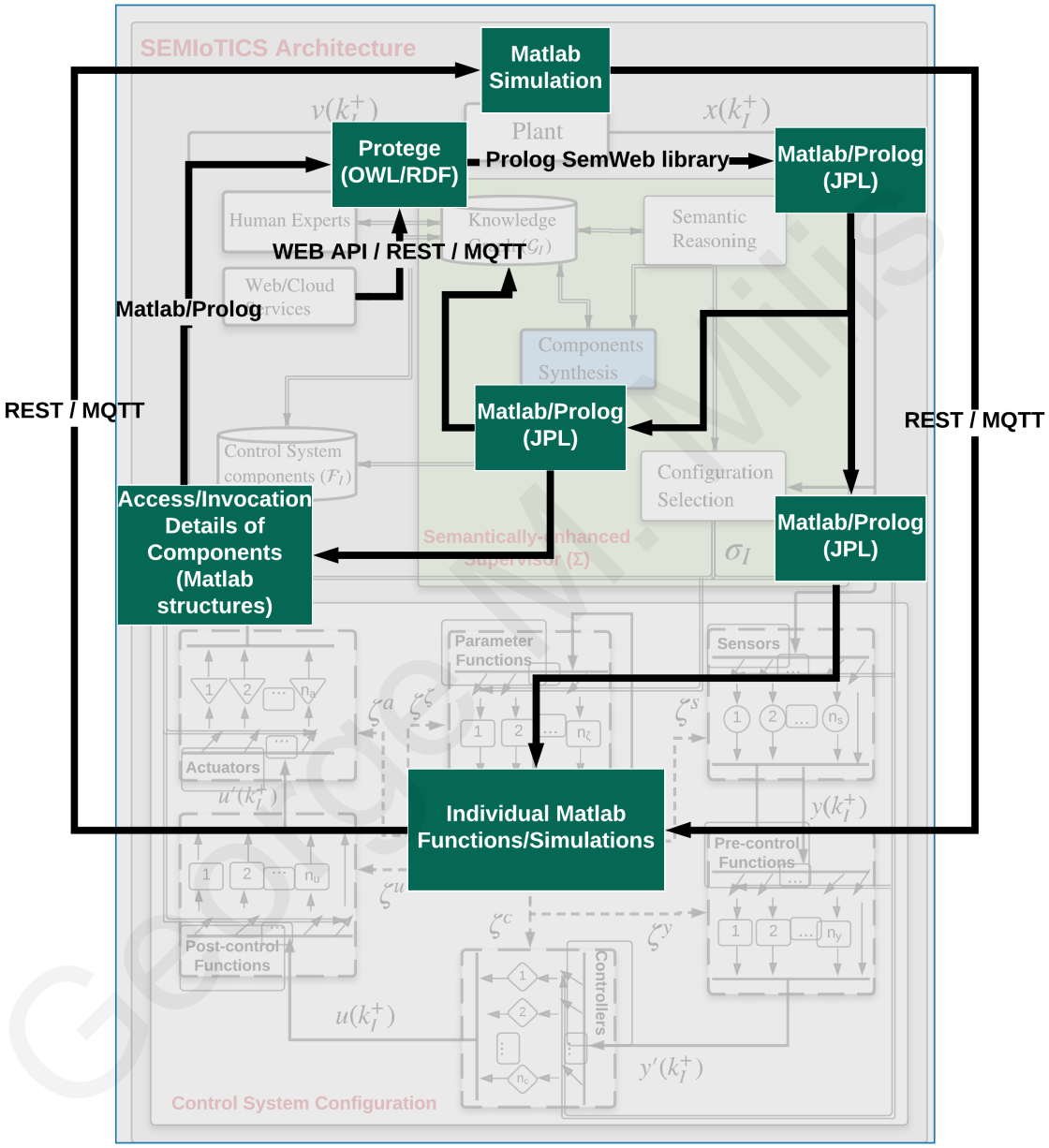


Figure 3.20: A diagrammatic representation of the tools and technologies adopted for the implementation of SEMIoTICS

It is noted that in a real implementation of SEMIoTICS, the components are located either in remote physical locations in the plant or in remote Web/Cloud locations of providers. The interaction would be implemented through dedicated RESTful APIs [99] or implementations of the widely used MQTT communication protocol [83] in IoT applications. Furthermore, although it is out of the scope of this Dissertation, we emphasize that a real implementation should consider the communication and energy/battery limitations of remote IoT devices, especially due to the need for IoTs to communicate additional information beyond the signal values. The research community is progressing fast in investigating solutions that overcome these limitations and that remove any barriers from adopting the IoT paradigm and subsequently the SEMIoTICS architecture. The most promising solutions come from the domain of "edge computing". For instance, the authors in [120] propose a framework called AdaM, which can run on the hardware of IoTs and implement adaptive sampling and adaptive filtering of data streams, thus saving energy and communication cost by adapting the communication based on the characteristics of the signal. Another interesting approach is the one of [122], where instead of communicating data, the IoTs learn a model of the data streams they produce and exchange only the parameters of the model with a central processing unit that also helps them adjust those parameters.

Web/Cloud services communicate with the "Knowledge Graph" through the aforementioned communication protocols as well. As already mentioned, we do not specifically implement the communication with these protocols in the framework of this Dissertation. We rather simulate any relevant interactions in the scenarios e.g., reading temperature data from a Web service of a local weather station. We are currently working towards implementing the protocols part by integrating SEMIoTICS in existing IoT platforms.

The "Knowledge Graph" $\mathcal{G}$ is implemented in *Protege* [111]. This is a widely adopted and very powerful framework tool for creating and linking ontologies (knowledge models) and is provided for free by the University of Stanford, USA. The knowledge models we reuse, develop and merge, are exported through Protege in the "Terse RDF Triple Language" (Turtle) syntax [125] of the OWL/RDF data models. This format is similar to the one used by SPARQL semantic query language [121].

Human users are able to interact with the "Knowledge Graph" either directly through the Protege tool (e.g., this is applicable for the responsible knowledge engi-

neer of a SEMIoTICS instance) or through dedicated Web-based interfaces. We do not specifically implement the latter in the framework of the Dissertation; we rather simulated in Matlab any such interaction, e.g. a technician creating the semantic annotation of a newly installed sensor or a domain expert introducing a new physical property and its measurement units.

The "Semantic Reasoning" is implemented using the *SWI Prolog* [2] which is a free implementation of the *Prolog* logic programming language. The selection of Prolog was done due to the inherent flexibility it provides to not only query the knowledge graph but also implement the required higher-level reasoning rules for the control system configuration options' extraction, with very good performance. The communication between Matlab and Prolog is achieved through JPL [41], which is a Java interface to Prolog. We have used this interface directly through Matlab, since the latter is built on top of Java and provides transparent integration.

Alternatively to the use of Prolog, e.g. in case a Prolog engine/server is not available, implementations of SEMIoTICS may consider the use of *SPARQL* [121] that is a pure query-script language that facilitates running pattern-matching queries on knowledge graphs. This can be used in Internet applications, as well as directly on IoT devices, since it has smaller computational power requirements. The drawback would be some decrease in computational efficiency, since SPARQL implementations make use of storage means that do not transparently work with graph data structures, thus generating additional needs for data processing at the low level. A much better solution in terms of computational efficiency would be the "GraphX" [68], which is a lightweight graph processing and querying library. GraphX overcomes the main drawbacks of SPARQL implementations, by employing pure RDF-compatible data representation structures, thus being able to support many of common computations on graphs. Still, Prolog provides a more powerful and computationally efficient tool for the implementation of the semantic reasoning process and should be considered in applications where the use of a Prolog engine is not excluded. Besides the pure graph-based operation and its efficient backtracking search algorithm, the key advantage of Prolog is the ability to write higher-level functions to perform multiple lower level reasoning, thus hiding the complexity of the logic operations from the application layer.

The communication of Prolog with the "Knowledge Graph" is achieved through the use of the *SWI-Prolog Semantic Web Library 3.0* [114], which provides inherent

logic programming functions for the interfacing with the Turtle format of the Graph.

The implementation details of the Configuration Selection process are similar to the ones of the Semantic Reasoning process. That is, a dedicated Matlab function undertakes the interaction with the rest of the modules, while Prolog is used for the execution of the required semantic queries and the returning of the logically valid results from the Knowledge Graph.

Finally, the diagram presents also the Components Synthesis process/module, which again has same implementation details as the Semantic Reasoning process. This module, as well as its interaction with the Semantic Reasoning process and the Knowledge Graph and Components Database, will be introduced and detailed in Chapter 4.

Chapter 4

Online Synthesis of Distributed Feedback Control Schemes

In this Chapter we describe how we further exploit the capabilities of SEMIoTICS, to achieve the online synthesis of components, which can take part in semantically valid control system configurations. Without loss of generality, and in order to improve the readability of the formulation, we consider the building shown in Fig. 4.1, comprised of three rooms/zones. The objective is to apply heating control in the building, assuming homogeneous heat capacity in each zone and thus modelling the inside temperature with the state-vector $x(k) \in \mathcal{R}^3$. We also assume homogeneous ambient and floor temperatures, modelled by the vector $w(k) \in \mathcal{R}^2$.

The building is assumed controlled by a Building Management System (BMS), which is equipped by an instance of SEMIoTICS, aiming to allow online (re-)configuration of the control system as detailed in Chapter 3. Our hypothesis is that we can synthesize and plug specific controllers automatically, so as to achieve better performance against certain pre-defined QoS criteria.

4.1 Case with one actuator per zone

Heating control in buildings, e.g. in HVAC systems, is currently performed by applying simple strategies, e.g. using a thermostat to activate/inactivate the valve of a variable refrigerator flow unit (VRF) at few pre-defined levels or apply proportional flow-rate through valve/fan control [48]. Literature also presents model-based control methods for achieving better control performance and energy savings in HVAC

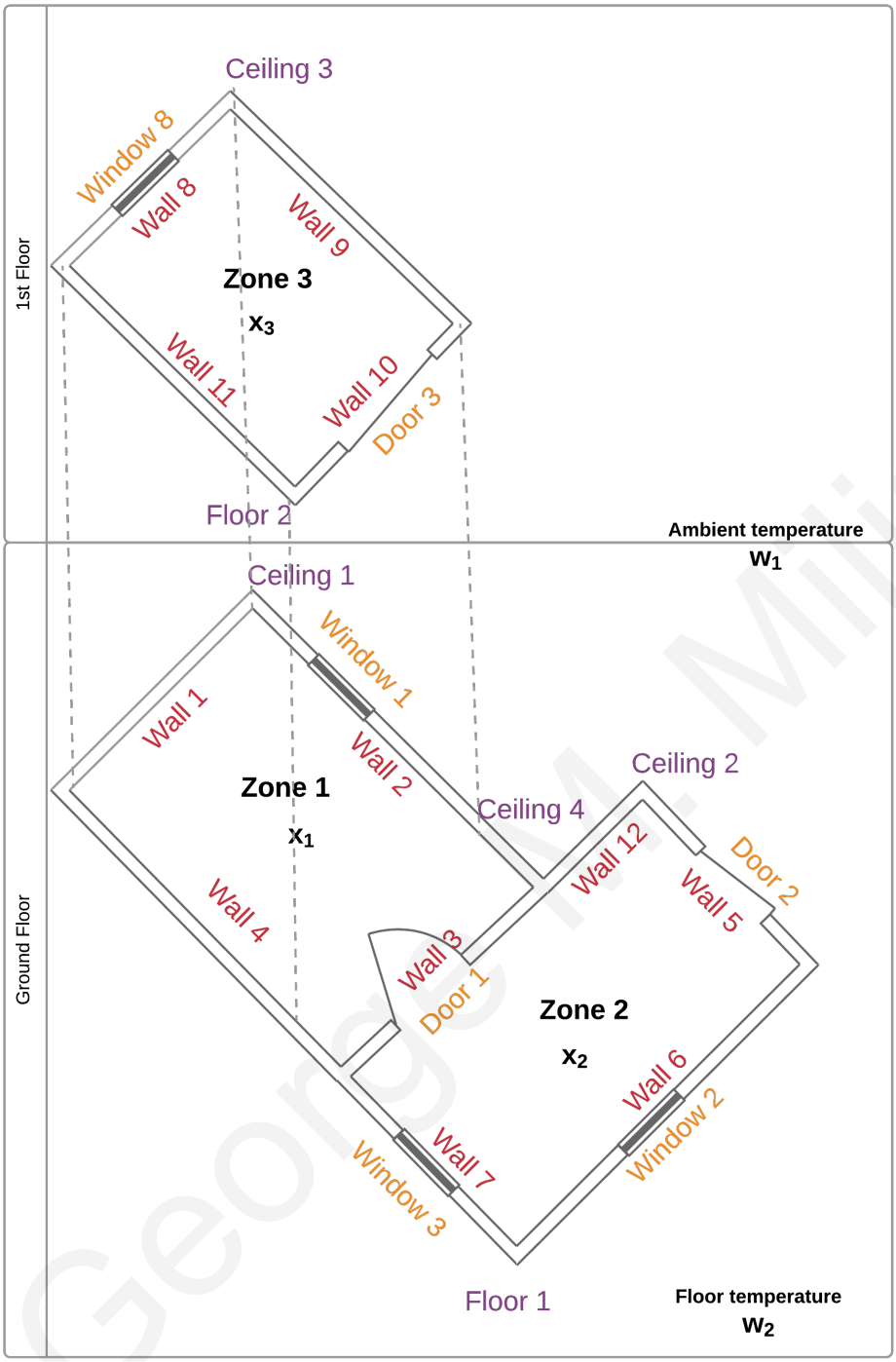


Figure 4.1: A building with three zones spanning two floors. It involves 12 wall-areas with 3 doors and 4 windows, 4 ceiling-areas and 2 floor-areas. The zones have temperatures x_i, $i \in \{1, 2, 3\}$. The ambient temperature is modelled by w_1 and the floor temperature by w_2.

systems [123]. However, the latter do not typically find their way to real applications, since the dependence on plant models reduces significantly the portability of the solutions and subsequently the potential return on the investment. In our work we

utilise the SEMIoTICS reference architecture and system to allow the online passing of building topology and heat characteristics' parameters to model-based controllers, thus achieving their online synthesis and subsequently the plug-and-play control of heating devices. The solution is portable since the operation of the system does not depend on the prior knowledge of the building or the heating devices' parameters, which are considered given as input, through the "Knowledge Graph" $\mathcal{G}_I$.

More specifically, we design controllers adhering to the syntactical form presented in Section 2.2. E.g., a controller being part of the configuration selected at time k_I, is modelled by $u_I(k) = f_c(y'_I(k); \zeta^c_I(k))$, where $u_I(k)$ is the vector of control decision signals, $y'_I(k)$ is a vector of (potentially processed) measurement signals and $\zeta^c_I(k)$ is a vector of parameter values. These vectors have appropriate dimensions and content, which depend on the order of the system and the specific implementation of the controller operating in the time interval $[k_I, k_{I+1})$. Specifically, the vector $\zeta^c_I(k)$ corresponds to internal parameters of the controller implementation that can be passed to the controller online, as will be explained later. The end-points producing/consuming these signals are subject to the Semantic Matching and Semantic Reasoning operations detailed in Sections 3.3.2 and 3.3.4 respectively. Therefore, if the controller is selected in a configuration, its end-points' signals will be routed accordingly. The details of the low-level communication architectures and routing protocols are not in the scope of this Dissertation.

We start by defining the plant subject to control (the pointer I is omitted from the notation, since the same controllers' synthesis process is executed once following each k_I). The heat flow equations of the zones in Fig. 4.1 are derived using the Fourier's law and the conservation of energy [17], considering that the heat energy flow-rate per surface unit area is proportional to the negative temperature gradient across the surface. Assuming heat flowing through a surface in one dimension, the heat equation of zone i, with $i \in \{1, 2, 3\}$, in a discrete time formulation, is given by (4.1):

$$\begin{aligned} x_i(k+1) =& x_i(k) + a_i(\zeta^p(k))x_i(k) + h_i^T(\zeta^p(k))x_{(i,adj)}(k) + d_i^T(\zeta^p(k))w(k) \\ &+ g_i(x_i(k), k; \zeta^c(k))u_i(k) + \bar{\gamma}_i(\zeta^p(k)) \end{aligned} \tag{4.1}$$

where $x_i(k+1)$ is the homogeneous temperature value in zone i at time $k+1$, the scalar function $a_i(\cdot)$ depends on a heat parameters' vector $\zeta^p(k)$ (heat characteristics

and areas of building surface structures and openings at time k, as detailed in the sequel), $x_{(i,adj)}(k)$ is a vector of appropriate dimensions representing the temperatures of adjacent zones to zone i, $h_i(\zeta^p(k))$ is the associated vector of scalar functions that depend on the state interdependency heat parameters, $w(k)$ and $d_i(\zeta^p(k))$ are vectors of appropriate dimensions representing the ambient and floor temperatures and the dynamics associated with them, respectively. The model accounts also for a heat energy input $u_i(k)$ affecting the state through the function $g_i(x_i(k), k; \zeta^c(k))$, where $\zeta^c \subseteq \zeta^p$ is the part of the parameters associated with the controlled input, as well as some bounded additional uncertainties (e.g., due to model inaccuracies), grouped here in the term $\bar{\gamma}_i(\zeta^p(k))$.

In this use case, the vector ζ^c represents the zone desired temperature, the uncontrolled ambient and floor temperatures, the heating device characteristics (maximum flow-rate; specific heat capacity of present liquid/gas; the inflow air temperature; etc.), as well as the plant characteristics (volume of subject zone; specific heat capacity and mass density of air in the zone; the area, thickness and thermal conductivity of the material of each wall and opening; the doors/windows opening percentages). The following operations are executed automatically through SEMIoTICS. The "Knowledge Graph" is able to model the building topology in detail, and thus enable the semantic annotation of the parameters, by linking to the DogOnt ontology [15]. The values for the above parameters are assumed extracted from a "Building Information Model" [23,95]. This facilitates SEMIoTICS, following change events at times k_l, to synthesize online the temperature change model of a building by retrieving the parameters ζ^c from the "Knowledge Graph" state $\mathcal{G}_l$. The synthesis is performed by the "Controller Synthesis" module of the Supervisor Σ, represented by the blue-highlighted box in the extended SEMIoTICS architecture in Fig. 4.2. The operation of this module is triggered by the "Semantic Reasoning" process before the triggering of the "Configuration Selection" process.

With certain modifications, the previously described plant model is brought to the general form in 4.2.

$$x_i(k+1) - x_i(k) = a_i(\zeta^p(k))x_i(k) + \xi_i(x_i(k), w(k); \zeta^p(k)) + g_i(x_i(k); \zeta^c(k))u_i(k) \tag{4.2}$$

The model is feedback linearizable, since, $g_i(\cdot)$ never becomes zero for the plant we consider in our work and the resulted linear model is controllable [88]. Therefore,

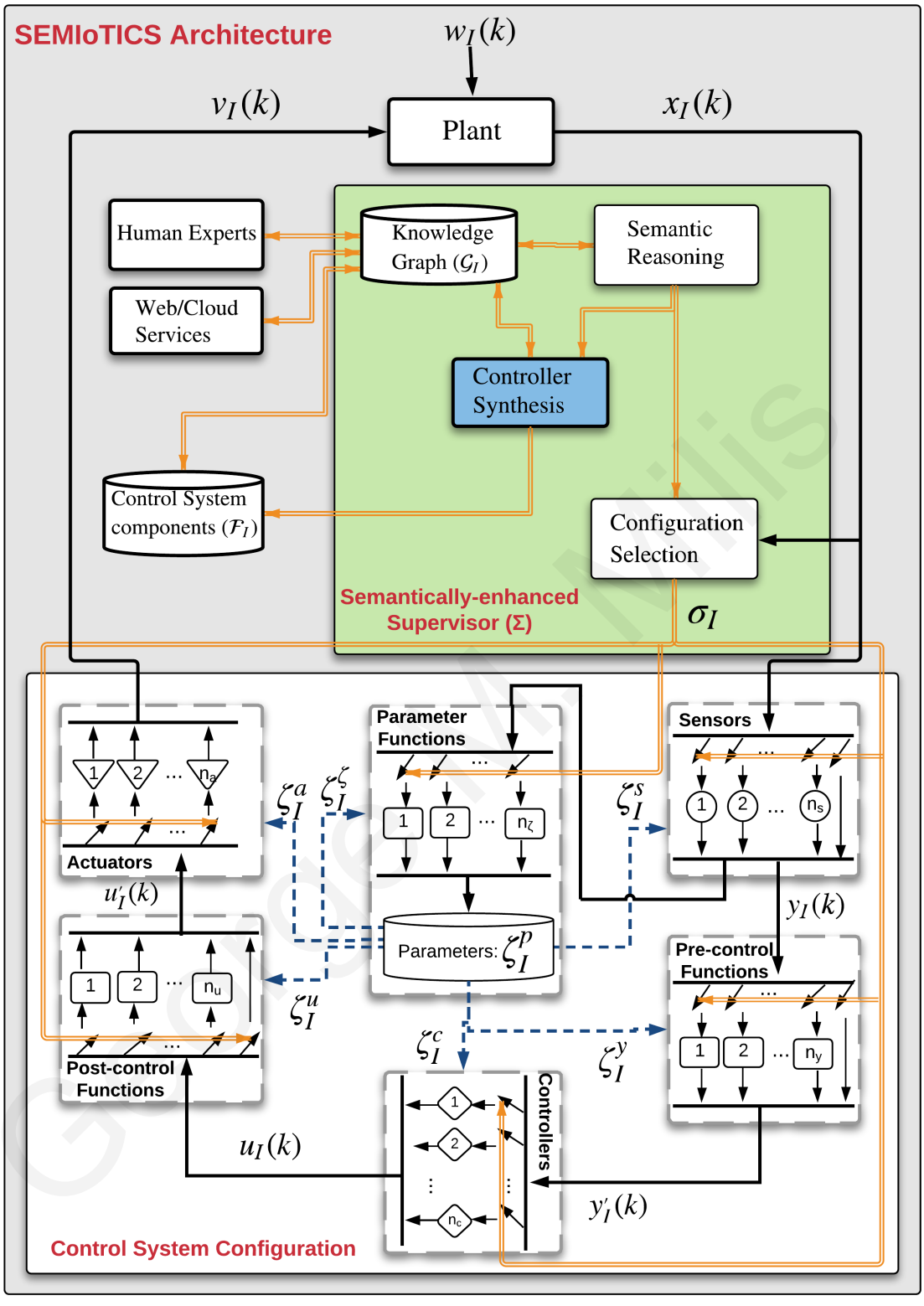


Figure 4.2: Extended SEMIoTICS architecture, incorporating the "Controller Synthesis" module

we use a feedback linearization controller and choose a signal $u_i(k)$ of the form in (4.3), to regulate the state to follow a reference signal $r_i(k)$.

$$u_i(k) = g_i^{-1}(y_i'(k); \zeta^c(k))[-\xi_i(y_i'(k), w(k); \zeta^c(k)) + v_i(k)] \quad (4.3)$$

where $y_i'(k)$ is a (potentially processed) measurement of the state $x_i(k)$ at time step k, $\zeta^c(k) \subseteq \zeta^p(k)$ is the aforementioned vector of controller-defined parameters and $v_i = -K_i^c(y_i'(k) - r_i(k))$ is a proportional linear control law. The control gain K_i^c is computed so as to stabilise the linearised dynamics of the zone, by placing the poles of the closed-loop system on the negative part of the horizontal axis in the s-plane in a way to achieve a high to moderate error convergence rate and smooth enough response.

For each building zone that has at least one heating device installed, the "Controller Synthesis" module synthesizes a controller as in (4.3). The semantic annotation of each controller is created online and stored in $\mathcal{G}_I$. This involves the annotations of:

i) the vector y_i', with one element representing the respective zone temperature, as well as additional elements representing the temperature measurements of adjacent zones, where applicable;

ii) the scalar control decision u_i, representing the heating device flow-rate;

iii) the vector of parameters ζ^c, with multiple elements to cover the full list of parameters defined earlier.

Where the above controllers are selected by SEMIoTICS following an event at k_I, the state measurement and parameter vectors are generated and passed to them online, taking advantage of the SEMIoTICS semantic annotation, semantic matching and semantic reasoning processes. It is noted that SEMIoTICS makes no pre-assumption on the existence of actuators (i.e., heating devices) in any of the zones. Therefore, the "Components Synthesis" module synthesizes a distributed control scheme, i.e., a separate controller function for each zone where there is at least one semantically matching actuator installed. The full state measurements, as well as the knowledge about disturbances and plant parameters are shared among the zone controllers.

4.2 Case with multiple actuators per zone: optimal allocation of control signal

The control signal in each zone is utilised to drive a semantically matching heating device, considering the outcome of the "Configuration Selection" process. In an IoT framework, however, additional heating devices may become available in a zone. We show here how we modify the aforementioned controllers to allow allocation of the control signal to multiple actuators in a cost-effective way. The solution considers variable liquid flow units that heat certain volume of liquid and allow the radiation of heat into a zone.

Accounting for multiple actuators in a zone, changes the plant model to the one in (4.4).

$$\begin{aligned} x_i(k+1) =& x_i(k) + a_i(\zeta^p(k))x_i(k) + \xi_i(x_i(k), w(k); \zeta^p(k)) \\ &+ \sum_{j=1}^{n_i^a} g_{ij}(x_{ij}^h(k), x_i(k), u_{ij}(k); \zeta^c(k)) \end{aligned} \tag{4.4}$$

where the heat energy input dynamics are produced by n_i^a actuators installed in zone i, where $i \in \{1, 2, 3\}$ and are given by the sum of all individual input dynamics affecting the state. The pointer $j = 1, 2, ..., n_i^a$ serves each time as iterator of the heating devices in zone i. Using the heat energy flow parameters, we can write this sum as in (4.5):

$$u_i'(k) = \frac{1}{H_{air}} \sum_{j=1}^{n_i^a} C_{ij}^h A_{ij}^h (x_{ij}^h(k) - x_i(k)) \tag{4.5}$$

where H_{air} is the heat capacity of the air computed using the zone air volume, mass density and specific heat capacity, C_{ij}^h is the heat transfer coefficient of the surface of the heating device j in zone i measured in $W/m^{2o}C$, A_{ij}^h is the surface area of the heating device j in m^2, $x_{ij}^h(k)$ is the temperature of the liquid in the same heating device in ^oC and $x_i(k)$ is the zone i temperature state at time k. The heating device liquid temperature is affected by the device's internal dynamics and is controlled by the flow of hot liquid through a valve, as shown in (4.6), where H_{ij}^l is the heat capacity of the liquid in the heating device j of zone i, $\bar{u}_{ij}$ is the maximum valve opening and subsequently the maximum liquid flow-rate that can be achieved by

that heating device, C_{ij}^{l} is the specific heat capacity of the liquid of device j, T_{ij}^{h} is the constant temperature of the inflow liquid of the heating device j in zone i.

$$x_{ij}^{h}(k) = x_{ij}^{h}(k-1) - \frac{1}{H_{ij}^{l}} C_{ij}^{h} A_{ij}^{h} (x_{ij}^{h}(k-1) - x_i(k-1)) + \frac{1}{H_{ij}^{l}} \bar{u}_{ij} u_{ij}(k) C_{ij}^{l} (T_{ij}^{h} - x_{ij}^{h}(k-1)) \tag{4.6}$$

Substituting (4.6) into (4.5) and then into (4.4), we end up with a model of the form in (4.7).

$$x_i(k+1) = x_i(k) + a_i(\zeta^p(k)) x_i(k) + \xi_i(x_i(k), w(k); \zeta^p(k)) + u_i'(k) \tag{4.7}$$

We again design a feedback linearization controller, this time choosing the signal $u_i'(k)$ of the form in (4.8).

$$u_i'(y_i'(k); \zeta^c(k)) = -\xi_i(y_i'(k), w(k); \zeta^c(k)) + \nu_i \tag{4.8}$$

where again the linear control law is given by $\nu_i = -K_i^c(y_i'(k) - r_i(k))$. The overall control input dynamics cancel all other non-linearities. What remains is to compute individual control signals $u_{ij}(k)$ with $j = 1, ..., n_i^a$, so as to sum up to the total required control input $u_i'(k)$ for the zone i.

We formulate the problem as an "Economic Dispatch" process, which is usually used in the control centres of the Electric Power Systems (section 12.4 of [46]). It concerns the dispatch of the power load demand to available generator units in an area, in the most cost-effective way. The semantic annotation of each heating device in the "Knowledge Graph" $\mathcal{G}$, defines its operation-cost parameters, which help to compute the function: $c_{ij}(k) = (\alpha_{ij} + \beta_{ij} u_{ij}(k) + \gamma_{ij} {u_{ij}}^2(k)) c_f$. This is a quadratic function mapping the output heat power of a heating device at time k to the cost of producing it; c_f models the cost of electricity, fuel cost, energy losses, etc. The formulation of the optimization problem is given in (4.2). The objective is to minimize the total cost of operating the devices. The cost is measured in a unit of currency per time of operation.

$$\underset{u_{ij}(k)}{\operatorname{argmin}} \sum_{j=1}^{n_i^a} (\alpha_{ij} + \beta_{ij} u_{ij}(k) + \gamma_{ij} u_{ij}^2(k)) c_f$$

s.t:

$$\sum_{j=1}^{n_i^a} \frac{C_{ij}^h A_{ij}^h \bar{u}_{ij} C_{ij}^l (T_{ij}^h(k) - x_{ij}^h(k)) u_{ij}(k)}{H_{air} H_{ij}^l} = u_i'(k),$$

$$0 \leq u_{ij}(k) \leq \bar{u}_{ij}, \forall j$$

Therefore, at times $k \in \mathcal{K}$, the designed controller for each zone, computes the total control input dynamics $u_i'(k)$, which considers the dynamics of all available heating devices, and solves online the optimisation problem of finding the part of the control signal to be allocated to each of the available n_i^a units in the zone, such as for the overall cost to be minimised. It is noted that the minimum is achieved when all heating units operate at equal incremental operating cost [46], which is found by adding the energy preservation constraint on the overall cost function with a Lagrange multiplier and applying the partial derivative-based minimization criteria. We enforce also inequality constraints, limiting the minimum and maximum control signal that each unit can accommodate. The result is that the system exploits the cost of the installed heating devices in a zone in order to allocate the control signal. It is emphasized that the solution makes no pre-assumption about the number of heating devices. Moreover, the operation parameters of the heating devices, which are required to compute their flow-rate, are extracted online from the respective semantic annotations of the devices.

4.3 Simulation Results and Impact

We run three heating control simulations for the building in Fig. 4.1, with the feedback control configurations managed online by an instance of SEMIoTICS (Fig. 4.2). The values for the parameters discussed in Section 4.1 are taken from the literature and are listed in Table 4.1 for convenience, together with several other simulation-specific parameters. The three simulations are identical as far as the content of this table is concerned. Moreover, the following scenario is implemented in all three simulations:

Table 4.1: List of simulation parameters and (indicative) values

Parameter name	**Value**
Model of ambient temperature	Erbs formula [33]
Sensor recording (sec)	1
Sensor sampling for control (sec)	60
Reference zone temperatures (^oC)	Zone 1: 25 Zone 2: 23 Zone 3: 27
Floor temperature (^oC)	15
Maximum liquid flow rates of all heating devices (kg/s)	0.025
Zones volumes (m^3)	Zone 1: 6 x 4 x 2.80 Zone 2: 6 x 4 x 2.80 Zone 3: 4 x 4.5 x 2.80
Specific heat capacity of air on constant pressure (J/kg^oC)	1004
Specific heat capacity of air on constant volume (J/kg^oC)	717
Density of air at 25^oC, at sea level (kg/m^3)	1.2250
Specific heat capacity of water (J/kg^oC)	4184
Density of water (kg/m^3)	998
Thermal conductivity of walls' material (W/m^oC)	0.15
Thermal conductivity of doors' material; wood oak (W/m^oC)	0.09
Thermal conductivity of windows' material; glass (W/m^oC)	0.096

Thickness of doors (m)	0.10
Thickness of windows (m)	0.05
Thickness of walls (m)	0.30
Area of openings (m^2)	doors: 2 windows: 1.2
Heating devices' inflow liquid temperature (^oC)	40
Electricity kWatt-hour cost, normal tariff (EUR)	0.157
Electricity kWatt-hour cost, low tariff (EUR)	0.07
Heat oil cost (EUR)	0.09
Volume of liquid in coils of main heating devices (m^3)	0.02
Volume of liquid in coils of secondary heating devices (m^3)	0.002
Maximum liquid flow rate of heating devices (kg/s)	0.025
Heat transfer coefficients of main heating devices ($W/m^{2o}C$)	12
Heat transfer coefficients of secondary heating devices ($W/m^{2o}C$)	150

Initially, each of the three zones is equipped with one IoT-enabled temperature sensor and one heating device. The "Components Synthesis" module synthesizes three controllers online, as presented in Section 4.1, one for each of the three zones. Each zone is controlled separately, in a distributed setting. Three occupancy sensors are installed, one in each zone, detecting whether there is any presence in the zone or not. In addition, three "Parameter Functions" are given, one for each zone, which reduce the reference value of the zone temperature from its current value to 18^oC if the zone is not occupied. Moreover, sensors are deployed on all doors and windows, measuring their opening percentage. Respective "Parameter Functions" are also given, which update the values of the opening parameters in the "Parameters Registry" $\mathcal{Z}$. At certain non-periodical and non-predictable time intervals, zone 1

becomes unoccupied (as detected by the installed occupancy sensors). The same happens in zone 3. Also, "window 1" presents a 2% opening during another time interval and this is taken into consideration by SEMIoTICS accordingly.

The three simulations differ in terms of utilization of certain control system components during each configuration:

Simulation 1: The three zones are equipped with IoT-enabled heating devices with heat and cost parameters as defined for the "main heating devices" in Table 4.1. These devices operate with heat-oil fuel.

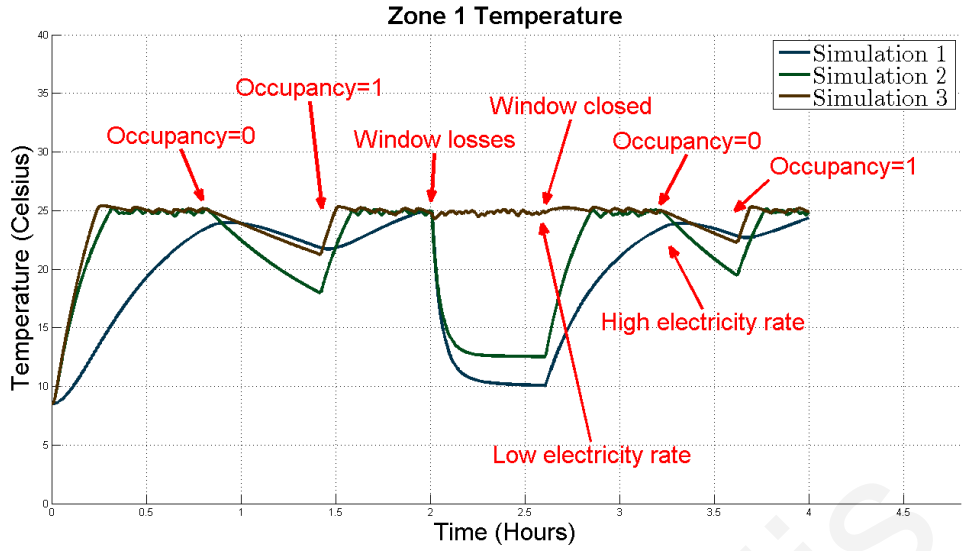
Simulation 2: The three zones are equipped with IoT-enabled heating devices with heat and cost parameters as defined for the "secondary heating devices" in Table 4.1. These devices operate on electricity.

Simulation 3: Zone 1 and zone 3 are equipped with both the aforementioned main and secondary IoT-enabled heating devices each. No change in zone 2.

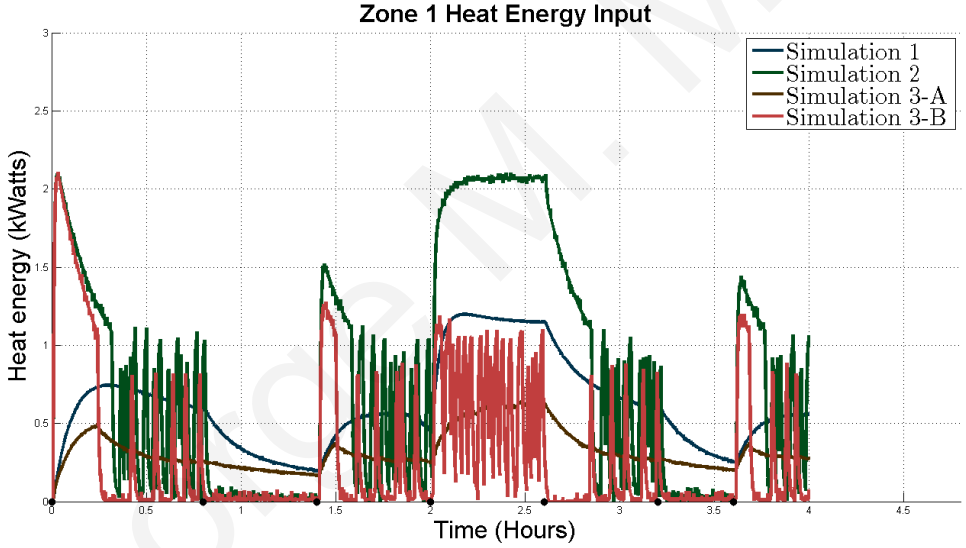
Fig. 4.3a shows the simulated temperature dynamics of zone 1 for each of the three simulation scenarios described above, whereas Fig. 4.3b shows the heat energy input to the plant as produced by the heating devices described earlier. To facilitate visualisation, we show 4-hour executions of the simulations, presenting the results only for zone 1. The figures help obtain a visual idea of how the feedback control behaves through each of the three simulations.

In order to further study the impact of the solution, we model the building's overall performance, per each zone $i \in \{1, 2, 3\}$. We adopt the QoS criteria introduced in Section 3.3.7 to model the building's thermal QoS per each zone. It is noted that the evaluation of the performance is performed outside of the online decision making process and aims at measuring the impact of SEMIoTICS operation in retrospect, within certain operation intervals. Table 4.2 presents the results of the three executions against the occupants' thermal comfort and the energy efficiency criteria defined in Section 3.3.7, this time from the full 24-hour executions of the simulations.

We emphasize that the focus of our work is not on the design of new types of controllers. We demonstrate that SEMIoTICS offers effective automatic synthesis features for the control applications in large-scale buildings, which enable the online plugging of components that improve certain characteristics of the overall system. Specifically, our results show:



(a) The temperature of zone 1 in a 4-hour simulation of SEMIoTICS, for each of the three described simulation executions



(b) The heat energy entering zone 1 in a 4-hour simulation of SEMIoTICS, for each of the three described simulation executions

Figure 4.3: A 4-hour execution of the three simulations, presenting the results only for zone 1

Table 4.2: Simulation results

Characteristic	**Simulation** 1	**Simulation** 2	**Simulation** 3
Occupants' Thermal Comfort	0.926	0.974	0.978
Energy Efficiency	0.21	0.0	0.08

1. The ability to synthesize, semantically annotate and plug model-based "Controllers" online, which make use of existing sensing and actuation capabilities and a vector of parameters that is also passed online; the parameters can be updated online through respective "Parameter Functions" that are also pluggable online.

2. The ability to equip the controllers with mechanisms to retrieve information about the characteristics of actuators online and run optimization processes to allocate the control signals in a cost-effective way. It can be seen that the use of the optimization mechanism, taking advantage of multiple actuators online ("Simulation 3"), managed to retain the occupant's thermal comfort at the levels achieved by the heating device with the fast dynamics ("Simulation 2"), considerably improving at the same time the energy efficiency of the solution by about 8% comparing to the case where the expensive devices were used ("Simulation 1"). The energy consumption of the expensive devices was used as a normalization factor, thus shown as zero energy efficiency for comparison purposes.

It is noted that the actual improvements highly depend on the events to which SEMIoTICS will be called to respond during execution, however, we show how the solution exploits the available components so as to find an optimal involving two QoS criteria; the thermal comfort and the energy efficiency. It is also noted again that the controllers synthesized and plugged online, do not make any assumptions regarding the pre-existence of particular devices; instead, they semantically describe their end-points, so that their participation in feedback control system configurations is achieved if/when their connection is semantically validated through SEMIoTICS.

Chapter 5

SEMIoTICS applications in Critical Infrastructures

5.1 Semantically-enhanced Reconfigurability in State Estimation Structures of Power Systems

5.1.1 Introduction

The Electric Power System (EPS) comprises a very critical infrastructure for the operation of our modern society and economy. Therefore, the effective monitoring and control of an EPS is considered of utmost importance and is undertaken by a set of components that perform a wide range of functionalities and together comprise the "Energy Control Centre" (ECC) application. Considerable resources are allocated by international organizations and governments, as well as private organisations, to advance the state-of-art and subsequently the effectiveness of all components of the ECC.

One of the most critical components of an ECC, is the "State Estimator" (SE), which serves several functions that support the reporting to the human operators, as well as functions that support the operation planning, the stability and the security of the EPS. The key position of the SE is illustrated in Fig. 5.1 [124]. In essence, the SE produces an estimate of the operating state of the system (i.e., voltage magnitude and phase angle of each bus) in consecutive time intervals, by processing redundant measurements acquired by selected substations of the system. The usual measurements are the real and reactive power injections, real and reactive power flows of the

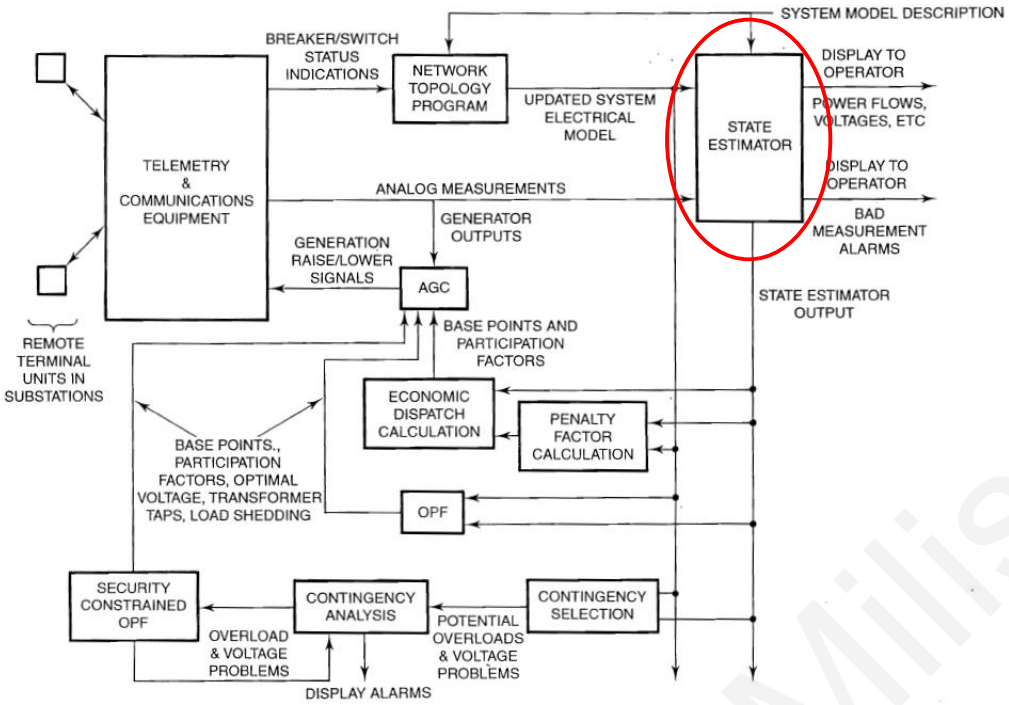


Figure 5.1: The typical architecture of an ECC, where the key position of the SE is highlighted. [Source: [124]]

transmission lines, as well as the voltage magnitudes at the system buses. A strict prerequisite for obtaining a unique solution by the SE is to have a fully observable power system through the utilised measurements.

The reliance of many ECC components on the state estimation imposes that the SE must provide as accurate and reliable results as possible. With the recent advancements in the measurement technology of EPS and the observed progress in the actual deployment of Phasor Measurement Units (PMUs) in the measurement layer of EPS, the research community has been investigating ways to take advantage of the available synchronized phasor measurements for improving the performance of the SE. Although the cost of PMUs has been decreased and is foreseen to further decrease in the near future, at the moment the measuring and communication infrastructure required for the deployment of PMUs turn the adoption of SE that rely solely on the PMU measurements impractical. This is further emphasized by the fact that conventional measurements can be useful in many other monitoring and control functions, such as measurement calibration and bad data detection methods [3]. Therefore, the research has been focusing on hybrid architectures for the SE that utilise both conventional and synchronized measurements for estimating the

operating state of the EPS. A potential problem in the hybrid architectures is the inclusion of the electric-current phasor measurements (PMU measurements) in the measurement vector that is usually detrimental to the performance of SE [3]. Many alternative techniques have been proposed for overcoming this issue, by utilising the concept of "pseudo measurements" [7,12,19,20,81], converting the electric current measurement to e.g., active power flow through an appropriate transformation function.

5.1.2 The need for flexible architectures

The common denominator in all hybrid state estimation techniques is that they require the use of a different SE implementation to be able to consume both types of measurements under synchronisation conditions. In current practice, the design of SE architectures is based on a fixed configuration of specific sub-components, with static measurement devices' configuration and predetermined state estimation routines. That is, the SE algorithm developer needs to design the algorithm based on pre-acquired knowledge about the available devices and their specifications or the technicians that install the devices need to know the specifications of the supervisory control and data acquisition (SCADA) system (mainly the SE implementation) in advance. Moreover, the EPS operators need to decide in advance whether they will perform the state estimation based only on conventional measurements or whether they will adopt any of the hybrid architectures discussed above, together with the corresponding (static) implementation of the SE. These cases reveal the inflexibility of the existing SE architectures. Although this does not create significant inconvenience in today's operation procedures, since the need for changes in the components (i.e., sensing devices and SE implementations) is not so frequent, it is expected that the flexibility may become more important in the near future and as the EPS adopt the IoT paradigm.

The today's Smart Grid advancements [62] with the improved integration of the Electric Power Grid with the ICT infrastructure, reveal the need and open up additional opportunities to offer flexibility in the SE architectures. A "smart" implementation would be expected to present online adaptability to changes in the composition of the measurement vector (e.g., introduction of new conventional and/or PMU devices, removal or moving of PMU devices to other positions in the network

topology, availability of new SE implementations) and perform any necessary online reconfigurations, in order to avoid the need of manual replacement of components and subsequent downtime. Therefore, our work contributes with the introduction of an SE architecture that can be used in several ECC applications and allows online (re)configurability. The proposed SE architecture is enriched with the semantically-enhanced Supervisor Σ of SEMIoTICS, which stores structured knowledge about the available components (e.g., measurement devices and SE algorithms' implementations), performs semantic reasoning after any change in the available components and takes decisions about the online reconfiguration of the SE. The situation awareness is achieved by utilising an adapted version of the SEMIoTICS "Knowledge Graph" $\mathcal{G}$ that fully describes the types of the components, their characteristics, their locations, as well as the physical properties they measure.

5.1.3 Formulation

The state estimation process can be illustrated with the block diagram of Fig. 5.2, where the vector of the states of the EPS (that is, the voltage magnitude and angle at all buses) is defined as $x \in \mathcal{R}^{n_x}$, the measurements' vector produced by the installed set of measurement devices is given by $y \in \mathcal{R}^{n_y}$, while $\hat{x} \in \mathcal{R}^{n_x}$ denotes the vector of the estimated system states. The diagram shows that the SE uses the set of available measurements as input and produces an estimate of the system states which is then utilised by several other monitoring and control functions that act on the system and support its operation.

The estimation of the EPS state is performed based on different approaches, with the one most commonly used being the WLS (Weighted Least Squares). According to the WLS method, the state vector x of the system is determined iteratively by minimizing the weighted residuals between the estimated and the actual acquired measurements, $J(x) = [y - h(x)]W[y - h(x)]$, where $h(x)$ is the function-vector associating the state variables to the measurements, and W is the inverse of the measurement error covariance matrix. Details on these state estimation algorithms can be found in [3]. It is assumed that the network topology and parameters are known prior to the state estimation and also the EPS is completely observable by the measurements contained in vector y. In general, the measurement vector of an SE is given as $y = [P_{flow}, P_{inj}, Q_{flow}, Q_{inj}, |V|, \theta_V, |I|, \theta_I]^T$, where the elements represent respectively

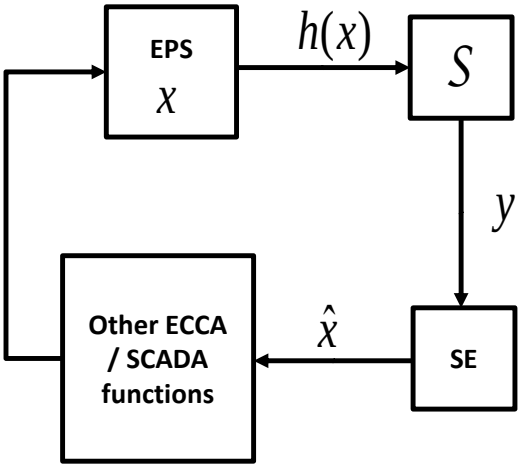


Figure 5.2: Block diagram of the SE implementation, focusing on the relation with the plant, the measurement signals and the rest of the monitoring and control applications.

vectors of active and reactive power flows in lines, active and reactive power injected on buses, as well as voltage and electric current magnitudes and angles. In case of no availability of PMU units, the phasor measurements are not present and the measurement vector is adjusted accordingly.

It is emphasised here that, on one hand it has been proved beneficial for the SE to use a hybrid structure for exploiting also the PMU measurements [91], but on the other hand the PMU devices, like any measurement device, are subject to failures (e.g., broken GPS communication links) which would turn the hybrid SE not applicable and subsequently would necessitate the return to a conventional SE architecture. Furthermore, the today's widely adopted practice by Utilities, is to operate a SE that considers the locations of measurement devices pre-defined. However, the recent advancements in technology justify a demand for additional flexibility in the SE architecture, which would turn it possible to adapt online to changes happening in the content of the measurement vector, as well as to changes in the available SE algorithms' implementations.

More specifically, assuming a conventional SE, operating on a vector of conventional measurements (no PMU measurements), it is desirable for the SE architecture to be able to adapt online when PMUs are installed in specific locations of the EPS topology. It is also desirable for the SE to continue operating when these PMUs are mobile and can therefore move from one location to another, which will subse-

quently change the mapping to the system states. Finally, the SE should continue operating when PMUs or conventional devices are removed from the network due to any reason (assuming still an observable system). The proposed architecture and solution is presented in the sequel.

5.1.4 Proposed Solution

The proposed architecture for the ECC is depicted in Fig. 5.3, which comprises a customised version of the SEMIoTICS architecture detailed in Section 3. The index $I \in \{0, 1, 2, ...\}$ is the index of the SE architecture configurations that are selected following a change in the components' availability. In configuration I, the operation of the EPS is monitored by a set of sensors $\mathcal{F}_I^s$, e.g., conventional sensors and PMUs. As mentioned earlier, in hybrid SE implementations, specific types of measurements (e.g., measurements of electric-current which present differentiation challenges through their mathematical relation to the voltage states of the system) may degrade the SE performance. In such cases, the measurements pass through a set of appropriate transformation functions $\mathcal{F}_I^y$ before given to the selected SE function from the set $\mathcal{F}_I^{se}$. For instance, the electric current of a line is converted to an active power value, resulting to a "pseudo-measurement".

In order to achieve the objective of shifting from one configuration to another when required, an instance of the semantically-enhanced Supervisor Σ is utilised, which is responsible for making the decisions and orchestrating the components. The Supervisor Σ first detects and identifies any new components added. The physical communication among components is facilitated through an assumed existing communication protocol, e.g., with extensions to the currently adopted SCADA systems (the details of this fall outside the scope of our work). Subsequently, Σ becomes aware of the characteristics and capabilities of the new components, through their semantic annotations (Section 3.3.2). Once the semantic annotations are received by Σ, they are stored in the "Knowledge Graph" $\mathcal{G}$, thus facilitating awareness of their functionality and capabilities. Subsequently, an implementation of the "Semantic Reasoning" process is utilized to reconfigure the existing SE architecture considering all available components, e.g., to return to conventional SE if PMU measurements are lost. It is emphasised that any switching to a new configuration happens strictly in the time between subsequent estimation cycles and no interruption of a running

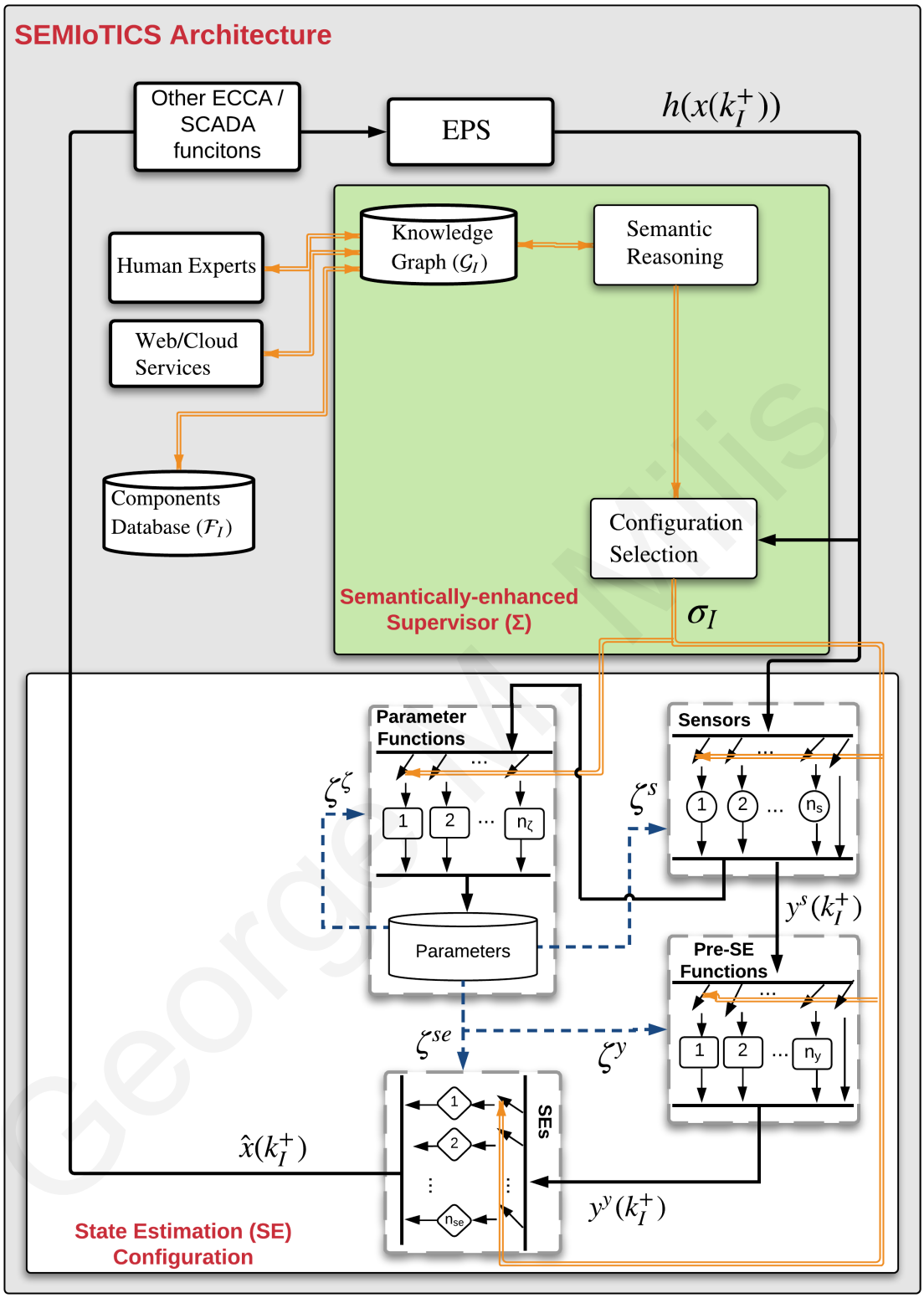


Figure 5.3: The SEMIoTICS architecture customised for the proposed SE implementation; The Supervisor Σ undertakes the maintenance of the Knowledge Graph, the semantic reasoning and the configuration selection process. Sensors, State Estimators and Pre-SE Functions are part of the online configuration selection.

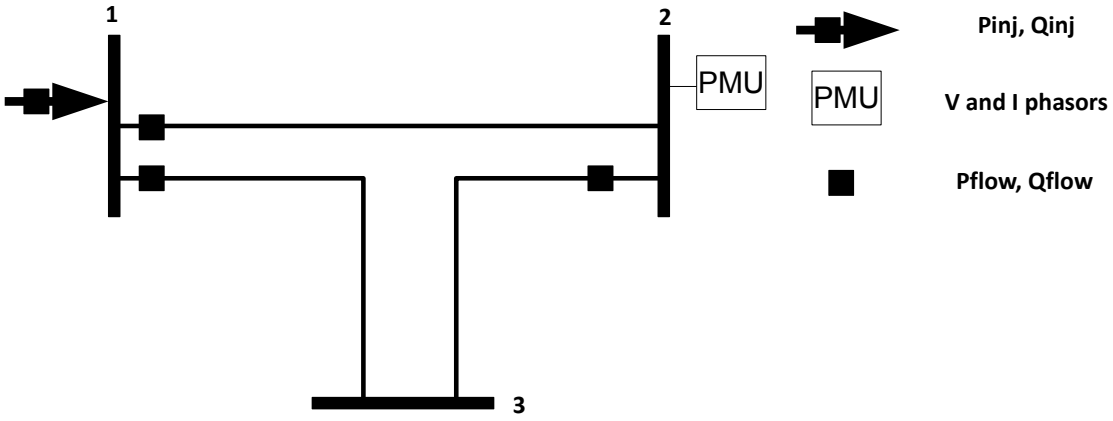


Figure 5.4: Three-bus EPS with five measurement devices (12 single measurements)

estimation execution is performed. It is assumed that the time between two subsequent estimation executions is enough to allow /*Sigma* to complete the reasoning and decide on the new configuration. In case of multiple valid configurations, the "Configuration Selection" process selects one of them taking into account pre-defined criteria as explained in the sequel.

The key advantage of the proposed architecture is the inherited flexibility and automation of the components' wiring layer, through the incorporation of the Supervisor Σ. The structured knowledge representation, enables a machine to undertake tasks that would otherwise be undertaken by humans (e.g., the update of the SE implementation when the measurement vector changes). The same approach can be adopted in other parts of the ECC application, where such flexibility may be of use. The following section gives details about the processes executed by Σ.

5.1.5 Knowledge Graph and Semantic Reasoning Process

The implementation of the decision mechanism for the online configuration of the SE is clarified through an illustrative case study.

Let assume a three-bus EPS as in Fig. 5.4, where five measurement devices are installed, performing twelve sensing tasks in total. As shown, the devices measure the injected power on bus 1, the flow of power on lines $1-2$, $1-3$ and $2-3$, as well as the voltage and electric-current phasors on bus 2. The voltage magnitude is measured in kV, the electric-current magnitude in kA, the angles in degrees and the power in kW (active) and $kVAr$ (reactive).

The "things" contained in the introduced EPS are:

- Twelve sensors ($f_i^s \in \mathcal{F}_s,\ i = 1, ..., 12$)
- The buses and transmission lines as features-of-interest (system locations): $\{l_j |\ j = 1, ..., 6\}$, e.g. 'bus1'
- Physical properties: $\{q_1$: active electric power, q_2 : reactive electric power, q_3 : voltage magnitude, q_4 : current magnitude, q_5 : voltage angle, q_6 : current angle$\}$
- Components' capabilities: $\{p_1$: state, p_2 : stateMeasurement, p_3 : processedStateMeasurement, p_4 : stateEstimation$\}$
- Measurement units: $\{m_1$: kW, m_2 : kVArs, m_3 : kV, m_4 : kA, m_5 : degrees, m_6 : pu$\}$

All above "things" are included in the "Knowledge Graph" $\mathcal{G}$ using the mechanisms discussed in 3.3.1. Let assume now a SE architecture as in Fig. 5.3, deployed in the control centre of the introduced EPS, such as to estimate the voltage magnitudes and angles of the three buses. The mapping of the measurement vector to the system states is given by (5.1),

$$y^y(k_I^+) = f^y(y^s(k_I^+)) = f^y(h_I(x(k_I^+)) + w^s(k_I^+)) \tag{5.1}$$

where $x(k_I^+) \in \mathcal{R}^5$ is the vector of voltage magnitudes and angles of the three buses, in per-unit (pu), following the event $I = 0$ at time $k > k_0$; $y^s(k_I^+) \in \mathcal{R}^{12}$ is the measurements' vector produced by the deployed measurement devices; $h_I(\cdot)$ are the known mappings of system states to measurements for the time following k_0; $w^s(k_I^+)$ is the noise associated with the measurements; $f^y(.)$ is a function-vector representing the processing of measurement signals that is applied (if and when required), to ensure compatibility with what the SE expects to receive.

The objective is for the Supervisor Σ to choose a subset of components from the available ones after a change event, based on the matching of their semantic annotations, so as to ensure the SE operates properly. The "Semantic Reasoning" process considers the types of the components and their expected role in the SE implementation. The position of each type of component is fixed, with sensors always positioned to measure EPS outputs and then passing the measurements to the SE either directly or after processing. Then, the process considers the matching

of the outputs to the inputs based on their semantic annotations. That is, the value produced by a sensor can be fed to the SE only if its location, physical property and measurement unit match to the respective properties expected by the SE.

The "Configuration Selection" process considers the SE implementations ranked off-line based on the following QoS criteria: i) preference is given to hybrid implementations if PMUs exist, otherwise conventional implementations are used; ii) preference is given to implementations that make use of the maximum number of measurements. Therefore, in case more than one semantically valid SE implementations are found, the one with higher ranking against the pre-defined QoS criteria will be selected. The "Semantic Reasoning" and "Configuration Selection" processes are combined in algorithm 1. The execution of the algorithm is made clearer with the illustrative use cases in Section 5.1.6.

5.1.6 Use Cases

For the purpose of presenting the concept, two SE implementations are assumed available (f_1^{se}, f_2^{se}) in the components' database $\mathcal{F}$, from which the Supervisor Σ will select according to the semantic reasoning output. The f_1^{se} is a WLS implementation of a conventional estimator and as such it uses only the eight conventional measurements from the twelve in the example EPS of Fig. 5.4. On each execution it (iteratively) estimates the EPS states. On the other hand, the f_2^{se} is a WLS implementation of a hybrid estimator, which is able to use additionally the synchrophasor measurements from the PMUs and integrate them in the iterative estimation of the states. The implementation is making use of "pseudo-measurements" to overcome the issue with the current phasor measurement, discussed in [7].

Concerning the measurement devices, it is assumed that no PMU is initially deployed at the EPS of Fig. 5.3. That is, the measurement vector at configuration $I = 0$, comprises eight measurements (modelled here as eight sensors respectively), including the active and reactive power injected on Bus 1, as well as the active and reactive power flows of all three lines. As explained in Section 5.1.5, each of the measurements corresponds to a specific location in the EPS (e.g., specific bus or line side) and a specific measurement unit (e.g., kW, V). On the other hand, the SE implementations need to receive measurements from specific locations, specific physical properties (e.g., reactive power, voltage) and specific measurement units

Algorithm 1 Semantic Reasoning and Configuration Selection Algorithm

procedure RUN ALGORITHM

for each SE application under the control of Supervisor Σ **do**

Find all sensors that are capable of measuring the properties expected by the SE.

for each available SE implementation, starting with the one of higher offline ranking **do**

Find whether the sensors identified earlier measure all of its required inputs.

Exploit also available transformation functions for the sensor's output signals.

if successful **then** $flag = 1$

Exit loop

else

Continue with next SE implementation

end if

end for

if $flag == 1$ **then**

Configuration is confirmed, therefore use the respective components and continue the operation of the SE.

end if

end for

end procedure

(pu), as derived by the function vector $h_0(x)$.

During the execution of algorithm 1, the semantic matching is checked between available sensors and SE implementations. For instance, the semantic annotation of the output of sensor f_1^s producing a value in kW at Bus 1 is $\lambda(y_1^s) = \{l_1 = \text{Bus 1}, q_1 = \text{active electric power}, p_2 = \text{stateMeasurement}, m_1 = \text{kW}\}$. On the other hand, the semantic annotation of one of the inputs of the SE implementation f_1^{se} representing the active power injected at Bus 1 in pu is $\lambda(\hat{x}_1^{se}) = \{l_1 = \text{Bus 1}, q_1 = \text{active electric power}, p_2 = \text{stateMeasurement}, m_6 = \text{pu}\}$. It can be seen that $\lambda(y_1^s) \neq \lambda(\hat{x}_1^{se})$, that is, the specific measurement cannot be used directly in the SE. The matching becomes possible only through a transformation function f_1^y which is

able to appropriately transform the kW unit to pu for the specific EPS. The semantic matching of other measurements to the respective inputs of the SE implementation f_1^{se} is checked and confirmed in the same way. Since we do not implement any online performance evaluation of the selected configurations, we assume that the SE implementations are ranked off-line, based on certain criteria, e.g. number of PMU measurements utilised, etc. Therefore, at configuration $I = 0$, considering the QoS criteria for the off-line ranking, the Supervisor Σ will select f_1^{se} to execute the iterative state estimation.

At time $k_1 > k_0$, the PMU has been deployed on Bus 2 (location l_2). This PMU is modelled with four (virtual) sensors semantically annotated in the Knowledge Graph, measuring the magnitudes and angles of the voltage and electric current at the subject location. Following the detection of this new availability of components, the Supervisor Σ re-executes the algorithm 1 before the next state estimation execution, to select the SE configuration $I = 1$. Checking for f_2^{se} first, which has the higher off-line ranking if all measurements can be consumed, Σ is able to confirm the semantic matching of all but one measurements to the corresponding f_2^{se} inputs; the function-vector $h_1(x)$ is assumed updated properly. The not matching measurement is associated with the electric current phasor produced by the sensor f_2^s in m_4 = kA, since f_2^{se} expects a power flow "pseudo-measurement" in pu. However, it is assumed that in parallel to the installation of the PMU, a function f_2^y becomes available and is semantically annotated in the Knowledge Graph, which transforms f_2^s output to the expected active power "pseudo-measurement". The semantic matching can be then confirmed, thus enabling the Supervisor Σ to selects the hybrid estimator f_2^{se} and feed its inputs with all available measurements in order to execute the state estimation. It is noted here that the SE implementation f_1^{se} also belongs to a semantically valid configuration option, however, that configuration does not make use of all available measurements and as such it is rejected by the pre-defined QoS criteria.

5.2 Semantic Mediation in Smart Water Networks

Water Distribution Networks (WDN) are the infrastructures responsible for delivering drinking water to consumers. They are considered among the most Critical Infrastructures, along with Electric Power Systems and Telecommunications Systems. The effective monitoring and control of these systems is of vital importance since

they can significantly affect the health, safety, security and/or economic well-being of the citizens when disrupted or when their operation degrades. The advancements in coupling WDN with the ICT infrastructure, combined with the more recent introduction of smart sensing and actuation technologies, have enabled the enhancement of "Supervisory Control And Data Acquisition (SCADA)"-based applications. In current WDNs, these applications assume pre-defined configuration and characteristics of the involved components (sensors, actuators, controllers, etc.). In our work, we have explored how a SEMIoTICS-based architecture may contribute to the online configuration of the WDN monitoring and control architectures by exploiting and reasoning over the capabilities of deployed devices.

Typically, the objectives of WDNs are to deliver water of sufficient quality and quantity to the consumers, maximize the efficiency of this delivery, as well as guarantee the safety of the system. WDNs are essentially large-scale systems, which consist of pipe networks and dynamical elements such as water storage tanks, pumps and valves to control pressures and flows in the system, as well as sensors measuring various hydraulic and quality water characteristics. In practice, the state-of-the-art in the monitoring and control of these systems, involves the use of SCADA systems coupled with an ICT infrastructure that enables the transfer of data and further processing of sensing and actuation signals [16]. Figure 5.5, presents the typical architecture of a WDN monitoring and control system, capturing both the hydraulic and quality characteristics. Hydraulic sensors measure tank water levels, hydraulic heads of junctions (i.e., the surface elevation of the junction comparing to some reference level), flows and pressures, while quality sensors measure pH, chlorine concentrations, Oxidation Reduction Potential, Total Organic Carbon, etc.. The inputs to the system are generated by hydraulic actuators (e.g., valves, pumps), as well as quality actuators (e.g., chlorine disinfection boosters). The control decisions are implemented based on pre-defined rules or control algorithms that map the measurements to appropriate actions. It is also noted that, in practice, water utilities typically employ manual sampling and control. However, in some countries they have started employing various types of sensors, such as Automatic Meter Readers (AMR) to measure water consumption in real-time, as well as other hydraulic and quality sensors; in contrast, real-time monitoring and control algorithms are still under research and have not been exploited by the WDN operators.

As the number of sensors and actuators in the WDNs increases, so does the com-

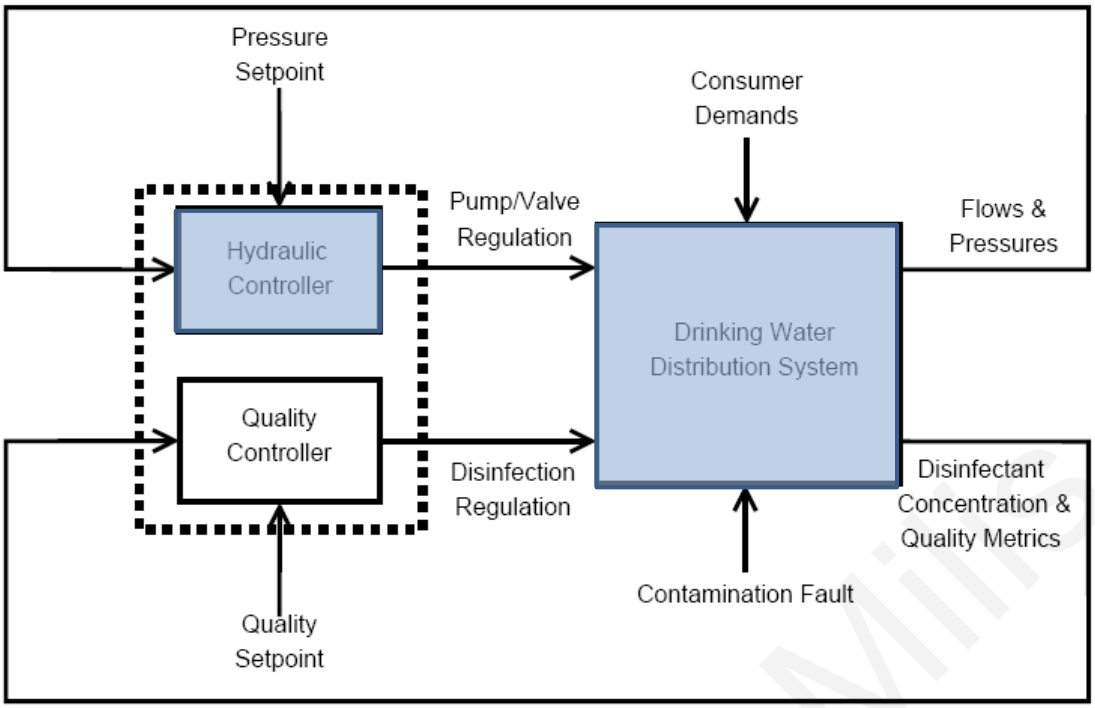


Figure 5.5: The typical architecture of a WDN Monitoring and Control system. Our contribution focuses on the components highlighted with blue transparent colour

plexity in managing these elements and reconfiguring the system whenever a sensor or actuator is added or removed. In practice, the measurements from the WDN are retrieved by physical devices of appropriate types and of a variety of (vendor-dependent) specifications. Therefore, either the control engineer needs to design the control law based on pre-acquired knowledge about the available devices and their specifications or the technicians that install the devices need to know the specifications of the SCADA system and the control algorithms in advance, so as to install appropriate devices. These cases show the existing inflexibility of the current WDN monitoring and control architectures. The recent advancements in Smart Water Networks and related sensing and actuation capabilities create additional considerable opportunities and challenges towards offering flexibility in the monitoring, control and event detection architectures [29–31]. However, changes in the sensing and actuation capabilities during operation of the system, may necessitate changes in the monitoring and control algorithms, as well. This implies a need for intervention of human experts, e.g., control engineers to update the controllers given any new sensing and actuation capabilities.

We introduce a customised version of the SEMIoTICS architecture and system,

so as to contribute to accommodating component changes and subsequent online (re-)configurability. The semantically-enhanced Supervisor Σ executes its "Semantic Reasoning" process after any change in the available components and takes decisions about the online re-configuration of the system. The situation awareness is achieved by utilizing a properly adjusted version of the "Knowledge Graph" $\mathcal{G}$, which fully describes the types of the components, their characteristics, their locations, as well as the physical properties they measure or act upon. Structuring the knowledge representation, enables a machine (i.e., the Supervisor Σ) to undertake the tasks that would otherwise be undertaken by humans.

5.2.1 Background on modeling and control of hydraulics

Modelling methodologies for WDN hydraulic and quality dynamics and their faults have received significant attention during the last decade [51, 108], while they are still an area of active research. The present work focuses on the hydraulic characteristics of a WDN, so as to achieve a simple presentation of the proposed semantic enhancements and demonstrate the concept.

The hydraulic feedback control problem in water systems can be defined as the problem of computing at each discrete time k, the input vector $u(k)$, representing the instructions to be given to the pumps and valves, so that the measured hydraulic parameters $y(k)$ (tank water levels, hydraulic heads, as well as the pipe flows) operate within certain bounds or follow a reference signal vector $r(k)$ specified for safe operation. The control law is given by (5.2).

$$u^c(k) = f^c(y^c(k), r(k); \zeta^c). \tag{5.2}$$

The above control law depends on the (potentially pre-processed) measurement or estimation of the hydraulic parameters, i.e., the vector $y^c(k)$ is in general a transformation of the actual measurements vector $y(k)$. The latter is defined as the hydraulic analysis problem in WDNs, which considers computing the hydraulic head at each junction (i.e., surface elevation comparing to a reference level), the water levels at each tank and the flows at each pipe. To solve this problem, the topology of the network and pipe characteristics, the control inputs, as well as the demand at each node, are assumed known *a priori*. Typically, structural information of the network is available by the water utilities, while pipe characteristics may require field mea-

surements and nodal demands at each discrete time can only be estimated using historical data and other hydraulic measurements available (if no online demand sensors are used by the utility to monitor each consumer).

The dynamic relation of water flow in pipes and the differences in the hydraulic heads can be described by a set of ordinary differential equations. In practice, however, the heads and flows are approximated using an iterative optimization algorithm (e.g., gradient descent), in discrete time and in steady state, so that the conservation of mass and energy is satisfied [117]. For example, consider a water distribution network composed of pipes, junctions and water storage units. The topology of this network can be represented as a graph with edges corresponding to pipes, and nodes corresponding to junctions and water storage units. At discrete time k with sampling time Δt, let $d_i(k)$ be the consumer demand outflow at the i-th junction node, and let $w_j(k)$ correspond to the flow in the j-th pipe connected to junction i ($j \in \mathcal{A}_i$ where $\mathcal{A}_i$ is the set of pipe indices which are connected to the i-th node, assuming that inflows have a positive sign and outflows have a negative sign). In accordance to the principle of mass conservation, the sum of all pipe inflows and pipe outflows must equal to the demand (Kirchhoff's junction rule), such that:

$$\sum_{j \in \mathcal{A}_i} w_j(k) = d_i(k) \tag{5.3}$$

Furthermore, in accordance to the principle of energy conservation, the flow-head-loss relationship across each link in the network must be balanced. Let $h_i(k)$ be the *hydraulic head*, i.e. a measurement of water pressure expressed in length units, at the i-th node. For water moving from node j (higher head) to node i (lower head) with flow $w_l(k)$ in the l-th pipe, the flow-head-loss relationship is given by:

$$h_j(k) - h_i(k) = f_h(w_l(k)) \tag{5.4}$$

where $f_h(\cdot)$ is a nonlinear function, such that $f_h(w_l(k)) = \alpha_r w_l(k)^{\alpha_f} + \alpha_m w_l(k)^2$, which depends on the pipe resistance coefficient α_r, the flow exponent α_f and the minor loss coefficient α_m. These parameters are computed using empirical methods [103]. Therefore, the set of hydraulic equations in a WDN is constructed, and at each discrete time, a gradient optimization algorithm is solved to estimate the heads at each junction/tank, using the current demand flows, current control inputs and current tank heads [117].

Tanks are dynamic elements in the system and can be considered as nodes in the WDN; the head state of the i-th water tank node is given by (5.5).

$$h_i(k+1) = h_i(k) + \frac{\sum_{j \in \mathcal{A}_i} w_j(k)}{f_{T_i}(h_i(k))} \Delta t, \tag{5.5}$$

where the tank head $h_i(k)$ corresponds to the relative tank water level plus the tank elevation, and the function $f_{T_i}(\cdot)$ computes the cross-sectional area of the i-th tank at a certain height. Initial tank heads are typically known.

Currently, a number of off-the-shelf software tools are used to perform the hydraulic analysis in WDNs, such as the open-source EPANET [103].

5.2.2 Formulation

Consider the network in Fig. 5.6. The arrows indicate the flow of water in pipes, as well as the inflow of the tank (w_i, $i = 0, ..., 9$). The tank flow is indicated by w_0. The nodes indicate the junctions with their hydraulic heads represented by h_i, $i = 0, ..., 7$ and the corresponding consumer demand outflows d_i. The tank head is indicated by h_0.

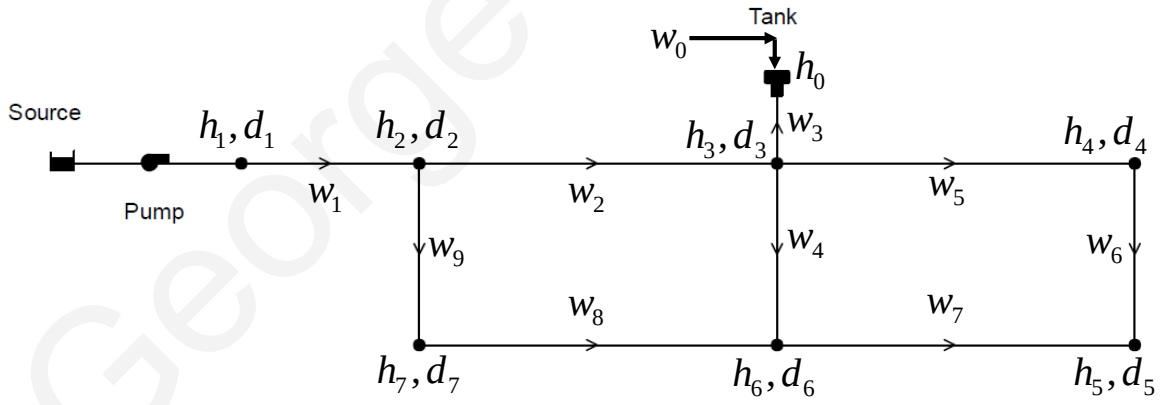


Figure 5.6: A simple WDN with six junction nodes; water is supplied by a reservoir and a tank. When the tank water level goes below 110ft, the pump is activated, and when the tank water level goes above 140ft, the pump stops.

Consider also the control objective to regulate the tank head $h_0(k)$ at a given reference level $r_0(k)$ (or within a certain bound), at discrete times k, through the application of a control law as in (5.2). In this example, the pump is activated only when the tank water levels has reached a minimum value (110 ft), and the pump stops working when the water level has reached a maximum value (140 ft).

As explained in Section 5.2.1, the tank and junction heads, if not directly measured, are estimated iteratively at each discrete time k, using the mass conservation equations on junctions, as well as the energy conservation equations that require the head-loss functions. The parameters of the head-loss functions $f_h(\cdot)$ are considered known or are computed empirically. In addition, the demand flows at each consumption node d_i, $i = 1, ..., 7$, as well as the initial tank head $h_0(0)$, are known. At time k, an optimization algorithm is used to compute the unknown states. Then, the new tank head is computed again and the problem is solved for the next discrete time $k + 1$.

The tank head regulation process can be illustrated with the block diagram of Fig. 5.7, where the vector of the states of the WDN (i.e., the tank and junction heads and pipe flows) is defined as $x \in \mathcal{R}^{n_x}$, the measurements' vector produced by the installed set of measurement devices is given by $y \in \mathcal{R}^{n_y}$, while $y' = \hat{x} \in \mathcal{R}^{n_x}$ denotes the vector of the estimated system states after the application of the iterative optimization algorithms. The diagram shows that a controller uses the measured or estimated tank-head as input and produces an action signal that is then utilized by the actuators (e.g., pumps, valves) to act on the system and affect the tank-head.

The control implementation for the regulation of the tank-head is typically comprised of a set of pre-defined rules, such as the following: `IF <tank-head> <expression referring to tank head> THEN <action to be performed on the specific tank>`. In current practice, the design of such control architectures in WDNs, is based on a fixed configuration of specific components, with specific measurement and actuation devices deployed and pre-determined control laws. That is, the WDN operators need to decide in advance the types of components to use and where to deploy them in the network's topology, as well as implement in advance the respective control rules. Moreover, integration of control components (e.g., after a change happens) typically requires manual configuration by an experienced engineer. It is therefore typical to interrupt the normal system operation to modify the control system, i.e., to manually configure the new sensor/actuator in the SCADA system, as well as manually modify the logic in the micro-controller (for when to turn-on/turn-off a pump).

As the scale of the system increases, having highly specialized personnel which is able to perform the above described increasingly complex functions, becomes a real challenge. A "smart water" cyber-physical implementation would be expected to be

able to adapt to changes in the configuration of the control system (e.g., removal of a faulty tank water-level sensor, addition of pipe flow or tank inflow measurement, updated set of control rules, etc.) and perform any necessary re-configurations online, in order to avoid the need of downtime. We demonstrate here the use of a customised version of SEMIoTICS architecture and system, which introduces an intermediate layer that becomes aware of the characteristics of available components, performs the necessary output-input semantic matchings and makes an online decision on how to (re)configure the control system.

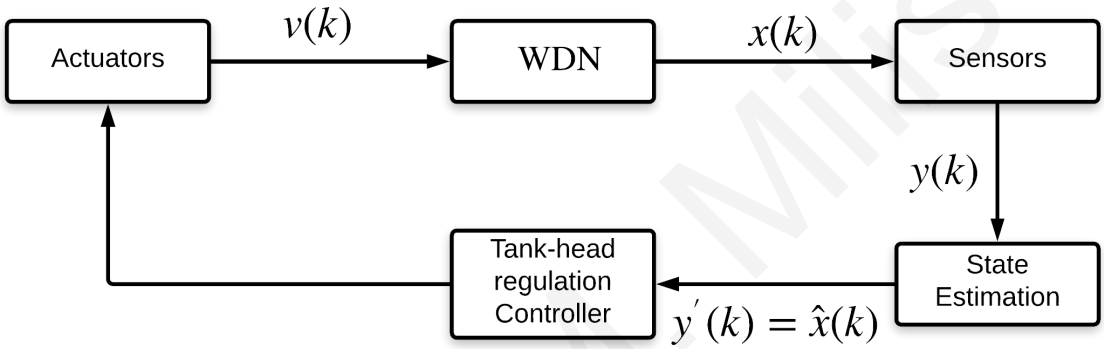


Figure 5.7: Block diagram of the tank head regulation implementation. A set of sensors measure part of the states of a WDN (i.e., the tank and junction heads and pipe flows). Then a state estimator (SE) estimates the complete set of states, by producing the vector $y' = \hat{x}$, which is then fed to the controller to help it compute the input to the set of available actuators (e.g., pumps, valves).

5.2.3 Solution

The relevant customised SEMIoTICS architecture for the tank head regulation problem, is depicted in Fig. 5.8, where $I \in \{0, 1, 2, ...\}$ is the index of the different control architecture configurations. In configuration I, the operation of the WDN is monitored by a set of sensors $\mathcal{F}_I^s$, e.g., tank and junction head, as well as pipe flow sensors. The measurements may pass through a set of pre-control processing functions $\mathcal{F}_I^y$ (such function might also be the state estimator that computes missing state-values). Then the appropriate control implementation is chosen from a set of available controllers $\mathcal{F}_I^c$, to drive a set of matching actuators $\mathcal{F}_I^a$.

The semantically-enhanced Supervisor Σ first detects and identifies any new

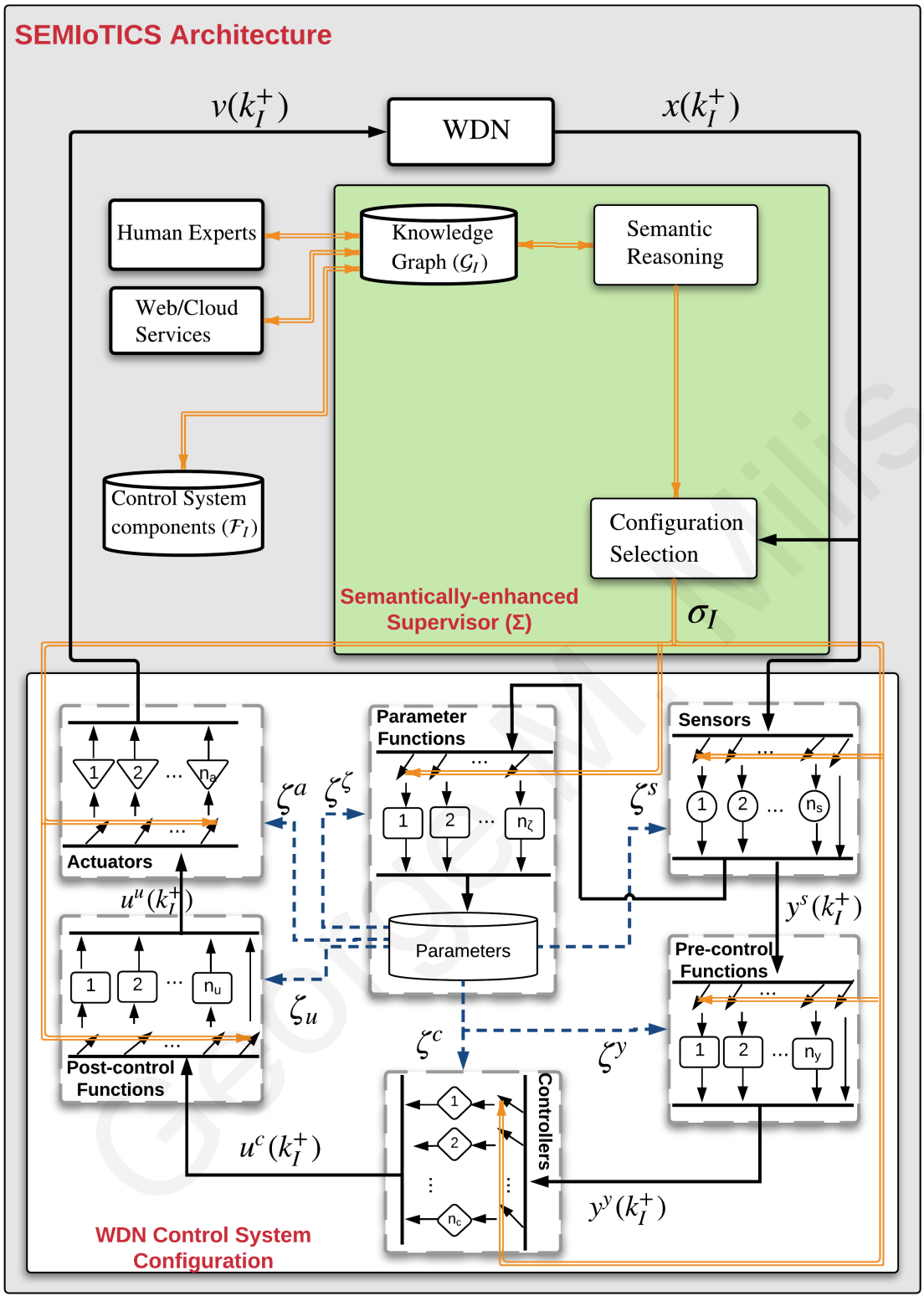


Figure 5.8: The customised SEMIoTICS-based architecture, where the Supervisor Σ performs the reconfiguration of the control system

component(s) added. The physical communication among components is facilitated through an assumed existing communication protocol, e.g., with extensions to the

currently adopted SCADA systems (the details of this fall outside the scope of our work). Subsequently, Σ becomes aware of the characteristics and capabilities of the new components, through their semantic annotations (see Section 3.3.2). The semantic annotation of each component describes its characteristics and capabilities. Once this information is received by Σ and is stored in the "Knowledge Graph" $\mathcal{G}$, it is integrated with the existing knowledge available about the overall system. The whole of the information is in turn used to apply the "Semantic Reasoning" process and reconfigure the control system considering all available components, e.g., to utilize the state estimation function if the tank water level (or tank head) is not measured directly. The Supervisor Σ becomes aware of any new measurement (e.g., a flow sensor) or actuation unit (e.g., a valve) and/or controller deployed in the WDN. It is emphasized here that the switching to a new configuration is triggered by changes in components. In case the change corresponds to failures of running components, the control service to the WDN will unavoidable be interrupted until the new configuration is put in operation. To reduce inconvenience, the switching is performed between two consecutive control decisions. The time constants of the WDN allow the appropriate selection of the time between consecutive control decisions, so as to be enough for Σ to complete the reasoning and decide upon the new configuration.

5.2.4 Semantic Reasoning and Configuration Selection

An illustrative scenario follows, aiming at clarifying how the semantic reasoning is performed over the stored knowledge facts to implement the decision mechanism for the online configuration of the WDN control system. Let assume the simple WDN of Fig. 5.6, where the initial configuration ($I = 0$) of the tank head regulation system consists of a sensor measuring the tank head h_0 (or water level) in meters (m) and an on/off pump which increases the pressure in another place of the system, in order for water to flow into the tank. In addition, a simple programmable logic controller (PLC) is considered available, implementing "IF-THEN" rules that, given the desired level of water in the tank, map the measured level to decisions on whether to close or open the pump. The "things" contained in the introduced system are essentially one sensor (f_1^s), the physical property "tank-head" (q_1), the measurement unit "meters" (m_1), as well as the "tank" and the "pump-position" as

system features-of-interest/locations (l_1 and l_2 respectively).

The semantic reasoning process first considers the types of the components and their expected role in the WDN control implementation. The position of each type of component is fixed, with sensors always positioned to measure WDN outputs and then passing the measurements to a controller either directly or after processing. Then, the output-input semantic matching is performed. The semantic reasoning and configuration selection processes of the Supervisor Σ, are described through the algorithm 2. The controllers are assumed ranked off-line according to pre-defined quality criteria, e.g., give priority to controllers that utilise bigger number of available components. This reduces the complexity of the configuration selection process. The details of the ranking is out of the scope of this work.

The execution of the algorithm will be made clearer with the use case below.

5.2.5 Case Study

Consider the configuration $I = 0$ introduced earlier and the control objective of regulating the tank-head at a desired point. The re-configuration algorithm first explores the actuators that can act on the tank-head in the example WDN. The semantic annotation of the output of actuator f_1^a is given by: $\lambda(v_1^a) = \{l_2 = \text{pump}, \boldsymbol{Q}, p_1 = \text{regulate}, m_2 = m^3/h\}$. The semantic annotation of the input of the plant is: $\lambda(v_1^p) = \{l_1 = \text{tank}, q_2 = \text{tank water inflow}, p_1 = \text{regulate}, \mathcal{M}\}$. Moreover, the pump position directly affects the tank head, therefore, the location l_2 is modelled in the "Knowledge Graph" $\mathcal{G}$ as related to the tank location l_1. Through the location transformation, it can be derived that the two semantic annotations share a point in the respective four-dimensional semantic annotation space.

The next step of the algorithm considers the matching between a controller and the selected actuator in previous step. The input of the actuator is semantically annotated as: $\lambda(u_1^a) = \{l_2 = \text{pump}, q_1 = \text{flowRate}, p_2 = \text{controlDecision}, m_3 = \text{on/off}\}$. On the other hand, the output of the controller is annotated as: $\lambda(u_1^c) = \{l_1 = \text{tank}, q_1 = \text{flowRate}, p_2 = \text{controlDecision}, m_3 = \text{on/off}\}$. Again, the semantic matching is confirmed through the locations' transformation. The semantic matching of the plant's output to the sensor input and the semantic matching of the sensor's output to the controller's input are checked in the same way.

At time $k_1 > k_0$, the sensor f_1^s stops working and needs to be replaced. Since

Algorithm 2 Semantic Reasoning and Configuration Selection Algorithm

procedure RUN ALGORITHM
    Find all actuators that are capable of acting on the tank water level
    If no appropriate actuators can be found, then report inability to meet the control objectives and stop, otherwise proceed to next step.
    Find all sensors that are capable of measuring the hydraulic parameters.
    for each available controller implementation, starting from the one with higher off-line ranking **do**
        Check if it has the capacity to drive the actuators found in previous step.
        Exploit also available transformation functions for the control signals.
        If successful, $flag1 = 1$
        if $flag1 == 1$ **then**
            Find whether the sensors identified earlier measure all required by the controller's inputs.
            Exploit also available transformation functions for the sensor's output signals.
            If successful, $flag2 = 1$
            Exit loop
        else
            Allow capacity to drive fewer actuators than available and continue with the next controller
        end if
    end for
    if $flag2 == 1$ **then**
        Match is confirmed, therefore close the loop with the matching components and continue operation of the system for the specific application.
    end if
end procedure

an immediate replacement is difficult to be found, the utility gives instructions to technicians to install two other sensors, f_2^s with output semantic annotation: $\lambda(y_2^s) = \{l_3 = \text{pipe entering tank}, q_2 = \text{tank water inflow}, p_3 = \text{stateMeasurement}, m_4 = \text{feet}\}$ and f_3^s with output semantic annotation: $\lambda(y_3^s) = \{l_4 = \text{pipe leaving tank}, q_3 = \text{tank water outflow}, p_3 = \text{stateMeasurement}, m_4 = \text{feet}\}$. The Knowledge Graph $\mathcal{G}$

has been also enriched by an "adjacent-to" relation between locations l_3 and l_4 with l_1, as well as the semantic annotation of a function f_1^y that takes as input the inflow and outflow of the tank and computes the tank head, as discussed in Section 5.2.1. The change of the components triggers the execution of the algorithm 2 by the Supervisor Σ, which exploits whether a new configuration $I = 1$ can be achieved by utilizing the new components.

As far as the semantic matchings of the available actuator to the plant and the controller to the actuator are concerned, the algorithm retrieves exactly the same results as for configuration $I = 0$. However, when it comes to the matching of the sensor's output to the controller's input, this can be confirmed only through the function f_1^y which produces the signal expected by the controller f_1^c after processing the outputs of the sensors f_2^s and f_3^s. The outputs of the sensors (a value in feets) is a measurement unit of the tank head as required, therefore the Supervisor Σ decides the configuration $I = 1$.

5.3 A Semantically-enhanced Fault-Detection Design Architecture

Large-scale systems consist of several cyber and physical components, like sensors for monitoring system states, electrical and mechanical actuators, controllers, and a number of other data/signal processing components. Over time, it is inevitable that one or more of these components will fail or system dynamics will move out of expected bounds due to external events, necessitating the utilization of fault/event detection and isolation mechanisms [22,64]. By enabling the detection and diagnosis of miss-operation of systems, these mechanisms can help, e.g., in saving energy, in reducing economic cost and/or in avoiding critical consequences of cascading effects due to inter-dependencies with other systems.

During the last two decades, various methodologies have been developed and proposed for detecting, identifying, isolating and accommodating faults [22, 59, 64]. In general, fault detection methods can be classified into model-free (or data-driven) and model-based methods.

Model-free methods are the most commonly used, since they can be developed without the requirement of understanding the details of the underline system's

dynamics [64]. Examples are, quantitative methods (e.g., neural networks, statistical classifiers), and qualitative methods (e.g., expert systems, fuzzy logic, pattern recognition, trend analysis) [127].

Model-based methods, on the other hand, require additional modelling and calibration effort, since a model with physical significance has to be developed using *a-priori* knowledge of the system. Again, there are examples of quantitative methods (e.g., observer-based/Kalman-filter state and parameter estimation, parity space) and qualitative methods (e.g., fault-trees and other causal models) [127].

Fault diagnosis methodologies with learning capabilities have also been proposed in the past years, which combine model-based analytical redundancy and computational intelligence tools, i.e. neural networks, to detect faults and to learn the unknown fault dynamics [38,93,126]. By learning the unknown fault dynamics, isolating the type of fault and identifying its magnitude, it is possible to change the control input to accommodate the fault, during operation [97].

Designing a fault-detection scheme for a certain system is a complicated procedure which relies on the knowledge and reasoning capabilities of a human expert. In practice, a human expert should have a broad background knowledge of tools (e.g., state-of-the-art fault-detection methods, online learning methods based on computational intelligence, state-estimation methods, etc.) and in which situations these are best suited, in order to make an informed selection that fully exploits the available measurements, constraints and objectives. Depending on the system which needs the fault-detection service, as well as the preferred fault-detection method, different schemes can be designed, consisting of smaller components. In practice, it is very rare, if not impossible, to find and employ a human expert of such breadth of knowledge whenever a fault-detection scheme is required for a certain system. An additional drawback in current practices is the lack of mechanisms to allow online (and where possible automatic) replacement of individual components or of the overall fault-detection scheme. Therefore, in case new and advance methods become available and are implemented as components, they can only be deployed by expert engineers who will re-design the overall fault detection scheme. This may inhibit industry adoption of advanced fault-detection methodologies and may act as a barrier to the exploitation of research results.

The above motivated our work on designing a SEMIoTICS-based architecture and system, to be able to reproduce part of the reasoning procedure of a human expert,

towards designing fault-detection schemes. This architecture allows new components (e.g., new on-line learning algorithms or new fault-detection algorithms) to be gradually deployed as they become available, by automatically configuring the components participating in the fault-detection scheme. It is emphasized that our work does not focus on the design of any new fault-detection algorithms or components, but rather on the online configuration of fault-detection schemes using existing components. Moreover, it is envisioned that the adoption of a component-based fault-detection design will facilitate the faster exploitation, testing and demonstration of academic research in industrial applications. To demonstrate the application of the SEMIoTICS architecture, a case-study of configuring a contamination event detection scheme with online learning capabilities for drinking water distribution networks is presented in this section.

5.3.1 Problem formulation

Fault detection is defined as the problem of determining whether a system is operating under normal or abnormal conditions (e.g., due to the occurrence of a system, actuator or sensor fault). Typically, a fault-detection algorithm is specifically designed for a certain system, taking into account the system's measurable variables, known dynamics and other information available. The output of a fault-detection algorithm at discrete time k, is given by:

$$d(k) = f(y(k), u(k); \zeta), \tag{5.6}$$

where $f(\cdot)$ is a fault-detection composite function, $y(k)$ and $u(k)$ are the measured output vector and the known input vector of the system respectively and ζ is a set of parameters related to the system and the considered fault-detection implementation. In general, the detection signal $d(k)$ can be a vector corresponding to a set of various fault-level classes. Depending on the specifications, the detection signal $d(k)$ can be: i) binary, i.e., of the form $\{0, 1\}$ or {True, False}, thus informing of the detection of a fault or not; ii) a real number, e.g., representing the probability or the risk of fault existence; iii) a more generic class of values, e.g., a fault class type, a color-based risk-level scheme, a linguistic variable, a fuzzy value, etc.

Similarly to the feedback control case, two families of fault-detection algorithms are typically considered: the "model-free" and the "model-based" methods [64].

The former process the measured output signals plus other known or computed signals that may be required, in order to generate certain features (e.g., operation state). These features are passed through a "detection logic" component and are compared to their value in non-faulty operation. Typical examples are the limit-checking approach, the change-detection approach (such as the CUSUM) [10] and other statistical-based approaches. In addition, learning methodologies have also been applied within a model-free fault detection context [5]. On the other hand, the "model-based" fault-detection methods process the measured output and known input signals utilizing also a known system-model, in order to generate certain features (e.g., state estimation residuals). As in the model-free case, these features are then passed through a "detection logic" module and are compared to their values in non-faulty operation. Typical examples are the analytical redundancy fault-detection schemes, utilizing tools such as state-estimation, filtering, parametric uncertainty learning and adaptive approximations [44, 65, 93, 96–98]. In the general case, the fault-detection algorithm $f_d(\cdot)$ can be considered as composed of sub-components, some of which are basic (mandatory) for all implementations of fault-detection while others are required only in certain cases. All these components are discussed in the sequel.

Basic Components

The first basic component of a fault-detection scheme is the "Detection logic", given by the function:

$$d(k) = f^d(r(k), t(k); \zeta^d), \tag{5.7}$$

where $d(k)$ is the detection signal defined also in (5.6), $f^d(\cdot)$ is the detection logic implementation, $r(k)$ is a feature signal which is computed by a separate function in order to be compared against a threshold signal $t(k)$, and ζ^d is a set of other parameters required by the adopted detection-logic method.

The second basic component in a fault-detection scheme is the "Feature" given by the function:

$$r(k) = f^r(y(k), u(k), \hat{x}(k); \zeta^r), \tag{5.8}$$

where $r(k)$ is the feature signal defined in (5.7), $f^r(\cdot)$ is the implementation of a method

to derive the signal, $y(k)$ and $u(k)$ are the measurable outputs and the known inputs of the system respectively, if available, $\hat{x}(k)$ is the estimated system state (optionally used) and ζ^r is a set of parameters required by the adopted method.

As an example, in a model-free fault-detection scheme, the detection logic function f^d may be a limit-check, e.g., comparing the measured state $r(k) \equiv y(k)$ or its difference $r(k) \equiv \frac{y(k)-y(k-1)}{\Delta\tau}$ with a given upper and/or lower bound $t(k)$, such that $|r(k)| \leq t(k)$. Another example is the CUSUM change-detection method, where the cumulative sum of the differences of the measured state from a pre-defined parameter (e.g., the statistical mean of the state signal) are compared with a given threshold. In a model-based fault-detection scheme, the detection logic function f^d may be comparing the state-estimation error $r(k) \equiv y(k) - \hat{x}(k)$ with a given or computed threshold signal.

A third basic component of a fault-detection scheme is the "Threshold", which undertakes the task of generating the threshold signal against which to compare the detection-logic features and is given by the function:

$$t(k) = f^t(g(\cdot), y(k), u(k); \zeta^t), \tag{5.9}$$

where $t(k)$ is the threshold signal defined in (5.7), $f^t(\cdot)$ is the implementation of a method to derive a threshold, $g(\cdot)$ is a function representing new (additive) system dynamics, $y(k)$ and $u(k)$ are the measurable outputs and known controlled inputs of the system respectively (optionally available) and ζ^t is a set of parameters required by the adopted implementation of the function. For instance, the threshold parameters may correspond to bounds on parameters or on parts of the system state dynamics, derived from expert knowledge about the system operation or from the off-line processing of historical data, etc.; it may be the output of a stochastic process on the detection logic features or it may be computed given knowledge about a system model with parameter uncertainty or function uncertainty. In all cases, the threshold may be a constant value or a time-varying adaptive signal.

In summary, at a minimum, the fault-detection scheme is composed of the functions specified above, such that $f \equiv f^d(f^r(\cdot), f^t(\cdot), \zeta)$. That is, the detection logic component compares measured or computed features of the system, against a pre-selected or computed threshold signal.

Advanced Components

In addition to the three basic components described in the previous sub-section, additional components may be required by certain model-based fault-detection schemes. For instance, in some model-based cases the estimation signal of the system states $\hat{x}(k)$ is required. For this, a "State-Estimation" component is required, given by:

$$\hat{x}(k+1) = f^e(g(\cdot), y(k), u(k); \zeta^e), \tag{5.10}$$

where $\hat{x}(k+1)$ is the estimated system states signal at the next time step, $f^e(\cdot)$ is the adopted implementation of the state-estimation, $y(k)$ and $u(k)$ are the vectors of system's measured output and known system inputs respectively, ζ^e is a set of other parameters required by the adopted implementation and $g(\cdot)$ is a function representing a new additive part of the system state dynamics. For instance, the State-Estimation component may correspond to a "Kalman filter" which produces estimates based on some prior knowledge about the states, a measurement vector and certain parameters of measurement and state's uncertainty; it can also be a "Luenberger observer" which, based on a known model of system dynamics and the available measurements, produces estimates of the state. The state-estimation may be also implemented as a black-box by a system simulation, e.g., using the EPANET software for simulating water distribution systems[1] or CONTAM for simulating contaminant propagation in buildings[2].

Furthermore, in the case of having a system model with unknown dynamics $g(\cdot)$, a "Learning Component" can be utilized, to learn the unknown function using a suitable approximation structure (e.g., neural network, polynomial function, radial-basis functions, wavelets, fuzzy systems, etc), such that $g \equiv \hat{g}$. This module undertakes the task to learn an unknown part of the overall state dynamics function and can be described in general by:

$$\hat{g}(k) = f^\theta(y(k), u(k), \zeta^\theta) \tag{5.11}$$

where $\hat{g}(k)$ is the estimated value of the unknown function, $f^\theta(\cdot)$ is the adopted online learning implementation and ζ^θ are any other parameters required by the adopted implementation (e.g., the convergence rate, knowledge about the structure of the

[1] http://www.epa.gov/water-research/epanet

[2] http://www.nist.gov/el/building_environment/contam_software.cfm

function, etc.). The output of this component may be used as input to components allowing update of the system model on which they base their implementation (e.g., certain State-Estimation or Threshold components).

All implementations of components (functions) of the types discussed above, can be considered as being elements of a function-set $\mathcal{F}$, thus forming a database of components (similarly to the case of control system components). The set $\mathcal{F}$ is defined as a superset of the following sub-sets of components:

- $\mathcal{F}^d = \{f_i^d | i = 1, ..., n_d\}$: all implementations of the Detection-logic function, with cardinality n_d
- $\mathcal{F}^r = \{f_i^r | i = 1, ..., n_r\}$: all implementations of the Feature function, with cardinality n_r
- $\mathcal{F}^t = \{f_i^t | i = 1, ..., n_t\}$: all implementations of the Threshold function, with cardinality n_t
- $\mathcal{F}^e = \{f_i^e | i = 1, ..., n_e\}$: all implementations of the State-Estimation function, with cardinality n_e
- $\mathcal{F}^\theta = \{f_i^\theta | i = 1, ..., n_\theta\}$: all implementations of the Online Learning function, with cardinality n_θ

Semantically-enhanced Supervisor

Depending on the system and the given fault detection specifications, an expert engineer would have selected and designed a fault-detection scheme using implementations of all basic components and possibly utilized additional components for state-estimation and learning, depending on the availability of measurements, models of state dynamics, as well as specific domain knowledge expertise. In other words, for the decision, the expert engineer relies on reasoning which considers the available knowledge about the domain and the fault-detection systems engineering, including the associated semantics of each component.

In this Section we describe a new, SEMIoTICS-based architecture, which is able to utilize pre-modelled expert knowledge and a set of fault-detection specifications (including performance criteria) and automatically design and configure a suitable fault-detection scheme, for a large class of systems. The subscript I, $I = 0, 1, ...$, when

used with variables or sets, denotes the state of the subject variable or the subject set following the re-configuration event I at time k_I. The challenge is formulated as:

$$\sigma_I = f^{\sigma}(\mathcal{G}_I, \mathcal{S}_I, \mathcal{F}_I), \tag{5.12}$$

$$f \equiv f_F(\sigma_I, \mathcal{F}_I^d, \mathcal{F}_I^r, \mathcal{F}_I^t, \mathcal{F}_I^e, \mathcal{F}_I^\theta), \tag{5.13}$$

where function f is the constructed fault-detection scheme, σ_I is a set of decision signals that correspond to the selection of specific components from the subsets of $\mathcal{F}_I$ defined earlier, $f_\sigma(\cdot)$ is the function implementing the semantic reasoning and the configuration selection processes, $f_F(\cdot)$ is the function which enforces the decision and the subsequent construction of the function $f(\cdot)$, $\mathcal{G}_I$ is the state of the respective "Knowledge Graph" that models the available knowledge, $\mathcal{S}_I$ is the set of fault-detection specifications given to the function (e.g., the preferred form for the detection signal).

5.3.2 SEMIoTICS (FD) Architecture

The SEMIoTICS-based fault-detection (FD) architecture, which implements the functions described in the previous section, is depicted in Fig. 5.9. The top-part of the figure illustrates the system on which the fault-detection service is performed. Assuming, for generality, a controlled plant by a closed-loop configuration, the system comprises the "Plant" with its states $x(k)$ measured by a set of "Sensors", the control configuration and the driven set of "Actuators" that act on the "Plant". The bottom-part of the figure shows the composite fault-detection scheme, comprising all components discussed in previous section, with their end-point connections. The inputs to the fault-detection scheme from the system are the measured outputs $y(k)$ and the known controlled inputs $u(k)$. The output is the resulted detection signal $d(k)$. It is noted that the dashed-line boxes host the sets of available components of each type at time k_I. The selection of specific implementations is performed through the decision signal σ_I (orange double line) given by the semantically-enhanced Supervisor Σ. The middle layer illustrates the design of the Supervisor Σ, which, in terms of processes, it is identical to the design described in 3. That is, the Supervisor Σ utilizes the state of the stored knowledge in the "Knowledge Graph" $\mathcal{G}_I$ (including the semantic annotations of components) and any given specifications $\mathcal{S}_I$ and produces a decision as to what implementations of components to adopt for the fault

detection. The decision signal σ_I is considered driving a function $f_F(\cdot)$ that invokes the selected implementations found in $\mathcal{F}_I$, the database of components at time k_I.

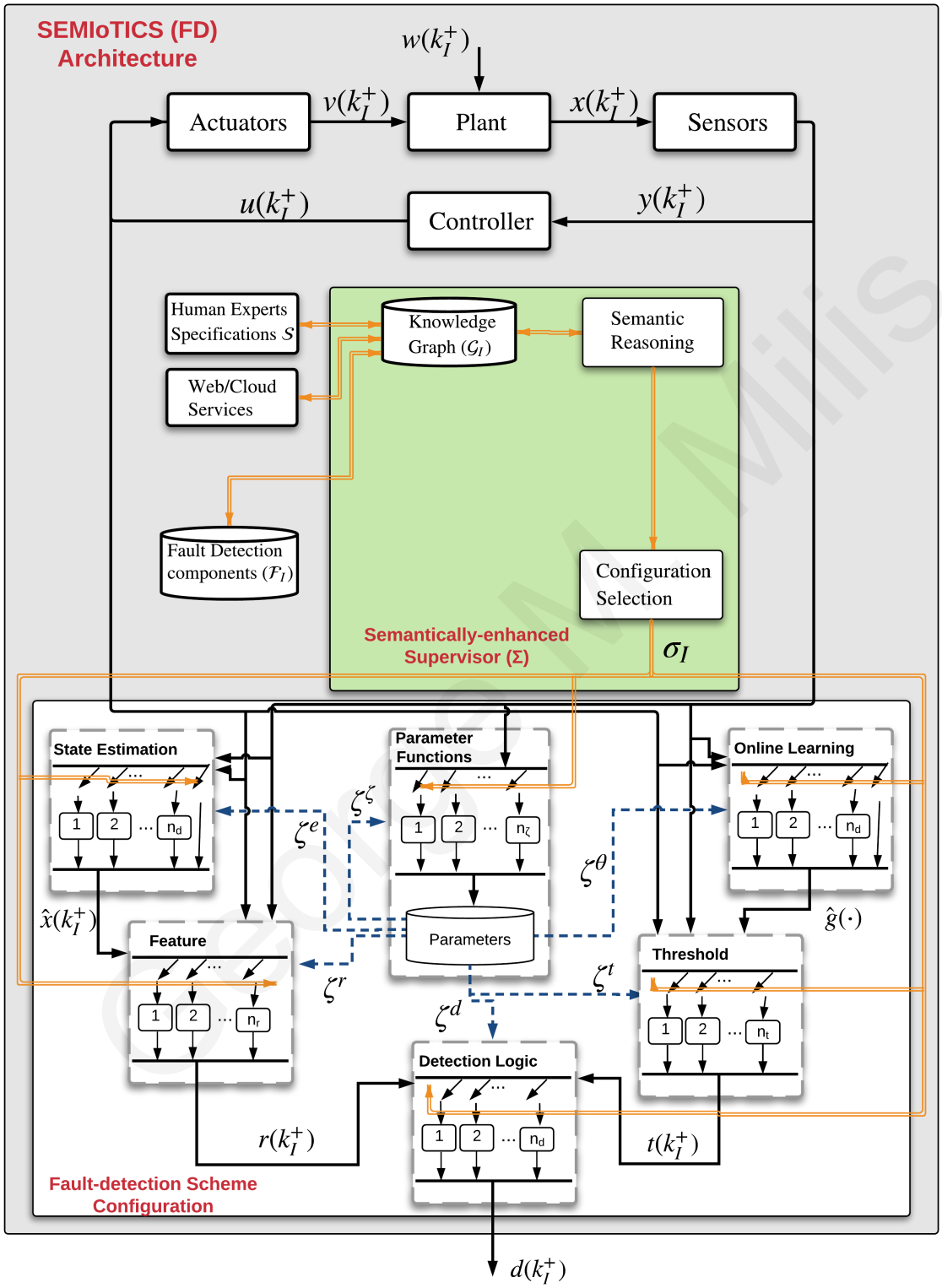


Figure 5.9: Block diagram of the architecture. Top: the System on which fault detection is performed; Middle: the Semantically-enhanced Supervisor Σ; Bottom: the Fault-Detection Scheme.

The following sub-sections go into details on the implementation of the "Semantic Reasoning" and "Configuration Selection" processes.

5.3.3 Semantic Reasoning and (FDS) Configuration Selection

The definitions and the design of the "Knowledge Graph" $\mathcal{G}$ have been already described in 3.3.1. What changes for the purposes of the application in fault detection schemes are the types of components and their fixed choreography, which affects the resulting "Configurations Graph". That is, the sets of sensors and actuators are considered pre-defined, the outputs of the components of types "Feature" and "Threshold" are given to components of type "Detection Logic", the outputs of components of type "Online Learning" are given to inputs of components of type "Threshold" and the outputs of components of type "State Estimation" are given to inputs of components of type "Feature". The combined objective of the semantic reasoning and configuration selection processes can be briefly described as: *Given the measured-known variables of the underline system, find the required types of components and the exact implementations of them that meet the pre-defined specifications*. In case of inability to find an appropriate fault-detection scheme configuration, the process terminates and informs accordingly. An illustrative use case is presented in the sequel, to clarify the processes.

5.3.4 Use Case: Water-Tank Contamination Event Detection

The use case is related to the problem of contamination event detection in drinking water distribution networks, exploiting the dynamics and measurements of chlorine concentrations, as approached in [32]. The underlying assumption is that contaminants injected in drinking water will react with and affect the concentration of chlorine [54]. For instance, a bacterial toxin may decrease the concentration of free chlorine. The low cost of sensors measuring chlorine concentration turns them appropriate for wide use by water utilities in the system monitoring for contamination events.

In most of the cases, the actual chlorine reaction dynamics are not known, resulting to the use of empirical models [56]. Chlorine concentration dynamics depend on a reaction rate coefficient, which in turn depends on several pipe parameters. Different studies have proposed various models based on laboratory experiments,

ranging from first-order linear models to more complex second-order ones.

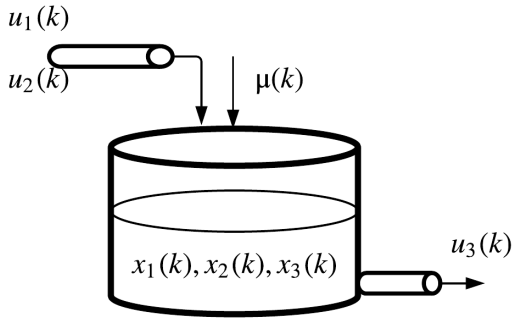


Figure 5.10: A water storage tank utilised in the illustrative use case

Consider a water storage tank of cylindrical shape in a water network, as depicted in Fig. 5.10. Water treated with chlorine is added into the tank from a pipe at the top and water is removed from the tank from a pipe situated at the bottom of the tank. Let k denote the discrete time with sampling interval $\Delta\tau$ measured in hours. The system state is denoted as $x(k) = [x_1(k), x_2(k), x_3(k)]^\top$ and the controlled inputs are denoted as $u(k) = [u_1(k), u_2(k)]^\top$, where: $x_1(k)$ is the tank volume in litres (L); $x_2(k)$ is the chlorine concentration of the tank water in mg/L; $x_3(k)$ is the concentration of a certain contaminant in mg/L; $u_1(k)$ is the volume of water entering the tank in litres (L), and is controlled through a valve; and $u_2(k)$ is the chlorine inflow in the tank measured in mg and is controlled using a set-point chlorination booster system. There is also an uncontrolled input $u_3(k)$ representing the volume of water exiting the tank, driven by consumer demands. The state-space formulation of the water tank system is as given in [32].

All components used in the use case, as well as the respective knowledge introduced in the Knowledge Graph $\mathcal{G}$ through their semantic annotation are listed in Table 5.1. For clarity of the concept's proof, only one measured system output (chlorine concentration) and only one known system input (chlorine inflow) are assumed; the inclusion of the rest can be addressed in a similar way.

Table 5.1: Components' list and semantic annotations at time $k_0 = 0$

Specifications
$\mathcal{S}$ = {detection signal as m_1: 'true-or-false', at location l_1: 'tank-1', for the property p_3: 'chlorine-concentration' }

	Detection Logic	
f_1^d	Inputs:	$\lambda(r_1) = \{\mathcal{L}, \mathcal{Q}, \text{feature}, \mathcal{M}\}$
		$\lambda(t_1) = \{\mathcal{L}, \mathcal{Q}, \text{threshold}, \mathcal{M}\}$
	Outputs:	$\lambda(d_1) = \{\mathcal{L}, \text{binary}, \text{detection}, \text{true/false}\}$
f_2^d	Inputs:	$\lambda(r_2) = \{\mathcal{L}, \mathcal{Q}, \text{feature}, \mathcal{M}\}$
		$\lambda(t_2) = \{\mathcal{L}, \mathcal{Q}, \text{threshold}, \mathcal{M}\}$
	Outputs:	$\lambda(d_2) = \{\mathcal{L}, \text{probability}, \text{detection}, [0,1]\}$
	Threshold	
f_1^t	Parameters:	$\lambda(\zeta_1) = \{\mathcal{L}, \mathcal{Q}, \text{histStateData}, \mathcal{M}\}$
	Outputs:	$\lambda(t_3) = \{\mathcal{L}, \text{constantBound}, \text{threshold}, \mathcal{M}\}$
f_2^t	Inputs:	$\lambda(y_3) = \{\text{tank-1}, \text{chlConcentration}, \text{stateMeasurement}, \text{mg/L}\}$
		$\lambda(u_1) = \{\text{tank-1}, \text{chlorineInflow}, \text{knownSystemInput}, \text{mg/L}\}$
	(optional)	$\lambda(g_1) = \{\mathcal{L}, \mathcal{Q}, \text{systemDynamics}, \mathcal{M}\}$
	Outputs:	$\lambda(t_4) = \{\mathcal{L}, \mathcal{Q}, \text{threshold}, \text{mg/L}\}$
	Feature	
f_1^r	Inputs:	$\lambda(y_1) = \{\mathcal{L}, \mathcal{Q}, \text{stateMeasurement}, \mathcal{M}\}$
	Outputs:	$\lambda(r_3) = \{\mathcal{L}, \mathcal{Q}, \text{feature}, \mathcal{M}\}$
f_2^r	Inputs:	$\lambda(y_2) = \{\text{tank1}, \text{chlConcentration}, \text{stateMeasurement}, \text{mg/L}\}$
		$\lambda(\hat{x}_1) = \{\text{tank1}, \text{chlConcentration}, \text{stateEstimation}, \mathcal{M}\}$
	Outputs:	$\lambda(r_4) = \{\mathcal{L}, \mathcal{Q}, \text{feature}, \text{mg/L}\}$
	State Estimation	
f_1^e	Inputs:	$\lambda(u_2) = \{\text{tank1}, \text{chlorineInflow}, \text{knownSystemInput}, \text{mg/L}\}$
		$\lambda(g_2) = \{\mathcal{L}, \mathcal{Q}, \text{systemDynamics}, \mathcal{M}$
	Outputs:	$\lambda(\hat{x}_2) = \{\text{tank1}, \text{chlConcentration}, \text{stateEstimation}, \text{mg/L}\}$
	Online Learning	
f_1^θ	Inputs:	$\lambda(y_4) = \{\text{tank1}, \text{chlConcentration}, \text{stateMeasurement}, \text{mg/L}\}$
		$\lambda(u_3) = \{\text{tank1}, \text{chlorineInflow}, \text{knownSystemInput}, \text{mg/L}$
	Outputs:	$\lambda(\hat{g}_1) = \{\text{tank1}, \text{chlConcentration}, \text{systemDynamics}, \text{mg/L}\}$
	Parameter Functions	
-		

Initially, the "State Estimation" and "Online Learning" components are not available. The Supervisor Σ selects the "Detection logic" component f_1^d because the component f_2^d violates the specification regarding the output in units 'true/false'. Moreover, the "Feature" component f_1^r is selected since the semantic annotation of its output matches with the semantic annotation of the input of the selected "Detection Logic" component, while the semantic annotation of its input is matched with the measured system state. The component f_2^r cannot be used since, although it matches the input of the "Detection Logic" component, there is no component producing an output of type 'estimatedState' which it requires as input. Subsequently, the "Threshold" component f_2^t is selected since it produces an output that matches with the respective input of the "Detection Logic" component and its two inputs are matched by measured and known variables of the system. The input that corresponds to the system dynamics function is not matched since there is no component producing a relevant output, however, it is mentioned as optional. At this stage, a possible configuration of the fault-detection scheme has been found, with component f_1^d producing the detection signal and receiving inputs from components f_1^r and f_2^t. It can be seen that all specifications are met by this configuration. Therefore, the configuration is confirmed and the decision signal σ_0 is generated.

At time $k_1 > k_0$, two more components become available in the database, one "State Estimation" and one "Online Learning", as listed in the respective rows of Table 5.1. The availability of the new components causes the re-execution of the Semantic Reasoning and Configuration Selection processes and produces the following results:

The "Detection logic" component f_1^d is selected again for the same reason as in the previous configuration. The "Feature" component f_2^r is selected since it produces an output that matches the input of the selected "Detection Logic" component and its inputs are now both matched, the first with the measured system state (as for the f_1^r) and the second with the output of the "State Estimation" component. Then, the same "Threshold" component f_2^t is selected as before. This time its input that corresponds to the system dynamics function is also matched by the output of the "Online Learning" component that became available. The "Online Learning" component f_1^θ is finally selected as well. It has been seen above that its output matches with the input of the selected "State Estimation" component, while both its inputs are matched by the measured system output and the known system input

respectively. A possible configuration of the fault-detection scheme comprises the component f_1^d producing the detection signal and receiving inputs from components f_2^r and f_2^t, while the component f_2^r and f_2^t receive input from the component f_1^e and the latter receives input from the component f_1^θ. All specifications are met by this configuration, as well, therefore it is confirmed and the decision signal σ_1 is generated.

5.3.5 Remarks

The work presented in this Section was an effort to automate the design process of a fault-detection scheme. The expert engineering and domain knowledge have been modelled in SEMIoTICS "Knowledge Graph" $\mathcal{G}$ and the semantically-enhanced Supervisor Σ was customised to be able to reproduce part of the cognitive process and reasoning performed by the engineer when designing the fault-detection scheme. The primary impact of these results is the fact that clear semantic interfaces have been defined between parts of the fault-detection scheme, which enables industrial set-ups and/or academic prototypes to allow online plugging-in of new implementations of components without the need to re-design the whole of the fault-detection scheme. The automation of the selection process in case of multiple matching configurations has been left out of this proof of concept, however, it can be performed based on quality-related criteria about the components.

Chapter 6

Conclusions and Future Work

The present Dissertation presented SEMIoTICS, a novel control system architecture, which enables the utilization of logic-based reasoning over declarative language models of IoT-enabled control system components, for the online re-configuration of feedback control systems. We showed that the system is able to extract configuration options online and implement some logic to evaluate them against pre-defined cost criteria so as to choose the best performing option for the operation of the feedback control system.

The applicability of the results has been tested with application use-cases and simulations from the smart buildings domain, as well as with smaller use-cases from the domains of Water Distribution Networks and Electric Power Grids. The use of such systems in large-scale buildings can considerably increase the flexibility of adding IoT components in control loops, either through physical installations or through downloading/importing software functions. It reduces the need for human intervention to components' changes and potentially increases the operation lifetime of a feedback control system. It has been shown that SEMIoTICS is able to respond to changes in the measurement, analysis and/or actuation capability and switch to a different (semantically valid) feedback control configuration. The key advantages are summarised as: i) unlike typical switching control systems, there is no pre-requisite for the system to know in advance the instances of the components or their capabilities. Our solution requires that the components syntactically adhere to the models presented in Section 2.2. It also requires that the end-points of the components be semantically described (online) in accordance with the semantic annotation operation defined in Section 3.3.2. As long as these two hold, the implementation

details of the components do not need to be known to SEMIoTICS in advance; ii) The switching decision is made online, by the Supervisor Σ, which is equipped with a logic-based system that exploits expert knowledge and deductive inference rules and helps producing an explicit decision signal. The expert knowledge comprises knowledge about the application domain, standards-compatible modelling of components, and feedback control system configuration rules; Current related work in literature, either does not address the re-configuration challenge at the closed-loop level or it does not provide an explicit mechanism for making the switching decisions when the configurations are not known in advance. For instance, existing solutions would neither have been able to feed a controller with the signal of a new sensor online nor to take advantage of a signal processing function online.

The provided solution has the following application limitations: i) although in theory the method works with wired components as well (for instance when certain standard communication protocols are employed, e.g., the BACnet/IP protocol [1] which facilitates plugging in of new components), its applicability is more straightforward with wireless IoT-enabled components; ii) new components deployed in the system must first be semantically described using the pre-defined models, otherwise they cannot be used effectively; iii) the logic-based switching decision-making is computationally intensive (see Section 3.4), therefore, it may not be applicable to systems with very fast dynamics, where the re-configuration decisions take more time than the system can accommodate without compromising the stability characteristics; iv) the system can only use types of components already considered in the modelling, i.e., sensors, actuators, controllers, pre- and post-control processing functions, and parameter functions; v) the supported measurement and actuation capability is limited by the modelled domain knowledge, i.e., the system cannot use an occupancy sensor if the property "occupancy" is not already described in the model, however, it provides online means of adding this new knowledge.

SEMIoTICS architecture and system has been tested through illustrative scenarios from the smart buildings domain, however, it is potentially applicable to a range of domains where IoT-enabled feedback control loops can be considered. It is clarified, however, that the application of SEMIoTICS in different domains requires the prior update of the knowledge model (not the core schema) so as to consider the domain features-of-interest and their properties.

Another studied aspect was the scalability characteristics of SEMIoTICS. The

Semantic Reasoning process that helps making the re-configuration decision is a combinatorial problem; it involves searching through combinations of linguistic variables' values in a graph, to satisfy certain logical constraints, which comes with increased computational cost and time. Although improvements can be achieved by combining the implementation with appropriate scaling and parallelization of computing resources, the solution may not be applicable to systems that cannot tolerate the re-configuration cost (in terms of time and computational load). We run certain experiments that help us derive important conclusions about the scalability characteristics of the solution. It appears that SEMIoTICS can be used in large-scale multi-zone buildings with big numbers of components without significant performance issues. However, there needs to be an upper bound to the number of configuration options SEMIoTICS examines, since high connectivity between components results in prohibitively low performance.

Finally, we went further into exploiting the plug-and-play features of SEMIoTICS for the automatic synthesis and online plugging of components in the smart buildings domain. Specifically, we presented the design of a distributed feedback linearization controllers' scheme for the heating control in a multi-zone building, where the synthesis of each zones' heating model is performed online by passing the required parameters to the respective controller. This enables SEMIoTICS to use the model-based feedback linearization controllers in a plug-and-play way. Furthermore, we presented a second design of the aforementioned control scheme that solves an online optimization problem and allocates the control signal within a zone, in an optimal way in terms of pre-defined quality-of-service criteria. This enables certain types of heating devices to be plugged in a zone and play for the heating control. It is shown that the solution is able to exploit the available components and the knowledge about the plant to make much more informed decisions, aiming to improve the occupants' comfort, as well as reduce the respective energy consumption costs.

We are currently working on applying SEMIoTICS in lab-based test-bests so as to demonstrate its plug-and-play features more effectively.

Bibliography

[1] BACnet/IP Tutorial. Accessed: 2016-10-21. [Online]. Available: http://www.bacnet.org/Tutorial/BACnetIP/

[2] "SWI Prolog," 1987. [Online]. Available: http://www.swi-prolog.org/

[3] A. Abur and A. G. Exposito, *Power system state estimation: Theory and Implementation*. New York: Basel: CRC Press, 2004.

[4] G. Acampora and V. Loia, "Fuzzy control interoperability and scalability for adaptive domotic framework," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 97–111, May 2005.

[5] C. Alippi, *Intelligence for Embedded Systems: A Methodological Approach*. Cham: Springer International Publishing, 2014, ch. Fault Diagnosis Systems, pp. 249–270.

[6] P. J. Antsaklis, B. Goodwine, V. Gupta, M. McCourt, Y. Po Wu, M. Xia, H. Yu, and Z. Feng, "Control of cyberphysical systems using passivity and dissipativity based methods," *Eur. J. Control*, vol. 19, no. 5, pp. 379–388, 2013.

[7] M. Asprou and E. Kyriakides, "Enhancement of hybrid state estimation using pseudo flow measurements," in *Power and Energy Society General Meeting, 2011 IEEE*, Detroit, USA, 2011, pp. 1–7.

[8] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Prentice Hall, 1994.

[9] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., *The Description Logic Handbook*. Cambridge University Press, 2003.

[10] M. Basseville and I. V. Nikiforov, *Detection of abrupt changes: theory and application*. Prentice-Hall, Inc., April 1, 1993.

[11] J. Bendtsen, K. Trangbaek, and J. Stoustrup, "Plug-and-Play Control—Modifying Control Systems Online," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 1, pp. 79–93, 2013.

[12] T. S. Bi, X. H. Qin, and Q. X. Yang, "A novel hybrid state estimator for including synchronized phasor measurements," *Electric Power Systems Research*, vol. 78, no. 8, pp. 2452–2458, 2009.

[13] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and fault-tolerant control*. Springer Verlag, 2003.

[14] M. Boasson, "Control systems software," *IEEE Trans. Autom. Control*, vol. 38, no. 7, pp. 1094–1106, 1993.

[15] D. Bonino and F. Corno, *DogOnt - Ontology Modeling for Intelligent Domotic Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–803. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88564-1_51

[16] M. A. Brdys and B. Ulanicki, *Operational control of water systems: structures, algorithms, and applications*. New York, USA: Prentice Hall, 1994.

[17] J. R. Cannon, *The One–Dimensional Heat Equation*, 1st ed., ser. 23. Cambridge–New York–New Rochelle–Melbourne–Sidney: Addison-Wesley Publishing Company/Cambridge University Press, 1984.

[18] N. Cauchi and A. Abate, "Benchmarks for cyber-physical systems: A modular model library for building automation systems," in *IFAC Conference on Analysis and Design of Hybrid Systems 2018*, 07 2018.

[19] S. Chakrabarti, E. Kyriakides, G. Ledwich, and A. Ghosh, "On the inclusion of phasor measurements in a power state estimation," *IET Generation, Transmission, and Distribution*, vol. 4, no. 10, pp. 1104–1115, 2010.

[20] S. Chakrabarti, E. Kyriakides, G. Valverde, and V. Terzija, "State estimation including synchronized measurements," in *Power Tech Conf.*, Bucharest, 2009, pp. 1–5.

[21] I. Chatzigiannakis, H. Hasemann, M. Karnstedt, O. Kleine, A. Kröller, M. Leggieri, D. Pfisterer, K. Römer, and C. Truong, "True self-configuration for the IoT," in *2012 3rd IEEE Int. Conf. on the Internet of Things*, Oct 2012, pp. 9–15.

[22] J. Chen and R. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[23] D. Conover, D. Crawley, S. Hagan, D. Knight, C. Barnaby, C. Gulledge, R. Hitchcock, S. Rosen, B. Emtman, G. Holness, D. Iverson, M. Palmer, and C. Wilkins, *An Introduction to Building Information Modeling (BIM) - A Guide for ASHRAE Members*. Amer. Soc. of Heating, Refrig. and Air-Cond. Eng., 2009.

[24] B. Copigneaux, S. Engell, R. Paulen, M. Reniers, C. Sonntag, and H. Thompson, "Proposal of a European Research and Innovation Agenda on Cyber-physical Systems of Systems – 2016-2025," CPSoS EU FP7-ICT Project (contract no. 611115) Consortium, Tech. Rep., 04 2016.

[25] S. Datta, R. D. Costa, and C. Bonnet, "Resource discovery in internet of things: Current trends and future standardization aspects," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 542–547.

[26] W. A. David Hilbert, Ed., *Grundzuge der theoretischen Logik*. Springer, Berlin, 1928, (2nd edition 1937). Reprint London, Allen and Unwin, 1948.

[27] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building Operating System Services," in *Proc. 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[28] B. L. V. der Waerden, Ed., *Moderne Algebra*. Springer, Berlin, 1930.

[29] D. G. Eliades and M. M. Polycarpou, "A fault diagnosis and security framework for water systems," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 6, pp. 1254 –1265, Nov. 2010.

[30] ——, "Water contamination impact evaluation and source-area isolation using decision trees," *ASCE Journal of Water Resources Planning and Management*, 2011, (available online).

[31] ——, "Leakage fault detection in district metered areas of water distribution systems," *Journal of Hydroinformatics*, vol. 14, no. 4, pp. 992–1005, 2012.

[32] D. Eliades, C. Panayiotou, and M. Polycarpou, "Contamination event detection in drinking water systems using a real-time learning approach," in *2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 663–670.

[33] D. Erbs, W. Beckman, and S. Klein, "Estimation of degree-days and ambient temperature bin data from monthly-average temperatures," *ASHRAE J.; (United States)*, vol. 25:6, 06 1983.

[34] D. Evans, "The Internet of Things - How the Next Evolution of the Internet Is Changing Everything," CISCO, Tech. Rep., 04 2011.

[35] J. Farrell and M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*, N. J. W. Hoboken, Ed. J. Wiley, 2006.

[36] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Distributed spatial indexing for the internet of things data management," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 1246–1251.

[37] R. M. G. Ferrari, H. Dibowski, and S. Baldi, "A message passing algorithm for automatic synthesis of probabilistic fault detectors from building automation ontologies," in *20th IFAC World Congress*, Jul 2017, pp. 4268–4274.

[38] R. M. G. Ferrari, T. Parisini, and M. M. Polycarpou, "Distributed fault detection and isolation of large-scale discrete-time nonlinear systems: An adaptive approximation approach," *IEEE Trans. Autom. Control*, vol. 57, no. 2, pp. 275–290, 2012.

[39] J. Ferreiros, "The road to modern logic-an interpretation," *Bulletin of Symbolic Logic*, vol. 7, no. 4, pp. 441–484, 12 2001. [Online]. Available: http://projecteuclid.org/euclid.bsl/1182353823

[40] M. Fliess and C. Join, "Model-free control," *International Journal of Control*, vol. 86, no. 12, pp. 2228–2252, 2013. [Online]. Available: https://doi.org/10.1080/00207179.2013.810345

[41] Fred Dushin, "JPL - A Java Interface to Prolog," 2003. [Online]. Available: http://www.swi-prolog.org/packages/jpl/java_api/

[42] F. Ganz, P. Barnaghi, and F. Carrez, "Automated semantic knowledge acquisition from sensor data," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1214–1225, Sept 2016.

[43] GeoSPARQL - A Geographic Query Language for RDF Data. Accessed: 2017-07-24. [Online]. Available: http://www.opengeospatial.org/standards/geosparql

[44] J. Gertler, "Analytical redundancy methods in fault detection and isolation," in *Preprints of IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS'91*, 1991, pp. 9–21.

[45] P. Glotfelter, T. Eichelberger, and P. J. Martin, "PhysiCloud : A Cloud-Computing Framework for Programming Cyber-Physical Systems," in *2014 IEEE Conf. on Control Applications (CCA)*, 2014.

[46] D. J. Glover, M. S. Sarma, and T. J. Overbye, "Power System Controls," in *Power Systems Analysis and Design*, 5th ed., S. Meherishi and T. Altieri, Eds. Stanford, USA: Global Engineering: Christopher M. Shortt, 2012, pp. 639–689.

[47] K. Godel, "Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i," *Monatshefte für Math. u. Physik*, vol. 38, pp. 173–198, 1931.

[48] W. Goetzler, "Variable Refrigerant Flow Systems," *ASHRAE Journal*, 2017. [Online]. Available: http://www.msidata.com/assets/VRF-explanation.pdf

[49] A. Gómez-Pérez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition*, 1st ed. London, U.K.: Springer-Verlag London, 2004.

[50] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hoffmeister, "Towards a semantic administrative shell for industry 4.0 components," in *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, Feb 2016, pp. 230–237.

[51] W. M. Grayman, "A Quarter of a Century of Water Quality Modeling in Distribution Systems," in *Proc. ASCE Water Distribution Systems Analysis*, 2006, p. 12.

[52] C. Griffiths and M. Costi, *GRASP : the solution*. Cardiff, UK: Proactive Press, 2011.

[53] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *2010 Internet of Things (IOT)*, Nov 2010, pp. 1–8.

[54] J. Hall, A. Zaffiro, R. Marx, P. Kefauver, R. Krishman, R. Haught, and J. Herrmann, "On-line water quality parameters as indicators of distribution system," *Journal of the American Water Works Association*, vol. 99, no. 1, pp. 66–77, 2007.

[55] A. Haller, K. Janowicz, S. Cox, D. L. Phuoc, K. Taylor, M. Lefrançois, R. Atkinson, R. García-Castro, J. Lieberman, and C. Stadler. Semantic Sensor Network Ontology. Accessed: 2017-07-24. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[56] F. Hua, J. West, R. Barker, and C. Forster, "Modelling of chlorine decay in municipal water supplies," *Water Research*, vol. 33, no. 12, pp. 2735–2746, Aug. 1999.

[57] J. Huang, F. Bastani, I. L. Yen, J. Dong, W. Zhang, F. J. Wang, and H. J. Hsu, "Extending service model to build an effective service composition framework for cyber-physical systems," in *IEEE Int. Conf. on Service-Oriented Computing and Applications, SOCA' 09*, 2009.

[58] J. Y. Hung, W. Gao, and J. C. Hung, "Variable structure control: a survey," *IEEE Trans. Ind. Electron.*, vol. 40, no. 1, pp. 2–22, Feb 1993.

[59] I. Hwang, S. Kim, Y. Kim, and C. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 3, pp. 636–653, May 2010.

[60] P. P.-S. I. Horrocks, "Reducing owl entailment to description logic satisfiability," In Proc. of the 2nd International Semantic Web Conference (ISWC), 2003, http://www.cs.man.ac.uk/ horrocks/Publications/download/2003/HoPa03c.

[61] *1855-2016 — IEEE Standard for Fuzzy Markup Language*, IEEE Std., 2016.

[62] IEEE SmartGrid. Accessed: 2016-10-21. [Online]. Available: http://smartgrid.ieee.org/

[63] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[64] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. New York, NY, USA: Springer-Verlag, 2006.

[65] ——, "Supervision, fault-detection and fault-diagnosis methods—an introduction," *Control engineering practice*, vol. 5, no. 5, pp. 639–652, 1997.

[66] C. Joslyn, "Semantic control systems," *World Futures: Journal of General Evolution*, vol. 45, no. 1-4, pp. 87–123, 1995.

[67] D. Jung and A. Savvides, "Estimating Building Consumption Breakdowns using ON/OFF State Sensing and Incremental Sub-Meter Deployment," in *8th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2010.

[68] B. Kassaie, "SPARQL over graphx," *CoRR*, vol. abs/1701.03091, 2017. [Online]. Available: http://arxiv.org/abs/1701.03091

[69] T. Knudsen, "Awareness and its use in Plug and Play Process Control," in *2009 European Control Conf. (ECC)*, Aug 2009, pp. 4078–4083.

[70] K. Kotis and A. Katasonov, "Semantic interoperability on the internet of things: The semantic smart gateway framework," *Int. J. Distrib. Syst. Technol.*, vol. 4, no. 3, pp. 47–69, Jul. 2013. [Online]. Available: http://dx.doi.org/10.4018/jdst.2013070104

[71] A. Krioukov, G. Fierro, N. Kitaev, and D. Culler, "Building application stack (BAS)," in *Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings - BuildSys '12*. New York, USA: ACM Press,

2012, p. 72. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2422531.2422546

[72] Y. Lirov, E. Y. Rodin, B. G. McElhaney, and L. W. Wilbur, "Artificial Intelligence Modelling of Control Systems," *The Society for Modeling and Simulation International*, vol. 50, no. 1, pp. 12–24, Jan. 1988. [Online]. Available: http://dx.doi.org/10.1177/003754978805000103

[73] X. Liu, B. Cheng, J. Liao, P. Barnaghi, L. Wan, and J. Wang, "Omi-dl: An ontology matching framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 580–593, July 2016.

[74] J. Lygeros, C. Tomlin, and S. Sastry, *Hybrid Systems: Modeling, Analysis and Control*. Univeristy of California, Barkeley, 2008.

[75] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. (2004) OWL-S: Semantic Markup for Web Services. Accessed: 2017-07-24. [Online]. Available: https://www.w3.org/Submission/OWL-S/

[76] G. Milis, D. Eliades, C. Panayiotou, and M. Polycarpou, "A Cognitive Fault-Detection Design Architecture," in *IJCNN, World Congress in Computational Intelligence*, 2016.

[77] G. Milis, C. Panayiotou, and M. Polycarpou, "Semantically-Enhanced Online Configuration of Feedback Control Schemes," *IEEE Trans. on Cybernetics*, vol. 48, no. 3, pp. 1081–1094, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7891022/

[78] G. M. Milis, M. Asprou, E. Kyriakides, C. G. Panayiotou, and M. M. Polycarpou, "Semantically-enhanced configurability in state estimation structures of power systems," in *2015 IEEE Symposium Series on Computational Intelligence*, Dec 2015, pp. 679–686.

[79] G. M. Milis, D. G. Eliades, C. G. Panayiotou, and M. M. Polycarpou, "Semantic mediation in smart water networks," in *2015 IEEE Symposium Series on Computational Intelligence*, Dec 2015, pp. 617–624.

[80] G. M. Milis, C. G. Panayiotou, and M. M. Polycarpou, "Semiotics: Semantically-enhanced iot-enabled intelligent control systems," *IEEE Internet of Things Journal*, 2017.

[81] Z. Ming, V. A. Centeno, J. S. Thorp, and A. G. Phadke, "An alternative for including phasor measurements in state estimators," *IEEE Transactions on Power Systems*, vol. 21, no. 4, pp. 1930–1937, 2006.

[82] G. E. Moore, "A Defence of Common Sense," in *Contemporary British Philosophy (2nd series)*, J. H. Muirhead, Ed. Allen and Unwin, London, 1925, pp. 192–233.

[83] Machine-to-machine Internet-of-Things Connectivity Protocol. Accessed: 2017-01-25. [Online]. Available: http://mqtt.org/

[84] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar, *Context and Semantic Composition of Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 266–275. [Online]. Available: http://dx.doi.org/10.1007/11827405_26

[85] J. V. Neumann, "Eine axiomatisierung der mengenlehre," *Journal fur die reine- und angewandte Mathematik*, vol. 154, pp. 219–240, 1925, reprint in Collected Works, vol. 1, Oxford, Pergamon, 1961.

[86] A. S. of Heating and A.-C. E. (ASHRAE), "STANDARD 55 – THERMAL ENVIRONMENTAL CONDITIONS FOR HUMAN OCCUPANCY," 2013. [Online]. Available: https://www.ashrae.org/technical-resources/bookstore/standard-55-thermal-environmental-conditions-for-human-occupancy

[87] A. Pablo, R. Valiente, and A. Lozano-Tello, "Ontology and SWRL-Based Learning Model for Home Automation Controlling," in *Ambient Intelligence and Future Trends-Int. Symposium on Ambient Intelligence (ISAmI 2010)*, J. C. Augusto, J. M. Corchado, P. Novais, and C. Analide, Eds. Berlin: Springer Berlin Heidelberg, 2010, pp. 79–86.

[88] P. M. Papadopoulos, V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Distributed diagnosis of actuator and sensor faults in hvac systems," in *20th IFAC World Congress*, D. Dochain, D. Henrion, and D. Peaucelle, Eds., vol. 50, no. 1. IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd, Jule 2017, pp. 4209–4215.

[89] P. Pauwels and D. Van Deursen, "IFC-to-RDF: adaptation, aggregation and enrichment," in *First Int. Workshop on Linked Data in Architecture and Construction, Abstracts*, 2012, pp. 1–3. [Online]. Available: http://multimedialab.elis.ugent.be/ldac2012/documents/LDACworkshopreport.pdf

[90] C. Petrie, *Web Service Composition*. Springer International Publishing, Switzerland, 2016.

[91] A. G. Phadke and J. S. Thorp, *Synchronized phasor measurements and their applications*. New York: Springer, 2008.

[92] J. Ploennigs, B. Hensel, H. Dibowski, and K. Kabitzsch, "Basont - a modular, adaptive building automation system ontology," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 4827–4833.

[93] M. M. Polycarpou and A. J. Helmi, "Automated fault detection and accommodation: a learning systems approach," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 25, no. 11, pp. 1447–1458, 1995.

[94] Q.Z.Sheng, X. Qiao, A. Vasilakos, C. S. ands S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Information Sciences*, vol. 280, pp. 218–238, Oct 2014.

[95] L. Z. Raja and R. A. Issa, "Ontology-based partial building information model extraction," *Journal of Computing in Civil Engineering*, vol. 27, no. 6, pp. 576–584, 2013.

[96] V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Multiple sensor fault detection and isolation for large-scale interconnected nonlinear systems," in *Proc. Eur. Control Conf.*, Zurich, Switzerland, 2013, pp. 1952–1957.

[97] ——, "Adaptive approximation for multiple sensor fault detection and isolation of nonlinear uncertain systems," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 137–153, 2014.

[98] V. Reppa and A. Tzes, "Fault detection and diagnosis based on parameter set estimation," *IET control theory & applications*, vol. 5, no. 1, pp. 69–83, 2011.

[99] RestApiTutorial.com, "Learn REST: A RESTful Tutorial." [Online]. Available: https://www.restapitutorial.com/

[100] S. Riverso, F. Boem, G. Ferrari-Trecate, and T. Parisini, "Plug-and-Play Fault Detection and Control-Reconfiguration for a Class of Nonlinear Large-Scale Constrained Systems," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3963–3978, Dec 2016.

[101] S. Robin, "Aristotle's Logic," in *The Stanford Encyclopedia of Philosophy*, winter 2016 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2016.

[102] E. Rodin, "Semantic Control Theory," *Applied Mathematics Letters*, vol. 1, no. 1, pp. 73–78, 1988.

[103] L. A. Rossman, *EPANET 2 Users manual*, EPA/600/R-00/057, National Risk Management Research Laboratory, Office of Research and Development, U.S. Environmental Protection Agency, Cincinnati, OH, Sep. 2000.

[104] B. Russel, "Mathematical logic as based on the theory of types," *American Journal of Mathematics*, vol. 30, pp. 222–262, 1908.

[105] B. Russell, Ed., *The principles of mathematics*. Cambridge University Press, 1903, (2nd edition 1937). Reprint London, Allen and Unwin, 1948.

[106] B. Russell, *Common Sense and Nuclear Warfare*, 1st ed. Routledge, Oxford, UK, May 2001.

[107] T. Samad, "Control systems and the internet of things [technical activities]," *IEEE Control Systems*, vol. 36, no. 1, pp. 13–16, Feb 2016.

[108] D. Savic, Z. Kapelan, and P. Jonkergouw, "Quo vadis water distribution model calibration?" *Urban Water Journal*, vol. 6, no. 1, pp. 3–22, Feb 2009.

[109] Sensor Model Language (SensorML). Accessed: 2016-10-21. [Online]. Available: http://www.opengeospatial.org/standards/sensorml

[110] T. Skolem, "Einige bemerkungen zur axiomatischen begriindung der mengenlehre," Dem Femte skandinaviska mathematikerkongressen, Akademiska Bokhandeln, 1923, helsinki.

[111] Stanford Center for Biomedical Informatics Research, "A free, open-source ontology editor and framework for building intelligent systems," 2016. [Online]. Available: https://protege.stanford.edu/

[112] J. Stoustrup, "Plug & Play Control: Control Technology Towards New Challenges," *European Journal of Control*, vol. 15, no. 3-4, pp. 311–330, Aug. 2009. [Online]. Available: http://ejc.revuesonline.com/article.jsp?articleId=13584

[113] K. Suri, W. Gaaloul, A. Cuccuru, and S. Gerard, "Semantic framework for internet of things-aware business process development," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, June 2017, pp. 214–219.

[114] SWI Prolog, "SWI-Prolog Semantic Web Library 3.0." [Online]. Available: http://www.swi-prolog.org/pldoc/doc_for?object=section(%27packages/semweb.html%27)

[115] The MathWorks, Inc., "MATLAB - MathWorks." [Online]. Available: https://www.mathworks.com/products/matlab.html

[116] W. A. H. Thissen and P. M. Herder, "System of Systems Perspectives on Infrastructures," in *System of Systems Engineering*, M. Jamshidi, Ed. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2008, ch. 11.

[117] E. Todini and S. Pilati, "A gradient method for the analysis of pipe networks," in *Proc. Computer Applications for Water Supply and Distribution*, Leicester, UK, 1987, p. 20.

[118] R. Toenjes, D. Kuemper, and M. Fischer, "Knowledge-based spatial reasoning for iot-enabled smart city applications," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Dec 2015, pp. 736–737.

[119] Top 5 Vendors in the Global Integrated Building Management Systems Market from 2017-2021: Technavio. Accessed: 2018-02-23. [Online]. Available: https://www.businesswire.com/news/home/20170106005181/en/Top-5-Vendors-Global-Integrated-Building-Management

[120] D. Trihinas, G. Pallis, and M. Dikaiakos, "Low-cost adaptive monitoring techniques for the internet of things," *IEEE Transactions on Services Computing*, 2018.

[121] W3C. (2004) Semantic Web Query Standards. Accessed: 2016-10-21. [Online]. Available: http://www.w3.org/standards/semanticweb/query

[122] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," *CoRR*, vol. abs/1804.05271, 2018.

[123] E. Witrant, S. Mocanu, and O. Sename, "A hybrid model and mimo control for intelligent buildings temperature regulation over wsn," in *IFAC Proceedings Volumes*, vol. 42, no. 14. Elsevier Ltd, 2009, pp. 420–425.

[124] A. Wood, B. Wollenberg, and G. Sheble, *Power Generation, Operation and Control*, 3rd ed. Wiley, 2013. [Online]. Available: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471790559.html

[125] World Wide Web Consortium (W3C), "PrimerExampleTurtle - OWL," 2008. [Online]. Available: https://www.w3.org/2007/OWL/wiki/PrimerExampleTurtle

[126] X. Zhang, M. M. Polycarpou, and T. Parisini, "A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 576–593, Apr. 2002.

[127] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.