International Journal on Advances in Systems and Measurements

2014 vol. 7 nr. 3&4

The *International Journal on Advances in Systems and Measurements* is published by IARIA.
ISSN: 1942-261x
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Systems and Measurements
Volume 7, Number 3 & 4, 2014

Mohamed Eltoweissy, Pacific Northwest National Laboratory / Virginia Tech, USA
Paulo Felisberto, LARSyS, University of Algarve, Portugal
Miguel Franklin de Castro, Federal University of Ceará, Brazil
Mounir Gaidi, Centre de Recherches et des Technologies de l'Energie (CRTEn), Tunisie
Eva Gescheidtova, Brno University of Technology, Czech Republic
Tejas R. Gandhi, Virtua Health-Marlton, USA
Teodor Ghetiu, University of York, UK
Franca Giannini, IMATI - Consiglio Nazionale delle Ricerche - Genova, Italy
Gonçalo Gomes, Nokia Siemens Networks, Portugal
Luis Gomes, Universidade Nova Lisboa, Portugal
Antonio Luis Gomes Valente, University of Trás-os-Montes and Alto Douro, Portugal
Diego Gonzalez Aguilera, University of Salamanca - Avila, Spain
Genady Grabarnik,CUNY - New York, USA
Craig Grimes, Nanjing University of Technology, PR China
Stefanos Gritzalis, University of the Aegean, Greece
Richard Gunstone, Bournemouth University, UK
Jianlin Guo, Mitsubishi Electric Research Laboratories, USA
Mohammad Hammoudeh, Manchester Metropolitan University, UK
Petr Hanáček, Brno University of Technology, Czech Republic
Go Hasegawa, Osaka University, Japan
Henning Heuer, Fraunhofer Institut Zerstörungsfreie Prüfverfahren (FhG-IZFP-D), Germany
Paloma R. Horche, Universidad Politécnica de Madrid, Spain
Vincent Huang, Ericsson Research, Sweden
Friedrich Hülsmann, Gottfried Wilhelm Leibniz Bibliothek - Hannover, Germany
Travis Humble, Oak Ridge National Laboratory, USA
Florentin Ipate, University of Pitesti, Romania
Imad Jawhar, United Arab Emirates University, UAE
Terje Jensen, Telenor Group Industrial Development, Norway
Liudi Jiang, University of Southampton, UK
Kenneth B. Kent, University of New Brunswick, Canada
Fotis Kerasiotis, University of Patras, Greece
Andrei Khrennikov, Linnaeus University, Sweden
Alexander Klaus, Fraunhofer Institute for Experimental Software Engineering (IESE), Germany
Andrew Kusiak, The University of Iowa, USA
Vladimir Laukhin, Institució Catalana de Recerca i Estudis Avançats (ICREA) / Institut de Ciencia de Materials de Barcelona (ICMAB-CSIC), Spain
Kevin Lee, Murdoch University, Australia
Andreas Löf, University of Waikato, New Zealand
Jerzy P. Lukaszewicz, Nicholas Copernicus University - Torun, Poland
Zoubir Mammeri, IRIT - Paul Sabatier University - Toulouse, France
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Stefano Mariani, Politecnico di Milano, Italy
Paulo Martins Pedro, Chaminade University, USA / Unicamp, Brazil
Don McNickle, University of Canterbury, New Zealand
Mahmoud Meribout, The Petroleum Institute - Abu Dhabi, UAE
Luca Mesin, Politecnico di Torino, Italy

Marco Mevius, HTWG Konstanz, Germany
Marek Miskowicz, AGH University of Science and Technology, Poland
Jean-Henry Morin, University of Geneva, Switzerland
Fabrice Mourlin, Paris 12th University, France
Adrian Muscat, University of Malta, Malta
Mahmuda Naznin, Bangladesh University of Engineering and Technology, Bangladesh
George Oikonomou, University of Bristol, UK
Arnaldo S. R. Oliveira, Universidade de Aveiro-DETI / Instituto de Telecomunicações, Portugal
Aida Omerovic, SINTEF ICT, Norway
Victor Ovchinnikov, Aalto University, Finland
Telhat Özdoğan, Recep Tayyip Erdogan University, Turkey
Gurkan Ozhan, Middle East Technical University, Turkey
Constantin Paleologu, University Politehnica of Bucharest, Romania
Matteo G A Paris, Universita` degli Studi di Milano,Italy
Vittorio M.N. Passaro, Politecnico di Bari, Italy
Giuseppe Patanè, CNR-IMATI, Italy
Marek Penhaker, VSB- Technical University of Ostrava, Czech Republic
Juho Perälä, VTT Technical Research Centre of Finland, Finland
Florian Pinel, T.J.Watson Research Center, IBM, USA
Ana-Catalina Plesa, German Aerospace Center, Germany
Miodrag Potkonjak, University of California - Los Angeles, USA
Alessandro Pozzebon, University of Siena, Italy
Vladimir Privman, Clarkson University, USA
Konandur Rajanna, Indian Institute of Science, India
Stefan Rass, Universität Klagenfurt, Austria
Candid Reig, University of Valencia, Spain
Teresa Restivo, University of Porto, Portugal
Leon Reznik, Rochester Institute of Technology, USA
Gerasimos Rigatos, Harper-Adams University College, UK
Luis Roa Oppliger, Universidad de Concepción, Chile
Ivan Rodero, Rutgers University - Piscataway, USA
Lorenzo Rubio Arjona, Universitat Politècnica de València, Spain
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Subhash Saini, NASA, USA
Mikko Sallinen, University of Oulu, Finland
Christian Schanes, Vienna University of Technology, Austria
Rainer Schönbein, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB), Germany
Guodong Shao, National Institute of Standards and Technology (NIST), USA
Dongwan Shin, New Mexico Tech, USA
Larisa Shwartz, T.J. Watson Research Center, IBM, USA
Simone Silvestri, University of Rome "La Sapienza", Italy
Diglio A. Simoni, RTI International, USA
Radosveta Sokullu, Ege University, Turkey
Junho Song, Sunnybrook Health Science Centre - Toronto, Canada
Leonel Sousa, INESC-ID/IST, TU-Lisbon, Portugal

International Journal on Advances in Systems and Measurements
Volume 7, Numbers 3 & 4, 2014

CONTENTS

Monitoring of Hazardous Scenarios using Multi-Sensor Devices and Sensor Data Fusion

Matthias Bartholmai, Enrico Koeppe, and Patrick P. Neumann
Sensors, Measurement and Testing Methods
BAM Federal Institute for Materials Research and Testing
Berlin, Germany
matthias.bartholmai@bam.de

***Abstract*— The combination of different types of sensors to multi-sensor devices offers excellent potential for monitoring applications. This should be demonstrated by means of four different examples of actual developments carried out by Federal Institute for Materials Research and Testing (BAM): monitoring and indoor localization of relief forces, a micro-drone for gas measurement in hazardous scenarios, sensor-enabled radio-frequency identification (RFID) tags for safeguard of dangerous goods, and a multifunctional sensor for spatially resolved under-surface monitoring of gas storage areas. Objective of the presented projects is to increase the personal and technical safety in hazardous scenarios. These examples should point to application specific challenges for the applied components and infrastructure, and it should emphasize the potential of multi-sensor systems and sensor data fusion.**

Keywords- monitoring, multi-sensor, hazardous scenarios, data fusion

I. INTRODUCTION

The safe operation in hazardous scenarios (conflagrations, chemical incidents, etc.) and handling of dangerous substances (toxic, explosive, harmful for human and/or the environment) often requires the usage of sensor systems, e.g., to measure the status of a process, to enable early warning in case of an accident, or to evaluate the situation after an accident happened [1]. In many cases not only one measuring variable is sufficient for a comprehensive evaluation of such scenarios, demanding for technical solutions with integration of multiple types of sensors. Technical enhancements like miniaturization, data processing, and wireless communication are the basis for application specific multi-sensor solutions. Data fusion offers sophisticated possibilities to analyze and clarify the hazard potential of relevant situations – in many cases quasi in real-time.

The following examples present multi-sensor concepts applied to different scenarios of condition monitoring and safety management. Often similar issues and requirements must be taken into account, regardless of whether the monitoring object is a firefighter, a cask for radioactive material or a subsurface storage area.

The paper is structured in 6 sections. The Sections II-V describe the above mentioned examples on basis of the physical principle, functionality and application. Section VI gives a short summary and the most relevant conclusions.

II. MONITORING AND INDOOR LOCALIZATION OF RELIEF FORCES

Rescue forces often operate in dangerous scenarios and situations, in which their localization can be crucial for safe operation and return. Fire, landslip-, or flood scenarios pose hazards like suffocation, burn, or undercooling. The localization and quick recovery raise the survival chance clearly. The use of Global Positioning system (GPS) technology allows the exact localization of persons or objects everywhere a sufficient satellite reception is possible. However, in many hazardous scenarios no or only insufficient GPS reception is available. This may be the case in underground, indoor, or fire scenarios, making GPS localization complicated or impossible.

A. Concept and Components

Objectives of the project "Localization and monitoring of relief forces in hazardous scenarios" with acronym OMEGa are the development and validation of a monitoring system, which complements GPS localization with indoor navigation [2] and in addition measures the most important vital functions. The overall system consists of two units, which operate spatially separated and communicate via radio with each other. The first unit are portable multi-sensor devices, which serve as personal protective equipment (PPE-Device) of the rescue force and should be implemented, e.g., by integration in the clothes. The second unit consists of the components of the control station for data processing and display (Figure 1).

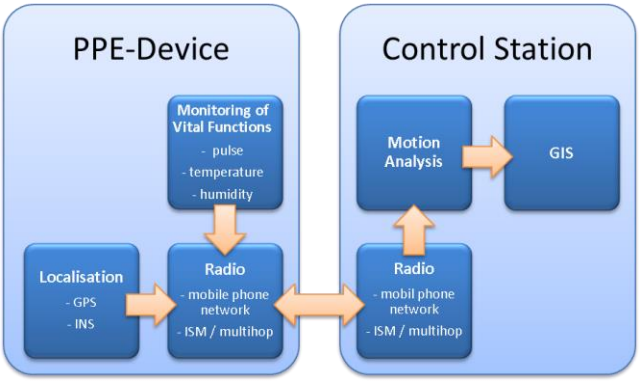


Figure 1. Scheme of the OMEGa units.

The multi-sensor device (Figure 2) should consist of an outdoor localization system (GPS), an inertial navigation system (INS) for indoor localization, and sensors for monitoring of vital functions like pulse, temperature and humidity at the body surface. The communication between both units should be implemented through a redundant solution of two radio modes, based on mobile phone network and ISM band, the latter with multihop routing. Principal elements of the control station are analysis tools for calculating motion sequences from the sensor data and a geographical information system (GIS) to track and monitor the equipped persons in map-based software.

Figure 2. Prototype of the OMEGa multi-sensor device.

Indoor localization on basis of an INS is the most sophisticated challenge in the OMEGa project. The INS itself is a multi-sensor microelectromechanical systems (MEMS) device consisting of 3-axes accelerometers, gyroscopes, magnetic field, and barometric pressure sensors, partly redundant. The calculation of motion sequences from the combined sensor data is performed by data fusion algorithms [2][3][4].

B. Results

Different motion sequences can be identified by analyzing the different sensor signals. In a series of experiments, persons were moving on a treadmill with different speed. The OMEGa device was placed at their central lower back in height of the hip. The type of movement (walking or running) and the speed lead to varying acceleration signals. Figure 3 displays the data for walking at speeds from 2 to 7 km/h and running at speeds from 8 to 12 km/h. The walking results show significant differences in length and time of single steps between putting down and lifting the feet. In contrast to walking, running results deliver similar step times, but the acceleration impulse differs for different speeds. This example shows how movement sequences can be characterized and identified by simple means of pattern recognition.

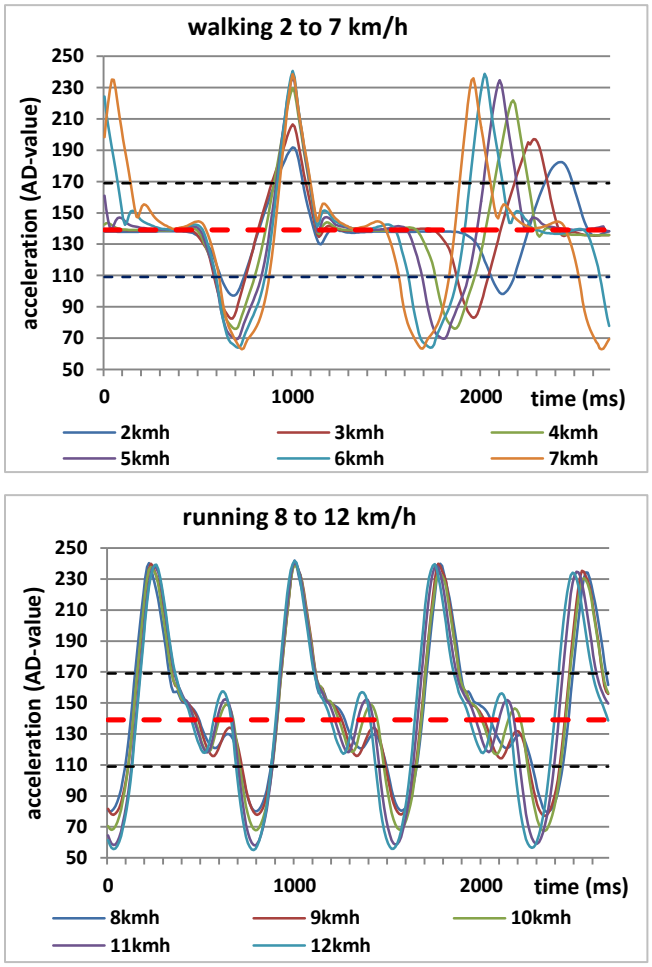


Figure 3. INS acceleration signals of different motion sequences.

The combination of these findings with the measurement of vital functions can be used to enable comprehensive monitoring of relief forces during operation. Further objectives are automated detection of critical situations and alarming.

Another result of the project was the implementation of a new calibration method for an INS. This principle is based on the free motion at the curved surface area of an ellipsoid, which allows free motion calibration of the sensor at any place or position [3][4]. In the same way, the algorithm can use the movement of the holder as input for a continuous recalibration during a normal operation. By moving the sensor system in a pseudo static motion, measurement data is generated and used to determine the ellipsoid. This geometrical figure describes the sensor idle state and amplitude at a known measurement value. An optimisation function was implemented in the algorithm to gain the ellipsoid out of noisy measurements. Furthermore, the advantage of this principle is that it is possible to calibrate a

free motion of the sensor system at any place or position on a person. In other words, the sensor system is calibrated and adjusted during normal operation. Hence, there are no more movements after the activation of the system or during the working process necessary for the calibration [4].

III. MICRO-DRONE FOR GAS MEASUREMENT IN HAZARDOUS SCENARIOS

A research project was carried out at BAM with the objective to develop a flying remote-controlled measuring system. The system is capable of operating in a variety of scenarios of gas emission, e.g., exhaust gas from a chimney, flue gas in case of a fire, gas emission in case of an accident of chemical or hazardous goods [5]. Another addressed field of application is spatially resolved emission control of geodynamic active regions, waste disposals, stockpiles, landfills, CO_2 storage areas (carbon capture and storage, CCS), industrial sites and pollution critical areas. Due to its mobility the system can measure the gas concentration in the immediate vicinity of the object, which causes the emission. A further stage of extension is the enhancement of the system for identification of gas source locations, plume tracking, and gas distribution modeling/mapping (GDM). The latter applications are implemented based on the combined analysis of position dependent gas concentrations and wind vector data.

Figure 4. Micro-drone with multi-sensor equippment in flight.

Gas concentration measurement from an air-borne platform (AR 100-B, Airrobot, Germany; see Figure 4) is demanding in terms of weight, dimensions, energy consumption, influence of the rotors, and speed of the sensing device. A gas-sensing payload was developed on basis of a commercially available gas detector (X-am 5600, Draeger, Germany), which was originally designed as personal safety equipment. The device features low weight and compact design. The modular concept allows the ad hoc exchange of four sensors in the gas detector, which enables users to customize it for their specific application.

Due to the weight restrictions imposed by the platform (max. payload 200 g), the micro-drone does not carry any wind sensing modalities. Instead, wind measurements are estimated by fusing the different on-board sensors of its inertial measurement unit to compute the parameters of the wind triangle [6]. The wind triangle is commonly used in navigation and describes the relationships between the flight vector, the ground vector, and the wind vector. The micro-drone can be operated manually or in GPS mode, e.g., by autonomous waypoint following.

A. Plume Tracking Algorithms

Both, gas distribution modeling and plume-tracking were enabled using data fusion algorithms. For plume tracking three promising algorithms were implemented and adapted accordingly to meet the system characteristics of the micro-drone: the surge-cast algorithm (a variant of the silkworm moth algorithm), the zigzag/dung beetle algorithm, and a newly developed algorithm called “pseudo gradient-based algorithm”. First successful tests were performed in real-world experiments [7][8].

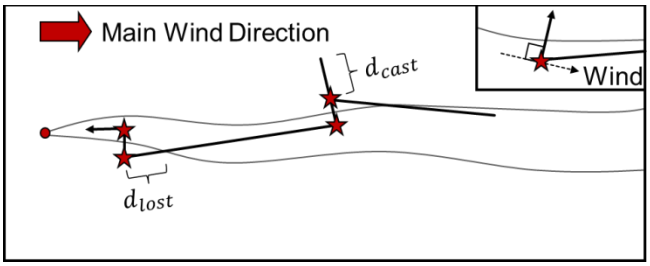


Figure 5. Surge-cast algorithm.

Lochmatter presented in [9] the surge-cast algorithm. It is a combination of plume tracking strategies used by the silkworm moth and works as follows (Figure 5): The robot moves straight upwind until it loses the contact with the plume for a certain distance d_{lost}. Then, it tries to reacquire the plume by searching crosswind for a defined distance d_{cast} on both sides. The chance of reacquiring the plume in the first crosswind movement is maximized by measuring the wind direction to estimate the side, from which the robot has left the plume. Every time the robot switches its behavior from upwind surge to casting and vice versa, the wind direction is re-measured. In comparison to the original algorithm, the plume is declared lost in the surge-cast algorithm used here, when the micro-drone measures an average gas concentration below the threshold after one step. To reacquire the plume, casting with increasing step size in crosswind direction is performed. These changes were necessary to address the constraints of the micro-drone in GPS-mode. Furthermore, the wind is re-measured every iteration of the algorithm to adapt faster to changing wind conditions. If casting fails to reacquire the plume (after a defined number of steps) the micro-drone returns to the sweeping strategy.

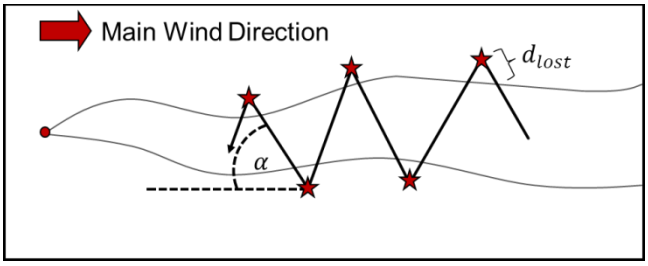


Figure 6. Zigzag or dung beetle algorithm.

The zigzag or dung beetle algorithm was first reported by Ishida et al. [10]. The basic algorithm works as follows (Figure 6): The robot moves upwind with an angle α (e.g., α = 60◦) across the plume constantly sensing gas concentrations. If the gas sensor measures a concentration below a given threshold, the robot is assumed to have reached the edge of the plume. It re-measures the wind direction and continues moving upwind with an angle -α with respect to the upwind direction. This procedure is repeated causing the robot to move in a zigzag fashion within the plume. The robot is stopped, when it has reached the source. In comparison to the original algorithm, the micro-drone does only collect gas and wind measurements at the waypoints where it stops.

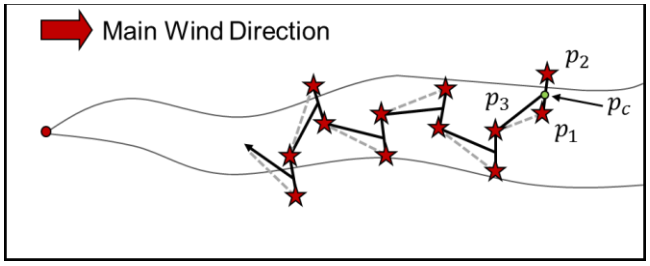


Figure 7. Pseudo gradient-based algorithm.

The idea for the first gradient-based algorithms for plume tracking goes back to Braitenberg [11]. The chemical gradient is measured by a pair of bilateral gas sensors mounted on each side of a robot, each directly controlling the speed of a wheel. Each sensor is connected to the motor on the same side, the motor on the opposite side (cross coupling), or both motors. Although it was a purely chemotactic approach, a Braitenberg-style robot is able to track a plume towards a gas source by following the concentration gradient [12]. As the first gradient-based algorithms do not consider wind information, the robot does not know whether it is following a plume towards or away from its source. Turning the robot in proportion to the concentration gradient in dependence of the upwind direction solves this problem [13]. As the rotors of the micro-drone introduce strong disturbances, measuring a local concentration gradient with spatially separated sensors is not feasible. Instead a new measuring strategy was developed, which basically splits up one measuring position into two spatially separated ones. In order to respect the minimum step size of the micro-drone of 1 m and to progress faster to the source, the step size in upwind direction was set to 1.5 × step size (Figure 7).

B. Gas Distribution Modeling/Mapping (GDM)

Gas distribution mapping can be used in a number of relevant application areas where a better understanding of the gas dispersion is needed, such as environmental monitoring and safety and security related fields.

To build a predictive gas distribution model, the Kernel DM+V/W algorithm introduced by Reggente and Lilienthal [14] was used. The input to this algorithm is a set $D = \{(x_i,r_i,v_i)\}1\leq i\leq n$ of gas sensor measurements r_i and wind measurements v_i collected at locations x_i. The output is a grid model that computes a confidence estimate, as well as the distribution mean and variance for each cell k of the gridmap (Figure 8).

Additional sensors for temperature and humidity are integrated into the gas-sensing payload but so far not taken into account. It is conceivable to use these data for sensor compensation algorithms or to correlate the environmental conditions, e.g., in the case of fire. Integration of optical or IR data is another viable aspect.

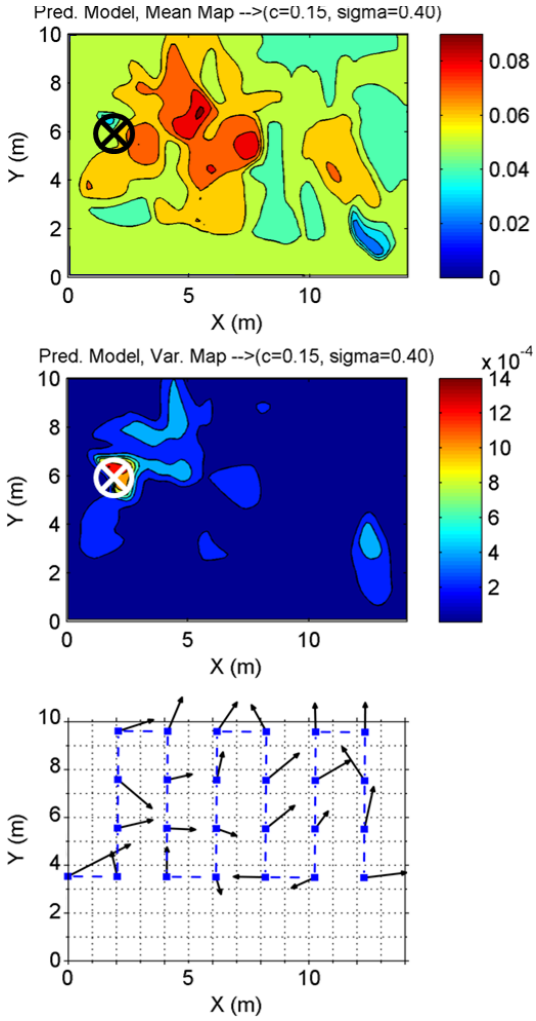


Figure 8. GDM Experiment: Predictive mean (top) and variance map (middle) of the gas distribution and the corresponding mean airflow map (bottom) and the path of the micro-drone created using Kernel DM+V/W. The gas source was located approx. at position (2, 6) m and is denoted by the cross. The concentration value of CO2 is given in % by volume.

IV. SENSOR-ENABLED RFID TAGS FOR SAFEGUARD OF DANGEROUS GOODS

The project "Sensor-enabled RFID tags for safeguard of dangerous goods" with acronym SIGRID investigates and assesses possibilities to improve safety and security of dangerous goods transports through the use of the latest RFID technology [15]. This technology can be used to

greatly enhance the transparency of the supply chain and aid logistics companies in complying with regulations. In the context of SIGRID, custom RFID sensor tags (Figure 9) were developed to monitor dangerous goods during transport and help to prevent hazards by allowing timely countermeasures. This requires the combination of communication technology and sensor functionality with low power consumption and small design.

To achieve long battery-life, the use of very energy efficient sensors is mandatory. Other desirable properties of the sensors include high accuracy, long lifetime, and short response time. For gas sensors a high selectivity is also very important. Currently, four types of sensors are integrated in the RFID tag, which are a combined humidity and temperature sensor, gas sensors for carbon monoxide (CO) and oxygen (O_2), and a tilt sensor. Other interesting sensor options that might be tested in future include sensors for detecting the filling level and sensors for monitoring the operation of equipment that is built into the container like a stirring unit.

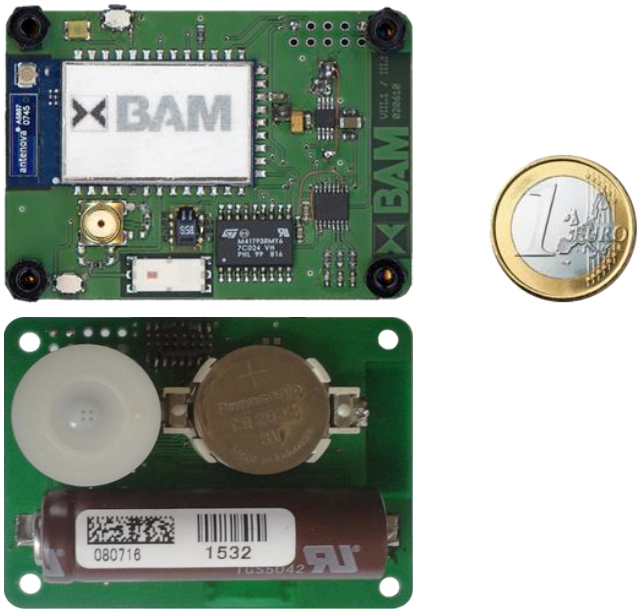


Figure 9. Prototype of the sensor enabled RFID tag

The integrated sensors enable the system for recognizing and evaluating of different scenarios. Adequate gas sensors indicate an emission from the containments via measured concentrations. If a possible gas release from the transported substance cannot be detected because of lacking the proper sensor, the O_2-sensor can indicate a leakage through decreasing oxygen values. For numerous dangerous goods a maximal transport temperature is defined to prevent any chemical reaction. Temperatures can be measured and compared periodically to substance specific values. If that value or a tolerance is exceeded an alarm or countermeasure can be activated. The tilt sensor can be triggered on heavy vibrations or tilting of the containment. In case of a dangerous goods accident the available information about the type, amount, and condition of the dangerous goods can be used to accurately inform the relief forces. Unavailable or inaccurate information represents a significant problem. This often leads to a delay of the rescue operation, because relief forces must be aware of the involved substances and their condition to effectively protect themselves against them.

Within the scope of the project, an RFID tag was developed, that allows connecting with different types of sensors. This RFID tag combines the advantages of semi active (only sensors are battery supplied) and active tags (sensors and radio communication are battery supplied). On one side, this tag is compatible to the ISO 18000, respectively EPC-Gen2 standards; on the other side, this tag has also the ability to communicate via the widely adopted wireless LAN standard Wi-Fi. Because the tag is woken up the same way as battery-less passive tags and for that reason does not need to power-up a receiver-module, battery-lifetimes of more than half a year are possible - just as with semi active tags. After the tag is woken up, the WLAN module is activated and allows very fast data transmission, that otherwise would only be achievable with active tags. This greater transmission speed makes the tag suitable as storage device for much larger amounts of data, than the ones that are normally possible with RFID tags. The possibility to store great amounts of data in combination with a very long battery lifetime makes this tag ideal for use as a data logger. Logging intervals can be configured individually for every sensor. The tag has also an open interface, which allows an easy integration of different kinds of sensors.

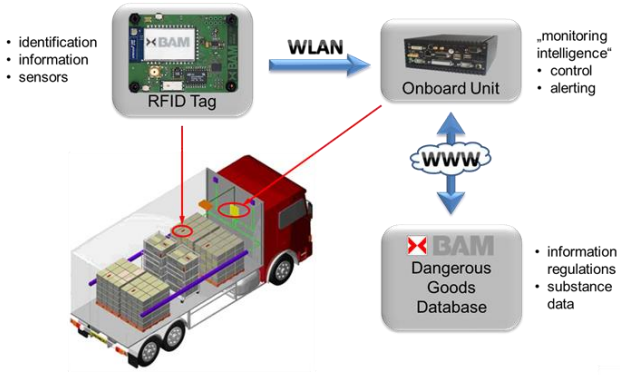


Figure 10. Interaction between the main system components during transport

Sensor-Tags, data communication, and software are combined to an interactive solution, which can tackle various scenarios during dangerous goods transports. The underlying information is provided by a data base with expert knowledge, in this case the BAM dangerous goods database "GEFAHRGUT" [16]. Figure 10 displays the interaction between the main system components during transport. The focal point of the vehicle equipment is the onboard unit (OBU), which consists of a ruggedized industry PC that is specially designed for use in a truck. The main functions of the OBU include acquisition of position data via GPS, routing, generation of transport documents,

data communication via the mobile phone network, monitoring of the load with sensors and surveillance cameras as well as WLAN connectivity. It is either possible to read the sensors of the semi-active transponders or sensors that are permanently installed in the loading area. The OBU constantly monitors the measurements to ensure, that they are in the allowable range. If that is not the case, an alarm is automatically triggered. Current status messages are transmitted to the centralized database, that has also the cargo manifest stored. In case of need, the OBU should supply the relief forces with all required information via WLAN. But if the OBU gets destroyed during an accident, all information is still available through the centralized database. Possible extensions of the system take into account vehicle data or GPS information in terms of route planning and geo-fencing.

V. Multifunctional Sensor for Spatially Resolved Under-surface Monitoring of Gas Storage Areas

One of the main unsolved issues of under-ground storages for, e.g., carbon dioxide, hydrogen, and natural gas (primarily methane) is the comprehensive surveillance of these areas with reasonable effort and costs. Conventional sensors, such as soil air probes or borehole probes, can only be used for punctual or locally limited measurements. Further they require invasive application, which causes structural influences.

Figure 11. Membrane based gas sensor.

A. Sensing principles

BAM in cooperation with the company MeGaSen UG carries out a research project to enhance and validate an innovative approach for distributed subsurface monitoring of gas storage areas. The concept combines different measurement technologies to one multifunctional sensor: membrane-based gas measurement technology for in-situ monitoring of gases in soil [17] and fiber optical sensing of temperature and strain as a measure for structural change [18].

The gas sensor (Figure 11) is based on the principle of selective permeation of gases through a membrane. The measuring method combines the gas specific diffusion rates through a membrane with Dalton's law of partial pressures. It enables the calculation of gas concentrations with the ideal gas law using measurements of pressure, time, and temperature. The sensor is implemented in form of a flexible tube. The synthetic material allows a variable subsurface installation, e.g., in meander or network form (Figure 12). So far the gas concentration measurement is implemented for carbon dioxide and oxygen, further gases should follow, e.g., methane and hydrogen sulfide.

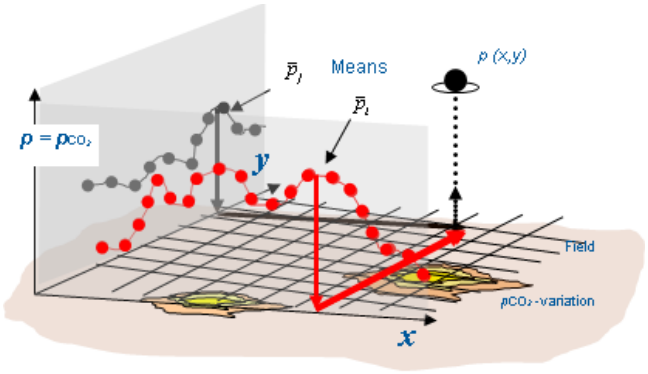


Figure 12. Spatially distributed gas monitoring built up of several membrane sensors. The brown and yellow areas indicate CO_2 hotspots underground. The red and grey curves display the averaged measurements of the partial CO_2 pressure over x and y.

Glass fiber optical sensors use the effects of stimulated Raman scattering (SRS) and stimulated Brillouin scattering (SBS) for spatially resolved measuring of temperature and strain. Distributed strain measurements can also be performed with polymer optical fibers using optical time-domain reflectometry (OTDR). BAM develops, validates and uses such sensor systems in different areas of application, such as geotechnics, structural engineering, and physical protection.

Combining these two sensor types (membrane sensor and fiber sensor) to a multifunctional sensor offers an innovative and promising approach for spatially resolved monitoring of large-scale areas [19]. Both technologies offer advantageous specifications, which support and encourage their combination:

- Distributed, area-wide applicable measuring system with spatially resolution of all variables
- Scalable and adaptable form of application, depending on monitoring object and problem
- Non-invasive system (no influence on the monitoring object, due to permanent presence of the sensor in the ground)
- No sensitivity against electro-magnetic fields (e.g., lightning and high-voltage lines)
- Applicable in explosive surroundings (no electrical components at the measuring locations)
- High thermal and chemical robustness
- Comparatively reasonable components

The structural combination is accomplished by linkage of the sensitive elements membrane sensor and optical fiber. For this purpose, geogrid materials (Figure 13) act as a carrier material.

Combined data analysis should be investigated and further developed to attain synergy effects, increase the sensitivity and informational value, and address new fields of application. Using sensor data fusion allows in-depth analysis of soil processes and early detection of relevant changes. For instance, the combined analysis of gas concentration, temperature, and strain can enable an indication of very small crack formation and gas emission, with significant higher reliability compared to sole gas measurements.

Figure 13. Geogrid with integrated fiber optical sensors.

Two immediate fields of application are addressed: Landfills produce greenhouse gas and warmth. The combination of both measurement methods should allow a potent landfill monitoring by containment of chemical active areas and leakages.

Underground storage of CO_2 as part of CCS as well as extraction and production of gases from geological areas can lead to mechanical changes of the deck rock (lowering / elevation), with which a regional tension field is build up. Thus, gas-leading gaps can be induced, which cause local ground structure changes. The simultaneous measurement of spatially resolved gas concentrations and strain allows the development of an efficient early warning system.

B. Experimental Validation

The validation, optimization, and practical demonstration of the overall system are carried out on the BAM Test Site Technical Safety (BAM TTS) [20][21]. For this purpose, a test field in application relevant scale of 20 x 20 m^2 was built up (Figure 14). Additionally, a corresponding laboratory setup was constructed. Both setups use the same sensors and measuring procedures as well as the same soil, which acts as ambient medium. The laboratory setup (Figure 15) was designed as a cooperative tool to prepare the test site build up and operation.

Comparable investigations can be performed in small-size and short-term to estimate the efforts and benefits of full-size experiments. Gas emission processes can be simulated as well as temperature and mechanical impact to validate and enhance the proposed multifunctional sensor. First, CO_2 leakage experiments demonstrate the applicability of the technology for rapid leak detection, and thus qualify the sensor particularly for safety application in Carbon Capture and Storage (CCS) areas [22].

Figure 14. Built-up of the test site. Top: level with 4 linear sensors. Bottom: level with 40 linear sensors. Each sensor line combines membrane gas sensing and fibre optical sensing of temperature and strain.

Figure 15. Laboratory setup with corresponding design to the test site and size of 2.5 x 1.5 x 0.1 m^3.

VI. CONCLUSION

Safety related monitoring often is necessary in complex scenarios. It requires distinct information to evaluate the situation and to determine the further operation. The combination of several measurands can improve the informative value of a monitoring system in terms of measuring diversity and accuracy.

To present the great potential of such systems, four examples for monitoring in safety relevant scenarios are presented in this paper, which combine multiple application specific sensor techniques. An important result considering each of the examples and multi-sensor systems in general is that data processing and display of the results with focus of the relevant information is crucial. The experiences gained from these projects show that the focus should lay on the

final application and end-users should be involved already in the conception of multi-sensor systems. Data fusion offers broad possibilities, but conditions and objectives should be well defined and expediently applied.

ACKNOWLEDGMENT

The authors thank all participating colleagues from BAM and their project partners. The authors also express their gratitude to the German Federal Ministry of Economics and Technology for funding the research (MNPQ Program; file numbers 28/07 and 17/11 and ZIM Program File KF2201041SM1).

REFERENCES

[1] M. Bartholmai, E. Koeppe, and P. P. Neumann, "Monitoring of Hazardous Scenarios using Multi-Sensor Devices," Proceedings of SENSORDEVICES 2013 - The 4th International conference on sensor device technologies and applications, pp. 9-13, 2013.

[2] E. Koeppe, M. Bartholmai, A. Liehrs, and J. H. Schiller, "Radio-based multi-sensor system for person tracking and indoor positioning," Proceedings of WPNC 2012 - 9th Workshop on positioning, navigation and communication, pp. 180-186, 2012, doi: 10.1109/WPNC.2012.6268761.

[3] E. Koeppe, D. Augustin, A. Liehrs, and J. H. Schiller, "Automatic 3D Calibration for a Multi-Sensor System," Proceedings of Indoor Positioning and Indoor Navigation (IPIN), pp. 1-6, 2012, doi: 10.1109/IPIN.2012.6418870.

[4] E. Koeppe, D. Augustin, A. Liehrs, and J. H. Schiller, "Self-calibration-method for an inertial navigation system with three 3D sensors," Proceedings of Inertial Sensors and Systems (ISISS), pp. 1-4, 2014, doi: 10.1109/ISISS.2014.6782522.

[5] M. Bartholmai and P. P. Neumann, "Adaptive Spatial-Resolved Gas Concentration Measurement Using a Micro-Drone," tm - Technisches Messen, vol. 78, no. 10, pp. 470-478, 2011, doi: 10.1524/teme.2011.0158.

[6] P. P. Neumann, S. Asadi, J. H. Schiller, A. J. Lilienthal, and M. Bartholmai, "Autonomous Gas-Sensitive Microdrone – Wind Vector Estimation and Gas Distribution Mapping," IEEE Robotics and Automation Magazine, vol. 19, no. 1, pp. 50-61, 2012, doi: 10.1109/MRA.2012.2184671.

[7] P. P. Neumann, V. Bennets, and M. Bartholmai, "Adaptive Gas Source Localization Strategies and Gas Distribution Mapping using a Gas-sensitive Micro-Drone," Proceedings of 16. GMA/ITG-Fachtagung Sensoren und Messsysteme 2012, pp. 800-809, 2012, doi: 10.5162/sensoren2012/P5.4.

[8] P. P. Neumann, M. Bartholmai, V. Bennets, and A. Lilienthal, "From Insects to Micro Vehicles - A Comparison of Reactive Plume Tracking Strategies," Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS), 2014.

[9] T. Lochmatter and A. Martinoli, "Tracking Odor Plumes in a Laminar Wind Field with Bio-Inspired Algorithms," Proccedings of 11th International Symposium on Experimental Robotics, vol. 54, pp. 473-482, 2009.

[10] H. Ishida, K. Suetsugu, T. Nakamoto, and T. Moriizumi, "Study of autonomous mobile sensing system for localization of odor source using gas sensors and anemometric sensors," Sensors and Actuators A, vol. 45, no. 2, pp. 153-157, 1994.

[11] V. Braitenberg, "Vehicles: Experiments in Synthetic Psychology," The MIT Press, February 1986.

[12] A. Lilienthal and T. Duckett, "Experimental Analysis of Gas-Sensitive Braitenberg Vehicles," Advanced Robotics, vol. 18, no. 8, pp. 817-834, 2004.

[13] R. Russell, A. Bab-Hadiashar, R. Shepherd, and G. Wallace, "A comparison of reactive chemotaxis algorithms," Robotics and Autonomous Systems, vol. 45, no. 2, pp. 83-97, 2003.

[14] M. Reggente and A. J. Lilienthal, "Using Local Wind Information for Gas Distribution Mapping in Outdoor Environments with a Mobile Robot," Proceedings of IEEE Sensors 2009, pp. 1715-1720, 2009, doi: 10.1109/ICSENS.2009.5398498.

[15] T. Goedecke, A. Pettelkau, S. Hohendorf, D. Damm, M. Bartholmai, and M. Farahbakhsh, "Securing of Dangerous Goods Transports by RFID-Tags with Sensor-Functionality and integrated Database "GEFAHRGUT" Information (SIGRID)," Proceedings of the 17th IAPRI World Conference on Packaging 2010, pp. 639-642, 2010.

[16] BAM Dangerous Goods Database. [online]. Available from: http://www.dgg.bam.de 2014.11.24

[17] D. Lazik, S. Ebert, M. Leuthold, J. Hagenau, and H. Geistlinger, "Membrane Based Measurement Technology for in situ Monitoring of Gases in Soil," Sensors, vol. 9, no. 2, pp. 756-767, 2009, doi: 10.3390/s90200756.

[18] S. Liehr. P. Lenke, M. Wendt, K. Krebber, M. Seeger, E. Thiele, H. Metschies, and B. Gebreselassie, "Polymer Optical Fiber Sensors for Distributed Strain Measurement and Application in Structural Health Monitoring," IEEE Sensors Journal, vol. 9, no. 11, pp. 1330-1338, 2009, doi: 10.1109/JSEN.2009.201835.

[19] M. Bartholmai, P. P. Neumann, and D. Lazik, "Multi-functional Sensor for Monitoring of CO_2 Underground Storage by Comprehensive and Spatially Resolved Measuring of Gas Concentrations, Temperature and Structural Changes," Energy Procedia, vol. 37, pp. 4033-4040, 2013, doi: 10.1016/j.egypro.2013.06.303.

[20] BAM Testside for Technical Safety. [online]. Available from: http://www.tts.bam.de 2014.11.24

[21] P. P. Neumann, H. Kohlhoff, K.-D. Werner, J. Erdmann, B. Eggeringhaus, M. Kammermeier, M. Schukar, F. Basedau, M. Bartholmai, D. Lazik, and S. Ebert, "Setup of a large scale soil test field with CO_2 injection for testing a novel distributed subsurface monitoring system for gas storage areas," 31th Danubia-Adria Symposium, pp. 238-239, 2014.

[22] M. Bartholmai, P. P. Neumann, K.-D. Werner, S. Ebert, and D. Lazik, "Linear Sensor for Areal Subsurface Gas Monitoring – Calibration Routine and Validation Experiments," IEEE Sensors 2014, pp. 942-945, 2014.

Optical, Mathematical, and Computational Foundations of Lensless Ultra-Miniature Diffractive Imagers and Sensors

David G. Stork and Patrick R. Gill
Computational Sensing and Imaging
Rambus Labs
1050 Enterprise Way, Suite 700
Sunnyvale, CA 94089 USA
{dstork,pgill}@rambus.com

***Abstract*—We describe the optical, mathematical and computational foundations for a new class of lensless, ultra-miniature computational imagers and image sensors. Such sensors employ phase gratings that have provably optimal optical properties and are integrated with CMOS photodetector matrices. These imagers have no lens and can thus be made extremely small ($\sim$100 μm) and very inexpensive (a few Euro cents). Because the apertures are small, they have an effective depth of field ranging from roughly 1 mm to infinity. The grating acts as a two-dimensional visual "chirp" and preserves image power throughout the Fourier plane; thus the captured signals preserve image information. The final digital image is not captured as in a traditional camera but is instead computed from raw photodetector signals. The novel representation at the photodetectors demands powerful algorithms such as deconvolution, Bayesian estimation, or matrix inversion with Tikhonov regularization be used to compute the image, each having different bandwidth, space and computational complexities for a given image fidelity. Such imaging architectures can also be tailored to extract application-specific information or compute decisions (rather than compute an image) based on the optical signal. In most cases, both the phase grating and the signal processing can incorporate prior information about the visual field and the imaging or estimation task at hand. Our sensor design methodology relies on modular parallel and computationally efficient software tools for simulating optical diffraction, for CAD design and layout of gratings themselves, and for sensor signal processing. These sensors are so small they should find use in endoscopy, medical sensing, machine inspection, surveillance and the Internet of Things, and are so inexpensive that they should find use in distributed network applications and in a number of single-use or disposable applications, for instance in military, hazardous natural and industrial conditions.**

Keywords: ***Computational sensing, phase grating, diffractive imager, application-specific sensing, face detection, QR code reading***

I. INTRODUCTION

Recent theoretical and computational advances provide a foundation for a new class of computational optical image sensor: one that forgoes the use of traditional optical elements such as lenses and curved mirrors and relies instead upon diffractive optical elements [1], [2]. Whereas diffractive methods have been employed in other wavebands, such as millimeter-wave imaging, prior, traditional optical imaging architectures have generally been based on the *camera obscura* model—in which each point in the scene is imaged onto a single point on a sensor or image plane. This model has dominated the science and technology of imaging systems for several millennia, at least for sources illuminated by incoherent light. The Chinese philosopher Mo Ti traced an inverted image produced by a pinhole camera to record an image in the fifth century B.C.E. [3] and Johannes Kepler traced a real image projected by a converging lens onto paper in 1603. Chemical recording of projected images, such as by mercury or silver halide, was invented in 1826 and the first true digital camera was built in 1975 [4], all these exploiting the fundamental camera obscura architecture.

As photodetector sensor technology has improved and pixel pitches have become smaller, pixels can be made smaller than the optical diffraction limit of systems such as commercial cameras [5]–[7]. Pixels smaller than the diffraction limit, however, do not provide new image information. Instead, such sub-diffraction-limit pixels provide opportunities to make "smart pixels" with functionality beyond mere direct conversion of photons to electric current [8].

The rise in digital imaging, where image processing can be incorporated into the data chain, has enabled new imaging architectures. Although related concepts were explored in computational radar and x-ray astronomy, it was Cathey and Dowski who took an early and conceptually important step away from the traditional camera obscura model for optical imaging by exploiting digital processing in a deep way [9]. They designed a cubic-phase optical plate which, when inserted into the optical path of a traditional camera, led to an image whose (significant) blur was independent of the object depth: the image on the sensor plane did not "look good" as it would in a traditional camera obscura. Subsequent image processing sharpened the *entire* blurred image, thus leading to enhanced depth of field. Since then the field of *computational imaging* has explored imaging architectures in which the raw signals do not superficially resemble a traditional image; instead, the final image is computed from such signals. More and more of the total imaging "burden" is borne by computation, thereby expanding the class of usable optical components. In this way, many optical aberrations can be corrected computationally rather than optically. This imaging paradigm has led to new conceptual foundations of joint design of optics and image processing [10], as well

as a wide range of non-standard imaging architectures such as plenoptic, coded-aperture and multi-aperture systems, each with associated methods of signal processing [11]–[15].

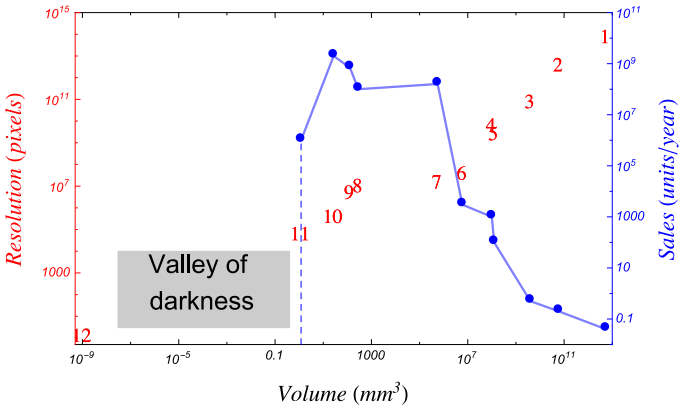


Fig. 1. The left ordinate axis in red shows the resolution (in pixels) versus the physical volume (in mm^3) of representative lens- and mirror-based telescopes and cameras (log-log scale). Notice there is a seven-order-of-magnitude range in physical volume devoid of such cameras (the **Valley of darkness**). 1 Grand Canaria telescope, 2 Hubble telescope, 3 1-m telescope, 4 30-cm telescope, 5 AWARE 2 camera, 6 Professional camera,7 Consumer DSLR, 8 iPhone 5 camera, 9 Pelican camera, 10 Miniature VGA, 11 Medigus camera, 12 Single photodiode (without lens). The right ordinate axis in blue indicates the sales of representative imagers of different physical volumes in units/year worldwide in 2011. (The unit sales figures are estimates based on historical data and market reports and do not include research prototypes and unreleased products.) There is a precipitous drop in sales at the Valley of darkness. Our lensless integrated diffraction grating/CMOS imagers lie within this "valley."

The economic pressures for miniaturization of electronic devices, including cameras, arising in the mobile computing market have led to smaller imager form factors [16]. Figure 1 shows the resolution, in total pixels per exposure, versus physical volume of imaging systems in the traditional camera obscura architecture (or curved mirror equivalent). While such imagers span 22 orders of magnitude in physical volume and 15 orders of magnitude in pixel resolution, the smaller the imager the greater the number sold commercially... but only down to a scale of roughly 1 mm^3. There is a conspicuous gap of seven orders of magnitude in physical volume—the "Valley of darkness"—between the smallest digital camera and a single unlensed photoreceptor. It seems that the camera obscura model has reached its physical limits and cannot be scaled much smaller. A new imaging architecture—with new optical, mathematical and computational foundations—is required to span the Valley of darkness.

Recently, a new miniature imaging architecture has been explored, one based on integrating optics with CMOS photodetectors [2], [17]–[19]. In brief, this architecture forgoes lenses and relies instead on simple square-wave diffraction gratings created in CMOS itself. The earliest designs in this architecture relied on CMOS wires to act as amplitude optical grating patches, the gratings producing a wavelet-like representation of the scene on the sensor matrix. More recently, square-wave *phase* gratings have also been explored [20]. For a given image resolution, such diffractive elements enable the construction of imagers much smaller than does the basic camera obscura model. (We mention in passing that related CMOS structures have been explored for integrated spectroscopy as well [21].) Note too that as given by the trends in resolution versus physical volume evident in Fig. 1, imagers in the Valley of darkness will have nominal resolutions (pixels per single frame) lower than roughly 10^5 pixels [22], [23]. Nevertheless, such low-resolution imagers—or high-resolution sensors—should find use in many applications, especially in the Internet of Things (see Section V).

There are a number of limitations of such previous work. First, amplitude gratings based on CMOS wires have poor low-light sensitivity because most of the incident light never strikes the photodetector. Second, regular diffraction gratings are by their very nature wavelength sensitive, i.e., the pattern of light on the photodetectors depends strongly upon the wavelength of incident light. Third, such imagers are sensitive to manufacturing defects—specifically a small deviation in the thickness of the grating layer can lead to a large (and difficult to correct) alteration of the diffraction pattern on the photodetectors [18].

The method we describe here, while based on integrated silicate phase optics and CMOS image sensors, is fundamentally different from prior work in a number of deep ways. Our method relies on novel special phase anti-symmetric spiral phase gratings, which overcome prior limitations and afford new functionality [24]. Moreover, our new sensor architecture enables the construction of new classes of ultra-miniature sensors whose output is an estimation of some property of the scene (e.g., visual motion) or a decision (e.g., face detection or barcode reading).

We begin in Section II with a discussion of our fundamental technology and turn in Section III to a short description of our software design and analysis tools. We describe our first hardware devices in Section IV. The full results of our hardware verification of the theory and design will be presented at a later date [25]. We mention a few application areas for such sensors and imagers in Section V and conclude in Section VI with some final remarks.

II. Sensor Optics and Technology

The following description of our sensor technology follows the data path—from target source through diffractive optics to photodetector to digital signal processing to final digital image or image estimation.

A. Optics of one-dimensional phase anti-symmetric gratings

The fundamental optical elements employed by our sensors are based on a new type of phase grating having phase antisymmetry. Figure 2 shows a cross section through a UV-curable acrylate binary phase grating, here specified by three free parameters, w_0, w_1 and w_2 [26]. (Generalizations to more free parameters and multiple thicknesses are straightforward.) Consider point **P** lying on the grating's plane of odd symmetry, shown as a vertical dashed red line. The steps in thickness of the acrylate grating correspond to a phase delay of π radians of the typical wavelength used in imaging. Such a phase difference means that light from each position on one side of the plane is cancelled via destructive interference by light from the symmetric position on the other side of the plane

because those waves arrive out of phase. Note especially that such cancellation occurs regardless of the vertical depth of **P**; as such, all points along the red dashed line are dark. We call this plane of destructive interference an "optical curtain" or simply "curtain" [27]. The location of the curtain on the sensor matrix below does not change despite manufacturing errors in overall grating thickness. Finally, as the angle of incidence of the light changes, the curtains tip by the same angle (Fig. 3), a transformation that makes calibration particularly simple problem of estimating a spatial shift. In this way, the sensor responses are invariant to variations in manufactured thickness and wavelength of incident light (Fig. 4). Greater wavelength invariance can be achieved by using an additional layer of silicate with different index of refraction and dispersion coefficient than the primary grating, much as chromatic aberration is corrected in classical lens-based imaging systems through the use of multiple lenses with different indexes of refraction and dispersion [6].

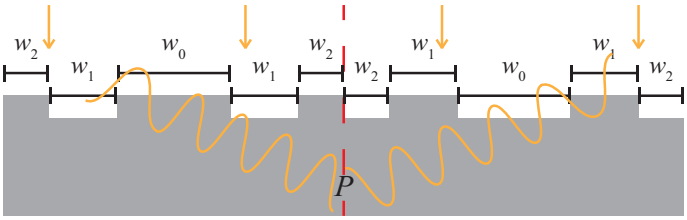


Fig. 2. A cross section through a binary anti-symmetric phase grating, where the plane of odd symmetry is marked with a vertical dashed red line. The parameters w_0, w_1 and w_2 describe the surface profile. For the medium's index of refraction n, the step height is chosen to corresponds to optical phase delay of π radians along the red dashed line or "curtain." For such a phase anti-symmetic grating, curtains exist even if the incident light is not normal.

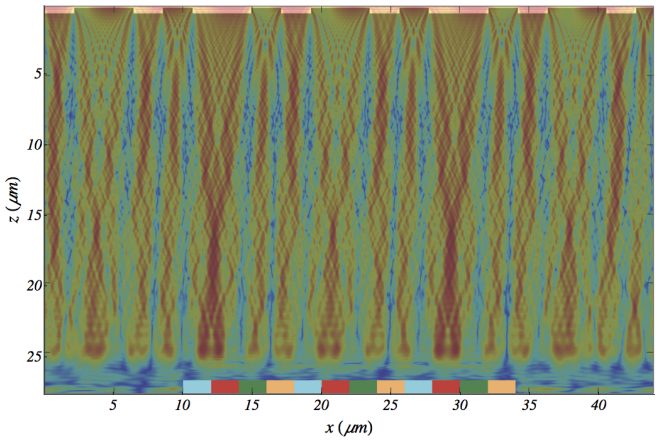


Fig. 3. A finite-difference wave simulation of the electric field energy density for monochromatic light incident at 3.5° passing through a phase anti-symmetric grating where x denotes the position left-to-right and z the depth within the silicate medium. The curtains lie beneath the points of odd symmetry and are tipped at the same angle as the incident light. Such curtains are invariant to the wavelength of incident light. The photodetector matrix, shown as pixels in different colors, lies along the bottom.

B. Phase anti-symmetric spiral gratings

The scenes we seek to image are two-dimensional and therefore the one-dimensional phase anti-symmetric grating and photosensor array just described must be generalized to two dimensions. Specifically, two-dimensional gratings must include segments at every orientation so as to sample the Fourier domain uniformly (and possess no zeros) and thereby enable computational reconstruction of the image from sensor responses. Figure 5 shows two examples of basic spiral grating tiles—having four-fold and six-fold chiral symmetry. These spiral grating tiles are constructed by sweeping one-dimensional phase anti-symmetric gratings perpendicularly along the length of each spiral arm. The phase anti-symmetric gratings are lengthened and made more complicated (use more ws) to cover the full tile area and feasible Fourier domain. Both spiral gratings pass information at all orientations and spatial frequencies up to the Nyquist limit, and can be tiled to cover a full photodetector matrix of arbitrary area (Fig. 6) [24]. In actual sensors, incident light covers an area at least as large as that of a full individual tile element.

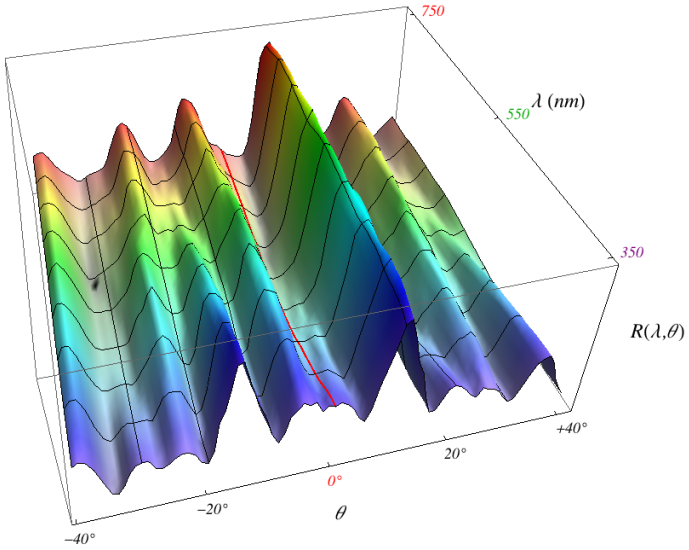


Fig. 4. The response of a single photodetector (pixel) beneath a phase anti-symmetric grating (such as **P** in Fig. 2) as a function of angle of incident light, θ, and wavelength of light, λ. Notice that for normally incident light ($\theta = 0°$) the response nearly vanishes at all wavelengths and that at each incident orientation, the response is nearly invariant with respect to wavelength. The specific form of this response function depends upon the profiles of the grating (described by w_is), which can be tailored to extract information most appropriate to particular applications, including non-imaging applications.

The wave optics described above assumes the incident illumination is plane-wave. In such a case the pattern of light produced by a grating does not depend upon the distance of the object, so long as the object is farther from the sensor than roughly 10 times the spatial scale of the sensor itself. As such, our sensor has extremely large effective depth of field, from roughly 1 mm to infinity.

The pattern of light produced by the diffraction grating strikes the CMOS photodetector matrix beneath and the signals are sent off chip for digital processing.

C. Signal processing

Sensed signals in our sensor do not resemble an image in a camera obscura but must be processed to yield a digital image. We assume the overall forward imaging model is described by:

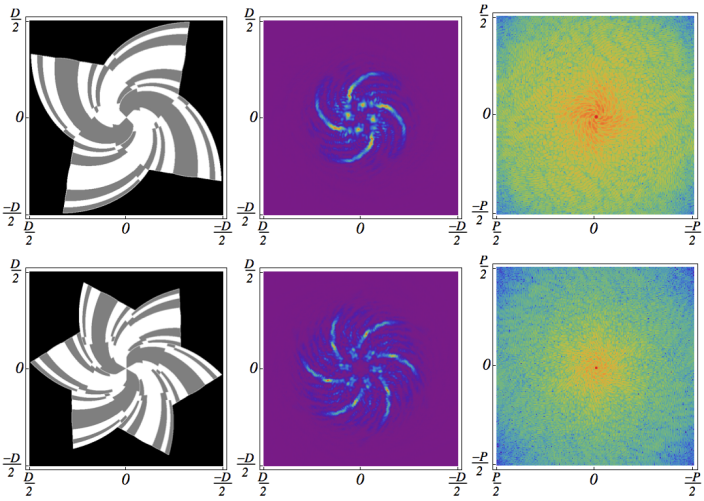


Fig. 5. The left column shows phase anti-symmetric spiral binary gratings, the middle column the point-spread function each produces (both figures of spatial extent $D \times D$, for some distance D). The right column shows the corresponding modulation transfer function (modulus of the Fourier transform) of extent $1/P \times 1/P$, where P is the pixel pitch and determines the Nyquist rate. The top row corresponds to four-fold chiral symmetry and the bottom row corresponds to six-fold chiral symmetry.

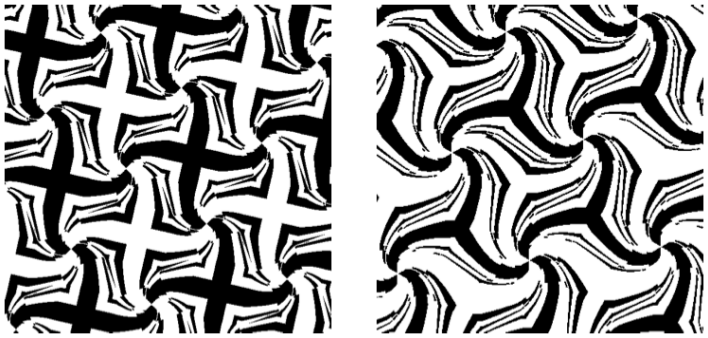


Fig. 6. The individual grating tiles of Fig. 5 can be packed to cover a photodetector matrix of arbitrary area. Alternate approaches to tessellating a sensor array with such individual grating designs are not as space efficient.

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{y}$ is the vector of photodetector pixel responses, $\mathbf{x}$ is a vector of inputs from the scene, $\mathbf{A}$ the system matrix describing the linear transformation performed by the two-dimensional optical grating, and $\mathbf{n}$ is additive noise, which describes photodetector noise, Poisson photon statistics, quantization noise, etc. (Other models, such as simple multiplicative noise, could also be assumed.) We let $\mathbf{x}$ be m-dimensional, both $\mathbf{y}$ and $\mathbf{n}$ be n-dimensional; hence $\mathbf{A}$ has dimensions $m \times n$.

The regularized least-square estimation problem—that is, the reconstruction of the image—can be expressed as finding the image $\hat{\mathbf{x}}$ that minimizes the error or cost function

$$\mathcal{C} = \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|^2 + \|\mathbf{\Gamma}\hat{\mathbf{x}}\|^2, \tag{2}$$

where $\mathbf{\Gamma}$ weights the different components of $\hat{\mathbf{x}}$, for instance to accommodate differences prior probabilities of pixel values in the expected scenes. The image that minimizes the cost $\mathcal{C}$ in (2) is [28]

$$\hat{\mathbf{x}} = \left(\mathbf{A}^t\mathbf{A} + \mathbf{\Gamma}^t\mathbf{\Gamma}\right)^{-1}\mathbf{A}^t\mathbf{y}. \tag{3}$$

In the special case that prior information about scene statistics implies that each component of $\hat{\mathbf{x}}$ should be penalized equally ($\mathbf{\Gamma} \propto \mathbf{I}$, the identity matrix), the solution can be written as

$$\hat{\mathbf{x}} = \left(\mathbf{A}^t\mathbf{A} + \gamma\mathbf{I}\right)^{-1}\mathbf{A}^t\mathbf{y}, \tag{4}$$

where γ is a scalar Tikhonov regularization parameter, whose optimal value depends upon the noise level [24], [29]. Cost functions other than that in (2) can be used as well, for instance those based on the *total variation* or *TV* norm of $\hat{\mathbf{x}}$, or on the L_1 norm, or on Bayesian prior information, or on weighted combinations of such penalty terms [30].

The computational burden of estimating the "best" image (in a sum-squared-error sense) compatible with the measured sensor signals $\mathbf{y}$ depends upon the particular form of the cost function $\mathcal{C}$. For the simple Tikhonov regularization in (4), before operation one precomputes the Moore-Penrose pseudo-inverse (possibly for different values of the regularization parameter)—an $\mathcal{O}(n^3)$ operation. Image estimation after signal capture is then a simple matrix multiply, an $\mathcal{O}(n^2)$ operation, easily parallelized to run at video rates in real-time on an FPGA or Graphics Processing Unit, if necessary. We note in passing that under certain circumstances (e.g., the function of the grating can be well approximated by a convolution operation), efficient Fourier estimation methods can be used instead, with an $\mathcal{O}(n \ln n)$ complexity.

Such estimation is well-conditioned and has higher fidelity when the modulation transfer function of the optical element contains no zeros, as is ensured by our spiral anti-symmetry phase gratings. The condition number of the real, non-negative matrix $\mathbf{A}$ is the ratio of the magnitudes of the largest and smallest eigenvalues, i.e.,

$$\kappa(\mathbf{A}) = \frac{|\lambda_{max}|}{|\lambda_{min}|}, \tag{5}$$

which of course is always greater than or equal to 1.0. The smaller the value of $\kappa(\mathbf{A})$, the less noise-prone $\hat{\mathbf{x}}$ will be. For instance, if the matrix is proportional to the identity matrix, that is $\mathbf{A} \propto \mathbf{I}$, then its inverse can be computed with negligible loss in information or in significant bits in its components. Simulation studies of the physics of our phase anti-symmetric spiral gratings show that the condition numbers are roughly 500.

Other reconstruction methods include inverse Wiener filtering and Bayesian methods such as Richardson-Lucy deconvolution [31], each with computational complexities and fidelities that depend upon the accuracy of prior information about the source and other parameters. Figure 7 shows the estimation of an image through simple matrix inversion with Tikhonov regularization summarized in (4).

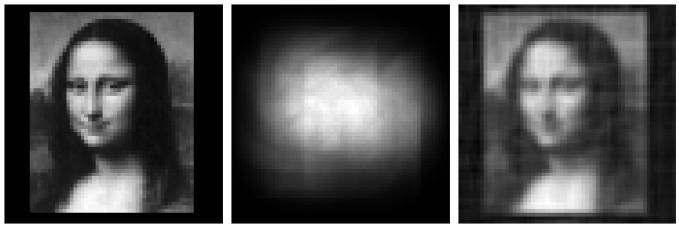


Fig. 7. Image sensing and computational reconstruction of Leonardo's *Mona Lisa* from a lensless phase anti-symmetric spiral phase grating sensor. (Left) The input image. (Middle) The simulated response on the photodetectors due to the six-fold grating in Fig. 5, and (right) the reconstruction by Eq. 4. This image estimate is of higher fidelity than the estimate based on traditional square-wave amplitude gratings and photodetector arrays of comparable number of pixels and overall noise level described in earlier work.

III. Simulation/design tools and methodology

Our sensor system design and analysis methods are based on a modular architecture comprising three software tools, all written in *Matlab* and executed on a large network of PCs:

- **Optics of phase gratings**: We simulate the interaction of light with gratings, for instance by finite-difference wave algorithms. These full-three-dimensional simulations reveal the electromagnetic energy density throughout the silicate grating volume (see Fig. 3) and predict the response of physical photodetector pixels to light of different wavelengths and incident angles, such as in Fig. 4.
- **CAD design of gratings and tiles**: We design gratings (spiral and otherwise) and their tilings starting from a mathematical description of the grating, often parameterized by the number of arms, arm chirality and curvature function, and phase cross-section as a function of distance from the center (i.e., the w_i shown in Fig. 2). The representation of our design is either a *Matlab*-compatible file for wave optics simulations or a *gdsII* file for silicon grating manufacture.
- **Sensor signal processing**: We continue to write our own image reconstruction, signal estimation and pattern recognition software in *Matlab*, often using standard libraries of matrix operations such as Moore-Penrose pseudoinverse. In some research systems, we incorporate free software such as QR code symbol reading software.

We can employ *Perl* software wrappers for these components in order to efficiently design and model the system's end-to-end performance. Such joint design methodology can often lead to superior system performance (higher fidelity reconstruction, few optical elements, etc.) than sequential design, where optics is designed first and only then is the signal processing designed [32].

IV. Hardware implementation

Our experimental hardware implementation of lensless imagers and sensors is based on a single pixel-addressable 10 Mega-pixel sensor from Aptina, Inc., with a single large grating platform comprising 40 experiments (Fig. 8). The gratings are made of a 50-μm-thick layer of acrylic (known as Ugoo)[1] with grating steps of 1.5 μm affixed to a 400-μm-thick glass substrate. Figure 9 shows a micrograph of one portion of the full grating. Input images are presented on an LCD display under computer control, and signals are read directly from the Aptina sensor and processed on a PC.

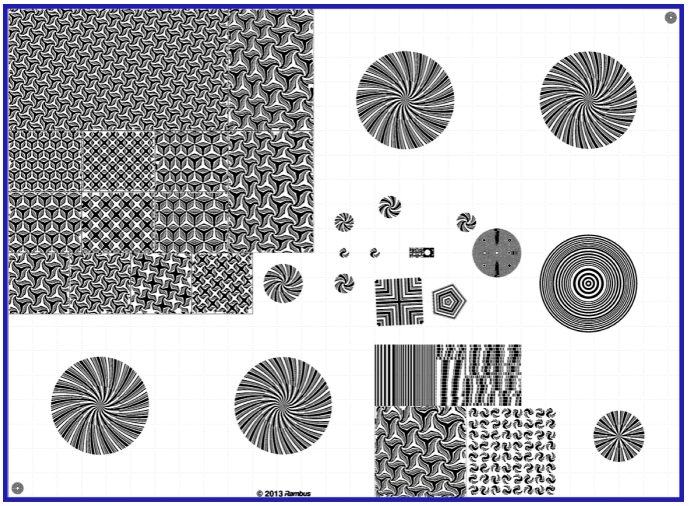


Fig. 8. The Ugoo silicate 5.5×4 mm grating platform contains 40 grating experiments. Some of the experiments involve tesselated areas for applications with lenses, as shown in Fig. 6. Fiducial marks at the lower-left and upper-right of the platform facilitate the estimation of the alignment of the grating with the underlying photodetector matrix.

Physical instantiation of the sensor, calibration of its $\mathbf{A}$ matrix, estimation of noise (photon and circuit), and development of accurate and computationally efficient image reconstruction methods for the hardware as built—all to verify the above theory—is in progress and will be presented separately [25].

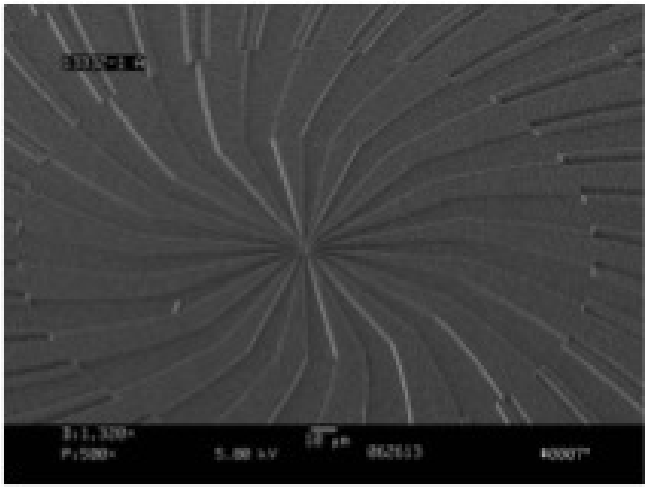


Fig. 9. A scanning electron micrograph of the grating at the lower left in Fig. 8.

V. Applications

There are many promising applications for our computational image sensors, which fall into a number of general

[1] manufactured by Holographix, LLC

categories. It is important to note, though, that these imagers were not designed to compete with high-resolution cameras that are larger and more expensive. Just as most animal and insect vision systems are fairly low resolution but numerous, so too our sensors are designed for numerous applications requiring only relatively simple vision and image analysis. It is as convenient to consider our devices as high-resolution sensors as it is low-resolution imagers.

Some general categories of applications follow.

A. Low-resolution imaging

The ultra-miniature size of our imagers and sensors make them especially appropriate for very small environments in medical and industrial endoscopy as well as traditional and some novel mobile computing devices. There are many surveillance applications that would profit from low- to mid-level resolutions as well. Because these sensors are so inexpensive (in bulk)—each less expensive than a single frame of 35-mm photographic film—they could find application in a number of one-use imaging scenarios arising in military theaters, hazardous industrial conditions (crash tests) and natural environments [32]. Another general area is inexpensive mobile medical imaging and sensing of the form pioneered by Ozcan and his colleagues [33]. A key design decision is where the signal processing should be implemented—close to the sensor itself, or instead on a host machine, possibly delayed from the signal capture.

The sensor described above is panchromatic, that is, it responds to any optical wavelength and yields a monochrome (grayscale) image. There are a number of ways to extend the lensless imaging architecture to yield color images. The most direct method would be to have three separate sensors, each optimized for a different optical wavelength—short, medium and long wavelengths, corresponding to blue, green and red—and integrating the component images.

B. Motion detection and estimation

The optical gratings and signal processing algorithms can be tailored to broad image sensing applications. For instance, because each pixel in such a sensor responds to light from an extended region in the visual field, only a few such pixels need be monitored in order to detect a change in the image. Therefore, such a sensor has very low power dissipation in its waiting or *sentinel* model. Once an image change has occurred, the full complement of pixels can be read so that an image can be captured or motion estimated. This kind of functionality is valuable for occupancy detection for controlled lighting, motion (motion-activated devices), visual looming (pre-impact automotive airbag deployment), interactive toys, and numerous applications in support of the Internet of Things [34].

C. Pattern recognition

These sensors can extract informative visual information for pattern recognition applications, such as face detection (authentication), one-dimensional barcode and two-dimensional QR code reading (Fig. 10), gesture recognition and many others. Of course, the signal processing is then based on principles of pattern recognition appropriate for the task at hand [35], [36]. For instance, QR code symbol reading software must determine the orientation or tip angle of a symbol, and does so by first locating the three fiducial concentric squares visible in Fig. 10 a), c) and d). This first step in QR symbol reading cannot be performed on the raw sensor representation. Moreover, because code analysis and error correction apply to the spatial domain, any lensless diffractive QR code symbol reader should first compute the pixel image of the symbol.

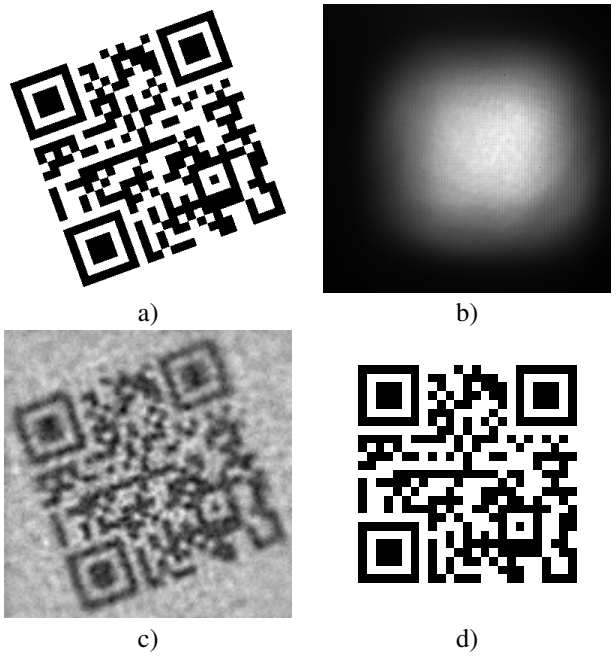


Fig. 10. a) A *Version 2* (25×25) target QR code symbol with information payload of 31 bytes. b) The raw signals in the 400 × 400 pixels array in our computational sensor. c) The digital image computed from the sensor signals using Tikhonov regularization. d) The final digital image, rotated and thresholded by line to yield roughly 50% white pixels. This final image is presented to *ZXing* QR code reading software, which decodes the image to extract its 31-byte code. Note that these barcode images relied on a grating designed for general imaging; a special purpose grating, designed to extract straight lines and right angles, with corresponding digital processing, would likely yield QR symbol images of higher fidelity and higher barcode recognition rates. Note the slight reconstruction errors in the upper-right pixels in d). Despite such slight reconstruction errors, error correction in the symbol reading algorithms ensured this symbol was decoded accurately.

Such a low-resolution sensor is unlikely to support high-accuracy face recognition among many candidate identities [37], but could be used to identify whether some face—*any* face—is present. Such functionality would be valuable for waking up appliances or other connected devices in the Internet of Things. Figure 11 shows the results of realistic simulations of such a face presence detector based on the sensor described above. The classifier is based on a nearest-neighbor algorithm [35, Chapter 4]. The test images consisted of 168 grayscale face images in various orientations and scales as well as simple non-face images. All recognition and classification was performed in the raw sensor representation—no traditional human-interpretable images were computed.

Let $\mathcal{F}$ denote the set of sensor patterns corresponding to faces (including transformations of rotations and scaling), and $\mathcal{G}$ the set of general (i.e., non-face) images. For each of the 168 3600-dimensional patterns $\mathbf{x} \in \mathcal{F}$, we computed the Euclidean distance $D(\mathbf{x}, \mathbf{x}')$ to the nearest other face pattern $\mathbf{x}'$ $(\neq \mathbf{x}) \in \mathcal{F}$. The histogram of such distances is shown at the front of

Fig. 11 in green. Then for each such pattern $\mathbf{x}$ we compute the Euclidean distance to the nearest non-face pattern $\mathbf{x}'' \in \mathcal{G}$. This histogram is shown in red. Of course, on-average such inter-face distances are less than the distances from faces to non-face patterns, i.e., $D(\mathbf{x}, \mathbf{x}') < D(\mathbf{x}, \mathbf{x}'')$. Because there is some overlap in the red and the green histograms, this face detection error is not 0 but in fact roughly 0.09. The Bayes classifier based on this distance D error is shown along the far-right face in Fig. 11.

The above analysis was repeated on the 1800, 900, 400, 200, 100 and 50 features, yielding the additional green and red histograms in Fig. 11. As expected, all histograms shift to smaller overall distance D in the subspaces and the overlap increases; thus the face/non-face error increases as the feature space has fewer and fewer dimensions. These simulation results show, however, that our computational diffractive imager design should yield an acceptable single-frame detection error rate of roughly 0.1 with as few as 100 features.

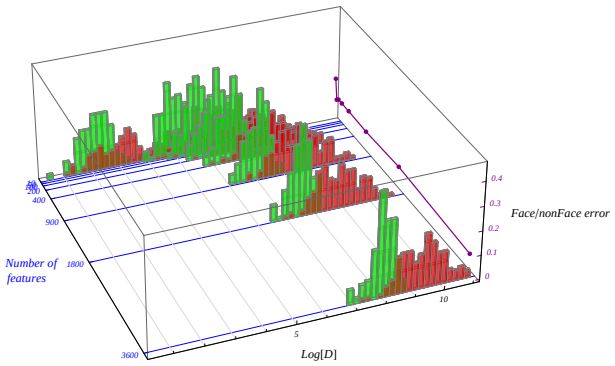


Fig. 11. The performance of a lensless ultra-miniature diffractive sensor for distinguishing faces from non-faces. The logarithm of the distance D in the full 3600-dimensional space and in subspaces of lower dimension (as listed at the left in blue) are shown. Along a blue line marking a given number of features, each green histogram represents the number of face patterns that have the indicated distance to other face patterns and each red histogram represents the (larger, on average) distance from a face to a non-face. The optimal classification rule is based on the crossing point of the red and the green histograms, and the overlap of the histograms represents the relative face/non-face classification error.

VI. Final remarks

We have designed and verified through full end-to-end system simulation a new class of lensless computational imagers based on phase anti-symmetric spiral gratings. We have built the components and are moving towards full hardware characterization of gratings and verification of imaging functionality. These imagers promise to be smaller (lower physical volume) than any existing lens-based imagers of comparable resolution, very inexpensive, and customizable to both imaging and a wide range of sensing and image measurement tasks. A full description of the hardware manufacture, calibration, and imaging performance are presented elsewhere [25].

Practical fielded applications will lead to many interesting problems in efficient application-specific algorithms, either on special-purpose ASICs, on highly parallel graphics processor units (GPUs), or on general-purpose central processor units (CPUs). Networks of such sensors highlight several problems and opportunities in power usage and bandwidth optimization.

Acknowledgments

We thank Thomas Vogelsang and Michael Ching for helpful comments.

References

[1] D. G. Stork and P. R. Gill, "Lensless ultra-miniature computational sensors and imagers," in *SensorComm 2013*, Barcelona, Spain, 2013.

[2] P. R. Gill, C. Lee, D.-G. Lee, A. Wang, and A. Molnar, "A microscale camera using direct Fourier-domain scene capture," *Optics Letters*, vol. 36, no. 15, pp. 2949–2951, 2011.

[3] T. Gustavson, *Camera: A history of photography from Daguerreotype to digital*. New York, NY: Sterling Publishing Co., 2009.

[4] D. Wooters and T. Mulligan, *A history of photography—from 1839 to the present*. New York, NY: Taschen, 2005.

[5] D. Falk, D. Brill, and D. G. Stork, *Seeing the light: Optics in nature, photography, color, vision and holography*. New York, NY: Wiley, 1986.

[6] M. V. Klein, *Optics*. New York, NY: Wiley Publishing, 1970.

[7] D. J. Brady, *Optical imaging and spectroscopy*. New York, NY: Wiley and Optical Society of America, 2009.

[8] T. Vogelsang, D. G. Stork, and M. Guidash, "Hardware validated unified model of multibit temporally and spatially oversampled image sensors with conditional reset," *Journal of Electronic Imaging*, vol. 23, no. 1, p. 013021, 2014.

[9] W. T. Cathey and E. R. Dowski, Jr., "A new paradigm for imaging systems," *Applied Optics*, vol. 42, no. 29, pp. 6080–6092, 2002.

[10] D. G. Stork and M. D. Robinson, "Theoretical foundations of joint design of electro-optical imaging systems," *Applied Optics*, vol. 47, no. 10, pp. B64–75, 2008.

[11] E. H. Adelson and J. Y. A. Wang, "Single lens stereo with a plenoptic camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 99–106, 1992.

[12] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 70:1–70:9, 2007.

[13] D. L. Marks, D. S. Kittle, H. S. Son, S. H. Youn, S. D. Feller, J. Kim, D. J. Brady, D. R. Golish, E. M. Vera, M. E. Gehm, R. A. Stack, E. J. Tremblay, and J. E. Ford, "Gigapixel imaging with the AWARE multiscale camera," *Optics and Photonics News*, vol. 23, no. 12, p. 31, 2012.

[14] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[15] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[16] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for 'Smart Dust'," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiComm 99)*, 1999, pp. 271–278.

[17] P. R. Gill, C. Lee, S. Sivaramakrishnan, and A. Molnar, "Robustness of planar Fourier capture arrays to colour changes and lost pixels," *Journal of Instrumentation*, vol. 7, pp. C01–61, 2012.

[18] A. Wang and A. Molnar, "A light-field image sensor in 180 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 257–271, 2012.

[19] A. Wang, P. R. Gill, and A. Molnar, "Light field image sensors based on the Talbot effect," *Applied Optics*, vol. 48, no. 31, pp. 5897–5905, 2009.

[20] S. Sivaramakrishnan, A. Wang, P. R. Gill, and A. Molnar, "Enhanced angle sensitive pixels for light field imaging," in *IEEE International Electron Devices Meeting (IEDM)*, 2011, pp. 8.6.1–8.6.4.

[21] C. Peroz, S. Dhuey, A. Goltsov, M. Volger, B. Harteneck, I. Ivonin, A. Bugrov, S. Cabrini, S. Babin, and V. Yankov, "Digital spectrometer-on-chip fabricated by step and repeat nanoimprint lithography on pre-spin coated films," *Microelectronic Engineering*, vol. 88, no. 8, pp. 2092–2095, 2011.

[22] O. Cossairt, M. Gupta, and S. Nayar, "When does computational imaging improve performance," *IEEE Transactions on Image Processing*, vol. 22, no. 2, pp. 447–458, 2013.

[23] O. Cossairt, M. Gupta, K. Mitra, and A. Veeraraghavan, "Performance bounds for computational imaging," *Imaging and Applied Optics*, 2013.

[24] P. R. Gill and D. G. Stork, "Lensless ultra miniature imagers using odd-symmetry phase gratings," in *Proceedings of Computational Optical Sensing and Imaging (COSI)*, Alexandria, VA, 2013.

[25] ——, "Hardware verification of an ultra-miniature computational diffractive imager," in *Proceedings of Computational Optical Sensing and Imaging (COSI)*, Kohala Coast, HI, 2014.

[26] R. L. Morrison, "Symmetries that simplify the design of spot array phase gratings," *Journal of the Optical Society of America A*, vol. 9, no. 3, pp. 464–471, 1992.

[27] P. R. Gill, "Odd-symmetry phase gratings produce optical nulls uniquely insensitive to wavelength and depth," *Optics Letters*, vol. 38, no. 12, pp. 2074–2076, 2013.

[28] R. Penrose and J. A. Todd, "On best approximate solutions of linear matrix equations," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 52, pp. 17–19, 1956.

[29] D. G. Manolakis, V. K. Ingle, and S. M. Kogon, *Statistical and adaptive signal processing: Spectral estimation, signal modeling, adaptive filtering and array processing*. Norwood, MA: Artech, 2005.

[30] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of statistical learning: Data mining, inference, and prediction*. New York, NY: Springer, 2009.

[31] D. A. Fish, A. M. Brinicombe, E. R. Pike, and J. G. Walker, "Blind deconvolution by means of the Richardson-Lucy algorithm," *Journal of the Optical Society of America A*, vol. 12, no. 1, pp. 58–65, 1995.

[32] D. G. Stork, "Joint optics/signal processing design for computational diffractive sensing and imaging," in *Computational Optical Sensing and Imaging (COSI)*, Kohala Coast, HI, 2014.

[33] D. Tseng, O. Mudanyali, C. Oztoprak, S. O. Isikman, I. Sencan, O. Yaglidere, and A. Ozcan, "Lensfree microscopy on a cellphone," *Lab on a chip*, vol. 14, pp. 1787–1792, 2010.

[34] H. Chaouchi, Ed., *The Internet of Things: Connecting objects*. New York, NY: Wiley, 2010.

[35] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY: Wiley, 2001.

[36] D. G. Stork and P. R. Gill, "Reading QR code symbols with an ultra-miniature computational diffractive imager," in *Proceedings of Computational Optical Sensing and Imaging (COSI)*, Kohala Coast, HI, 2014.

[37] S. Z. Li and A. K. Jain, Eds., *Handbook of face recognition*. New York, NY: Springer, 2005.

Cartesian versus Newtonian Paradigms for Recursive Program Synthesis

Marta Franova
LRI, UMR8623 du CNRS & INRIA Saclay
Orsay, France
mf@lri.fr

***Abstract* — In this paper, we bring a new solution to two unusual questions in Computer Science relative to recursive Program Synthesis (PS). To clarify our ideas we introduce the concepts of Newtonian and Cartesian paradigms to scientific creativity when related to PS. The main contribution of the paper is a thorough discussion on the difference between disruptive Cartesian creation and classical Newtonian construction of a theorem prover devoted to PS. We illustrate these ideas by an analysis of Peano's axioms defining the set of non negative integers, from the point of view of creativity and we explain why Newtonian systemic creativity is not suited for conceiving this simple recursive system. This analysis is then applied to a more complex case of the general framework for our own 'Constructive Matching Methodology' (*CMM*) as a Cartesian paradigm to the creation of an autonomous theorem prover for PS. This methodology illustrates that Cartesian Intuitionism can be viewed as a 'generator of new ideas'.**

***Keywords - evolving systems; Cartesian Intuitionism; Newtonian construction; Cartesian creation;* CMM.**

I. INTRODUCTION

Autonomous Program Synthesis is a desirable goal even though, in case of synthesis of recursive programs, it is recognized as a theoretically inaccessible one. After thirty years of experiments and deep systemic and epistemological studies to build solid justifications for new pragmatic foundations, we were able, in [1] and [2], to launch a clearly defined new approach. This paper goes deeper into the fundamentals of our approach. These fundamentals are useful for all who are concerned by systemic scientific creativity in their work.

There are two main ways to tackle with recursive Program Synthesis, namely inductive and deductive. Automatic construction of programs speeds up the conception process and, in the case of deductive way, it guarantees the correctness of synthesized programs. Therefore, in this paper we are interested in the deductive approach to Program Synthesis (PS) introduced by Manna and Waldinger in the eighties [57] and followed by many authors, for instance [10], [64], [32], [11], [25], [59], [61] [18], [30], [55]. This problem is however undecidable as a consequence of Gödel's Theorems [51]. In this paper, we shall present an attempt to, as much as possible, approximate the automation of the deductive approach to PS by introducing the conceptual switch of 'Cartesian Intuitionism', described in the book [41] in an informal way and presented shortly in [2] and [1]. This paradigm is, from an epistemological point of view, an interesting and even necessary complement to the more formal Newtonian paradigms. From a practical point of view, by introducing concepts that are disruptive in Newtonian paradigm, Cartesian Intuitionism improves the rigor of communication and increases the creative potential of researchers in various domains not only in those related to PS.

Since dealing with existentially quantified variables in inductive proof is recognized by scientific community as a difficult problem (see [13], [17]), it is still too soon to compare the application of Cartesian and the Newtonian paradigms in PS on performance basis. However, our presentation in this paper will show how a somewhat disruptive but pragmatically and epistemologically justified conceptual switch (or 'epistemological rupture', as Gaston Bachelard says in [4]), may change the perspective of the focus in conceiving a PS system and thus enlarge and improve not only a frame of thought of the creators of a PS system but also of a user of a theorem prover in the process of recovery from a failure.

The paper is structured as follows.

In Section II, we recall the formulation of the deductive paradigm to PS and we present two basic problems and two unusual questions related to PS. We present a new and disruptive way of perceiving the limitations determined by Gödel [51]. This disruptive way is justified in the epistemological (rather than mathematical) Cartesian Intuitionism we present in this paper. In Section III, we present the main features of Newtonian and Cartesian paradigms to scientific creativity related to PS. In Section IV, we use the example of Peano's axioms in order to underline the deep gap between Cartesian creating a set of axioms, and Newtonian making use of a given set of axioms. This detailed example enables us to precise what is the difference between Newtonian synergetic *construction* and Cartesian symbiotic *creation* of a system. In Section V, we recall the basic notions of Cartesian Intuitionism illustrated in Sections III and IV. We shall devote Section VI to the description of our Constructive Matching Methodology (*CMM*) in the light of Cartesian Intuitionism. In particular, we describe a technique called *CM*-formula construction that is a strategic basis not only in conceiving inductive proofs typical for the deductive paradigm but also in conceiving our whole PS system. In Section VII, we present the main drawbacks and the main advantages of our approach in comparison with Newtonian approaches.

II. PROGRAM SYNTHESIS

A. Definition of the Deductive Approach to Program Synthesis

By Program Synthesis (PS) we call here the deductive approach to automatic construction of recursive programs introduced in [57]. This paradigm starts with a specification formula of the form

$$\forall x \, \exists z \, \{P(x) \Rightarrow R(x,z)\},$$

where x is a vector of input variables, z is a vector of output variables, P(x) is the input condition. R(x,z) is a quantifiers-free formula and expresses the input-output relation, i.e., what the synthesized program should do. For instance, let us suppose that 'member' is a predicate deciding whether a natural number is an element of a given list and 'ltl' is a predicate that decides whether a given natural number is less than or equal to all elements of a given list. Then,

$$\forall x \in \text{LIST} \, \exists z \in N \, \{x \neq \text{nil} \Rightarrow \text{member}(z,x) \, \& \, \text{ltl}(z,x)\},$$

is a specification formula for a minimum of a list of natural numbers.

A proof by recursion of a specification formula, when successful, provides a program for the Skolem function sf that represents this program, i.e., R(x,sf(x)) holds for all x such that P(x) is verified. In other words, PS transforms the problem of program construction into a particular theorem proving problem.

The role of the deductive approach is thus to build an inductive theorem prover specialized for specification formulas (ITPPS).

B. Problems

There are two main problems with respect to the goal to build an inductive theorem prover specialized for specification formulas:

(1) treatment of strategic aspects of inductive theorem proving system specialized for specification formulae,
(2) treatment of strategic aspects of creativity related to the design of such theorem prover.

As to (1), there is the above mentioned limitation determined by Gödel [51]. Because of the practical importance of PS, to build an ITPPS, standard approaches to PS use this worst-case limitation as an argument for adapting already existing mechanisms that may too be undecidable such as general term rewriting systems (see [31]), rippling (see [15]) or SMT (see [24]).

To our best knowledge the problem (2) was not yet treated in Computer Science. We think that it is so simply because, as we have just mentioned, researchers prefer adapt already existing tools to PS instead of asking two questions:

a) Can the logical limits of Gödel's results be 'overcome' by a pragmatic reformulation of PS problem?
b) Can there be a custom-designed theorem prover for PS?

We have asked these questions in eighties and our work is directed by these questions since. This is why this paper is concerned mainly with (2), which puts then (1) in another perspective. In the following sub-section we present our argument in favour of positive answer for a) and then we shall proceed to an extensive answer for b).

C. A disruptive idea to 'overcome' limitations of Gödel's results

The goal of this section is to present a new pragmatic interpretation of Gödel's results. It is in no way intended as challenging Gödel's results. In other words, Gödel results hold also in this new paradigm. However, they have a stimulation effect instead of paralysis one. Understanding this new pragmatic interpretation is necessary for understanding the remaining parts of this paper.

First, let us recall what are the limitations specified by Gödel's results [51].

The first limitation is the total incompleteness result concerning natural numbers N. This practically means that there is a true statement F such that both F and not(F) can neither be proved nor disproved in N. Moreover, if F is added to the axioms defining N then there can still be found a new formula that is undecidable in this new system. And this holds *ad infinitum*.

The second limitation is the affirmation that there is no finite decision procedure for proving or disproving all formulae. This practically means that there is no deductive algorithm that could decide in a finite time whether an arbitrary formula G is true or false.

Let us consider the first limitation. What does incompleteness means practically? We have a very simple illustration for this problem in fifth Euclid's postulate (postulate for parallels) for geometry. For a long time mathematicians could not decide whether this postulate really is necessary for defining the usual geometry, i.e., whether the first four Euclid's postulates form a complete axiomatic system. It is only in 19th century that Lobachevski and Bolyai showed that when only first four postulates are considered, one can add to them one of negations of the fifth postulate and obtain thus new geometries completely different from that specified by Euclid. Nevertheless, while the notion of the straight line exists in all geometries, it *looks* differently in each of them. Similarly, in all geometries there exists the notion of triangle. However, in non-Euclidian geometries the sum of its angles is greater or less than 180°. So these triangles *look* differently from the Euclidian's one. This means that one postulate (in the case of the geometry the fifths one) can completely modify the perception of an incomplete system. What is the link to natural numbers? The incompleteness of N means that presently even banks use for computations a system of calculus which, for the same problem, can have different values for different banks such as we have seen for sum of angles of a triangle in different geometries. Nevertheless, there is a 'faith' that such situation cannot happen. It is thus possible that we all believe in some kind of 'practical completeness' of our natural numbers. Using this incomplete system we all believe that the formulae independent of N are somewhat properties of N that we do not need, that they are more a 'toy' for mathematicians to keep them busy in employing the undecidability results. More seriously now, as we pointed out previously, we do not suggest that the problem of

undecidability does not exist. What we try only to point out that if possible change of N will occur, we shall (or we should) simply 'be ready to deal with the situation' as we are used to be with our changing times. What it means for PS?

There are two cases to be considered but the solution is pragmatically similar in both cases:

- an incomplete axiomatic system with respect to which a specification formula is given
- an incomplete ITPPS system that provides proofs for specification formulae is built

Let us consider the first case.

We enlarge our view here by focusing not only to the consideration of incomplete system N, but to any incomplete theory T.

Classical way to the PS problem is to develop decision procedures for specification formulae. Decision procedures are interested only in providing one of the two possible answers (TRUE or FALSE). Such procedures are thus unsuitable to deal with the failure cases due to the incompleteness of T. Cartesian way is to build a 'construction' procedure which, in case of failure due to the incompleteness of T provides a suggestion for missing axioms. These axioms have then to be approved by the user who knows (or should know) by which model he wants to complete T and thus these missing axioms or new ones proposed by the user are added to T. In [48] and [40], we have presented a successful solving of a simple example in robotics that suggests two missing and immediately useful axioms for the given incomplete description of the problem. This is why this constructive Cartesian paradigm seems promising.

The classical way (building decision procedures) can thus be formalized in the following way:

$$\exists \text{ Theory } \forall \text{ Specification Formula}$$
$$\text{Has\_a\_solution\_in(Theory, Specification Formula).}$$

This means that the classical decision procedures are restricted to considering only one theory and this is another reason why they are not well suited to handle failures when the given theory is incomplete.

The Cartesian way (building a construction procedure instead of a decision procedure) can be formalized in the following way:

$$\forall \text{ Specification Formula } \exists \text{ Theory}$$
$$\text{Has\_a\_solution\_in(Theory, Specification Formula)}$$

This formalization says that the construction theorem proving procedure builds up the theory at the same time as it constructs the proof for the specification formula.

It means that instead of fixing our focus on building one closed system and arguing that such a system cannot exist, what is a mathematical truth, we change our focus to building 'evolving' systems that are changed when a necessity brings a formula by which N (or a given theory T) has to be completed. Formally, this can be expressed as a change from the classical formulation of PS problem:

$$\exists \text{ PS-System } \forall \text{ Specification Formula}$$
$$\text{Solves(PS-System, Specification Formula)}$$

to 'Cartesian' formulation that oscillates without much difficulties between this classical formulation and the following disruptive one:

$$\forall \text{ Specification Formula } \exists \text{ PS-System}$$
$$\text{Solves(PS-System, Specification Formula)}$$

We say that such an oscillation will not be a reason for unbearable difficulty since difficulties are here good for learning and discovering new paradigms and sustaining opportunities.

Once such an opening of our perspective is accepted, we can open our perspective even more as we shall show later in this paper.

As far as the second limitation is concerned (namely that there is no algorithm for a decision procedure handling PS), we first need to describe this limitation in a more pragmatic way. Gödel's results concern dealing with the formal theories in which such a decision procedure should be expressed (and, in fact, it cannot be). Without neglecting the necessary rigor in formulating an 'algorithm' for proving the specification formulae in the complete theories, we suggest that some creative features of human's mathematical brain are exploited when custom-designing an ITPPS procedure. We suggest here developing custom-specified machine learning (computational creativity) techniques. This means that we shall no more be allowed to employ the word 'algorithm' for this procedure, however, we can speak about an artificially intelligent procedure or technology for PS. In short, we shall speak of a technology and not of an algorithm. This means that we shall no more try to find an 'approximation' of a decision procedure, but we shall use our brain to invent a custom designed evolving technology.

We have thus introduced two features by which the Cartesian paradigm differs from the classical Newtonian one.

- First, as a response to the incompleteness results, to consider evolving systems instead of closed ones.
- Second, as a response to the restriction of purely formal framework, to consider a custom-designed artificially intelligent technology instead of formal decision procedures.

At a first glance our suggestions may seem too disruptive. This is why, in the next sections, we are going to give an epistemological justification provided by Cartesian Intuitionism rediscovered by our study of Descartes' work [41]. In contrast to a logical justification that provides a logical proof for a considered hypothesis, an epistemological justification consists in giving arguments confirming a reasonable character of the hypothesis and, if possible, in giving references to recognized predecessors. In our case, the predecessors are Francis Bacon by his idea of recursive long-term Progress and René Descartes by his development of Cartesian Intuitionism. Cartesian Intuitionism is counter-intuitive in usual thinking and this is also the reason why philosophical commentators of Descartes' work explained it in terms of the linear systems. This makes our task of transmitting Cartesian Intuitionism more difficult since we lack contemporary supporters. This means that the access to Cartesian Intuitionism is not an easy one and in the next

sections we give the reader an opportunity to understand why it is so. This means also that we need to present the basic notions of Cartesian Intuitionism intertwined with examples and only then, in Section V, we give a recollection of the basic notions used.

III. NEWTONIAN AND CARTESIAN WAY OF CONCEPTION NEW SYSTEMS

The main difference between Newtonian and Cartesian paradigms is easily perceptible from comments pronounced by Newton and Descartes themselves.

Newton wrote: "If I have seen further (than you and Descartes) it is by standing upon the shoulders of Giants."

Newtonian science is thus established on logic of sequential research. In a little more formalized way, we can thus describe the Newtonian way by the sequence

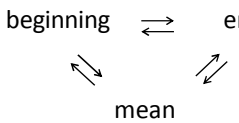
beginning … advancement-1 … advancement-2 … advancement-n … end.

Descartes wrote his first rule in the *Discourse on the Method of Rightly Conducting the Reason, and Seeking Truth in the Sciences* [27] in a following way: "The first was never to accept anything for true which I did not obviously know to be such; that is to say, carefully to avoid precipitancy and prejudice, and to comprise nothing more in my judgement than what was presented to my mind so clearly and distinctly as to exclude all ground of doubt."

Descartes speaks about the obvious truth. As says Descartes' commentator Ferdinand Alquié in [26], the act of thought that seizes the obvious truth is the intuition defined by Descartes in his *Rules for the direction of the mind* (*Regulae ad directionem ingenii* [29]). So, the study of Descartes' intuition, as presented in the book *Formal Creativity* [41] enables to notice that Cartesian science is based on logic of recursive research.

The same thing is expressed by Descartes in a little more complicated way by saying that "beginnings … can be persuaded well only by the knowledge of all the things that follow later; and that these things which follow cannot be understood well, if we do not remember all those that precede them." [26], p. 797. Thus, the Cartesian paradigm takes into account that the demarcation of a notion is not the initial stage but the final stage of its formation.

The Cartesian way can be described by the loop



where the arrow $\rightarrow$ means "leads to". This recursive loop will be illustrated in Section IV by description of the process of the creation of Peano's axioms defining natural numbers.

Thus, there are two basic styles to approach the problem of PS.

A. Newtonian paradigm for Program Synthesis

Newtonian paradigm in conceiving a system means its linear development. As far as PS is concerned it means that the reference system of the conception of a program synthesizer, that is, the axioms, the rules of inference and the mechanism of control of the program synthesizer, as well as the reference system of a given PS problem, that is the theory in which the PS problem has to be solved, are given at the beginning by the past history of scientific research. The Newtonian paradigm in PS takes as foundation the standard knowledge of the mathematical formal framework, which inevitably inherits the negative results of Kurt Gödel. By consulting the first paragraph of Gödel's article *On formally undecidable propositions of Principia Mathematica and related systems I* [51], we can observe that the keywords of this standard knowledge are

- exactness
- formal system justified in a logical way
- methods of demonstration reduced to some axioms and rules of inference
- decision and undecidability

Previously, we have described the Newtonian style by the sequence

beginning … advancement-1 … advancement-2 … advancement-n … end.

Gödel's results are called negative because they show that the aim of synthesis of programs formulated as the "beginning" in the classic framework cannot lead to a successful 'end' of the task. In other words, they show the impossibility to define a formal logical framework containing the natural numbers allowing to approach the resolution (confirm or counter) of specifications given in a general way. Nevertheless, there are approaches to PS in the Newtonian style and they are very interesting from the short term perspective as well as from the point of view of developing long term Cartesian evolving systems.

The best-known paradigms are presented in [57], [64], [11], [10], [25], [59]. Since the problem of proving by induction specification formulas, i.e., formulas containing existential quantifiers is very difficult, researchers focused on the problem of proving purely universally quantified formulas and on treating formulas with existential quantifiers by assisting the users in developing their own proofs. The best known are the system ACL2 [12], the system RRL [54], the system NuPRL [20], the Oyster-Clam system [14], the extensions of ISABELLE [60], [30], the system COQ [9], Analytica [bauer01], KeY [7], HipSpec [19], Zeno [65] and Matita Proof Assistant [3]. All the mentioned approaches have done a very good work in modelling *human reasoning* by exploring possibilities of *transformational* methods to inductive theorem proving and PS. The construction calculus of [21], that is the basis of the system COQ, is a constructive way of *representing* transformational methods. The paradigm presented in the next section attempts to find a *constructive* way of solving an 'almost' same problem by modelling *human creativity* based on Cartesian style of research.

B. Cartesian paradigm for Program Synthesis

Cartesian paradigm for PS is based on a logic of recursive research, where the reference system of the ITPPS

system as well as the reference system of PS problem are formulated hand in hand with the development of the solution, and where the exact demarcation of the both reference systems is the final stage of the process, and is too a part of the solution.

Recall that the Cartesian paradigm takes into account that the demarcation of a notion is not the initial stage but the final stage of its formation. The Cartesian paradigm thus specifies at the beginning the reference system in an informal way only. It is much like a hypothetico-deductive method.

The hypothetico-deductive method is a procedure of construction of a theory that consists in putting, at the start, a certain number of loosely defined concepts or proposals that are obtained by a study of experiments undertaken to specify these starting concepts or hypotheses. Then, by deductive reasoning, are obtained postulates that, when they are true, confirm the effectiveness of chosen concepts and hypotheses. If they are not true, the problem, because of the loose definitions of concepts, allows their new reformulation and the process is thus repeated on these new still loosely defined reformulations.

In contrast to hypothetico-deductive method that proceeds by deductive reasoning to access the 'truth', Cartesian paradigm uses Cartesian Intuition to access to 'truth', i.e., to the final description and justification.

Furthermore, in contrast to Newtonian paradigm and hypothetico-deductive method, in Cartesian style one can specify even the goal in a rather 'vague' manner. This is why we introduced the term of 'quite precise' purpose to indicate that this formulation, though informal, must describe a reasonable project.

For the construction of recursive programs from formal specifications, it is possible to give a 'quite precise' purpose by considering PS as a problem of realization or creation, rather than a decision-making problem. We adopted this paradigm when starting to develop the *Constructive Matching Methodology* (*CMM*) for Program Synthesis in 1983 [32]. In contrast with the Newtonian paradigm, the keywords of our particular Cartesian paradigm are

- realization and creativity
- system justified in an epistemological way
- methodology of construction
- realization of a program or sufficient conditions for the realization of such a program.

The most suitable way is thus to consider *CMM* as a technology (in a general sense) rather than a theory. The next section explains the main differences between a mathematical theory and an epistemological technology from the point of view of Newtonian construction and Cartesian creation.

IV. NEWTONIAN CONSTRUCTION VERSUS CARTESIAN CREATION

In this section, in order to underline the main differences between a Newtonian mathematical theory and an epistemological Cartesian technology, we shall be interested in the set of natural numbers N, seen here as a creation model for particular complex systems. More precisely, we shall point out the difference between the use (Newtonian) and the creation (Cartesian) of Peano's axioms.

Peano's axioms define the arithmetic properties of natural numbers N. These axioms include a constant symbol 0 and unary function symbol S. These axioms are usually used to build formal proofs about natural numbers. This section does not deal with the topic of theorem proving. It deals with the topic of understanding and reasoning about the construction of Peano's axioms, that is the creation process involved in their building.

Supposing that the membership relation "$\in$" and the equality "=" are already defined, the basic Peano's axioms read:

A1. $0 \in N$.
A2. if $n \in N$ then $S(n) \in N$.
A3. for all $n \in N$, $S(n) \neq 0$.
A4. for all $n, m \in N$, if $S(n) = S(m)$, then $n = m$.
A5. if M is a set such that
  - $0 \in M$, and
  - for every $n \in N$, if $n \in M$ then $S(n) \in M$

then M contains every natural number.

In order to tackle the difference between the use and the creation of these five axioms we need to precisely specify the difference between synergy and symbiosis.

An object is constructed *synergistically* when it can be considered as a result of the application of some specific tools from an existing tool-box. This tool-box represents all the tools that have been developed in all scientific domains beforehand and, usually, for various purposes. These tools are not built in such a way that one calls another tool to solve one of its problems before active tool has completed its computations. That is, tool B can call on tool A in one way only: the input of B contains a part of A computations, once A computations have been all achieved. It follows that these tools must be used and constructed independently of each other. The synergic construction is thus the main feature of Newtonian conception of independent modules for which it is meaningful to consider and prove properties independently of the whole system. For instance, the termination of rippling is proved by the team of Alan Bundy in [6], while the second order unification that is used by rippling (see [15]) is not at all considered.

In contrast to this, an object is conceived *symbiotically* when its parts, maybe seemingly independent (as it is the case for lichen that is a symbiotically living fungus and alga), have, during the conception process, no meaning as isolated entities. It means also that a slight change of one part influences the others and the whole as we illustrate below. The symbiotic composition is the main feature of the intuition defined by Descartes in his *Regulae ad directionem ingenii* [29].

Now, what we can underline about Peano's axioms is that their ***use*** is synergetic, while their ***construction*** process is symbiotic. In other words, when ***using*** them, we can use several axioms as being independent entities and the

constructing elements 0, S, and N can be considered as isolated from each other, though they are interdependent elements as show A1 and A2. The following example will show in which way Peano's axioms construction process is of symbiotic nature.

Let us first consider axiom A1 dealing with 0 and N. However, the full meaning neither of 0 nor of N is explained in this first axiom. (Recall that in hypothetico-deductive method the first notions, at the beginning, may be specified in a vague manner.) In particular, from this axiom we cannot conclude that 0 is a basic element and that N is the final object we want to define. The axiom A1 expresses only an interdependence between two symbols 0 and N. The symbol $\in$ does not tell more than 0 is an "element" and N is one of sets to which this element belongs. There is no difference, except substitution, between A1 and B1: "rose $\in$ garden". This means that the creator of Peano's axioms has already in mind a "vision" or an "informal specification" (or, as we say, a 'quite precise' purpose) of what 0 and N mean for him in this first axiom. This is why, in the cyclic presentation of Cartesian thinking (see Section III), there are two arrows, one linking beginning to the end and one doing the reverse. In other words, writing this first axiom, the axiom's creator intuitively knows what 0 and N will be once their description has been completed, i.e., when all the necessary (in this case five) axioms will be provided. In the creator's mind, the first axiom contains implicitly and intuitively all the remaining axioms and all the axioms are constructed from his/her intuitive vision of the "whole", i.e., N. Therefore, 0 and S do not belong to an already given tool-box and the meaning of 0, S and N in the construction process *is custom-made*. Moreover, 0, S, and N are symbiotic during the construction process and they are not synergetic parts. During the construction process, N steers the realization of 0 and S and vice versa, they cannot be considered as isolated already known elements. In this sense, the Newtonian paradigm is unable to provide and explain the process of creation of N and others systems that rely on Cartesian Paradigm. This is also why we say that N is a complex system, even if its description is short one.

We shall below present an example illustrating this symbiotic feature. However, we need first to introduce some more notions.

N is constructed with the help of three "elements", namely 0, S and N itself. Note that self-reference is already acknowledged as a constructive recursive 'trick'. (Look in Section III for the presence of the 'mean' in Cartesian recursive cyclic thinking). These construction parts are usually named 'the constructors'. We have already mentioned that these parts are symbiotic during the construction process, while when using the Peano's axioms for reasoning, we may consider them synergetic "*par la pensée*" (as Descartes puts it §62 of *The Principles of philosophy* [28]). In the following, instead of 'construction' we shall call this process 'Cartesian creation' in tribute to Descartes.

Now we can illustrate the symbiotic character of the constructors 0, S and N. Let us consider Peano's axioms without A3. In such a case we have the liberty to suppose that there exists $n \in N$ such that $S(n) = 0$. Let us suppose that $S(S(0))$ is such an element. We have then $S(S(S(0))) = 0$. Let us call B3 this hypothesis. Then, A1, A2, B3, A4 and A5 constitute a meaningful definition of the set that contains three elements, namely 0, S(0) and S(S(0)). This new axiomatic definition defines a set, N3, which is finite and thus is different from the infinite set N defined by Peano's axioms. In other words, a *little change* in a property of one constructor (as we have see also in the example of geometry) altered the properties of all the constructors, including N that changed into N3. This is not the case in a synergetic construction, where a change of one construction module may influence the behaviour of the whole but has no direct effect on the other modules. This explains why we so much stress the difference between symbiotic Cartesian creation and synergetic Newtonian construction. Once a symbiotic creation of a whole is completed, we may *think* of the constructors as being "unconnected" synergetic elements. (This is also the reason why Descartes' epistemological work is misunderstood and explained in terms of linear thinking and analysis, see our critics of [56] in [41]). We just have shown that this synergetic thinking is not valid during the creation process. This is why there is also a difference between a creation process and the use of the completed whole created by the same process. Descartes specified this difference in his notions of clear and distinct perception [28]. A clear perception is typical for perception and use of synergetic systems, while clear and distinct perception is imperative for symbiotic systems.

An interesting feature of a symbiotic creation is that one cannot produce a sample or "architectural" miniature before the whole creation process is completed. Moreover, partial results are often incomprehensible outside the creation process, which works mainly with informally specified problems that must be simultaneously solved. The drawbacks we just exposed must be one of the reasons why Cartesian creation is hardly reported in the scientific communications that concentrate on the results of the creation, not on the creative process itself. Researchers (and/or referees) seem to prefer tool-box Newtonian progressive construction that provides the security of familiarity with such linear or modular processes as well as immediate gratifications. This may also explain why our original Cartesian paradigm is not followed in the research on PS.

Summarizing this section, we can say that Cartesian creation focuses on building a system, a whole, by progressively inventing symbiotic constructors. Such a progressive process is possible since the first constructors and the whole are described by a 'mere' informal specification. The standard Newtonian research is not accustomed to such an informal goal specification and it usually gathers already existing mechanisms that have been certainly not custom-designed for the given goal. This choice

leads, during the construction process, to new problems, more often related to the chosen basic tools than to the given goal (we can mention the use of the second order unification in rippling [15]). These new problems ask for a new search of already existing tools and to attempts for adapting them to the given goal, a process that tends to fail when it is completely automated. In other words, in Cartesian creation, the basic tools, i.e., constructors and the whole system are custom-made, while in Newtonian construction, the basic words are "choice" and "adaptation" of already available tools.

V. CARTESIAN INTUITIONISM

Cartesian Intuitionism is specified by Descartes in his work mainly by four disruptive notions and the rules of his method. Namely, we have:

- a form of constructive symbiotic creation called **intuition**, in the Latin version of his *Rules for the direction of the mind* [29];
- the ability of thinking as isolated, one of many mutually dependant features (**division *'par la pensée'***) in §62 of *The principles of the philosophy* [28];
- **clear** and **distinct perception** in §45 and §46 of *The principles of the philosophy*[28];
- the **four rules of his method**, in his *Discourse on the method* [27].

These notions and rules are disruptive since they differ from linear, analytical, rigid and unemotional thinking that is usually attributed to Descartes (see, for instance, [56], [22], [23]).

The thinking of Descartes is not linear as we have illustrated by the quotation of Descartes before the recursive loop in Section III. However, the fact that his thinking is recursive is illustrated best by his method. Namely, one should ask the question: "How is his method obtained?" And the (not so) obvious answer is that *his method is conceived by his method*. This contradicts Popper who claims, in [62], that there can be no logical description of inventing new ideas. If one accepts that Descartes' notion of intuition is a logical way of inventing new ideas and that the Descartes' method describes this way, then Popper's opinion is challenged.

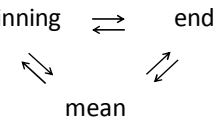
While the Descartes' thinking comprises also analysis (synergy), it is highly symbiotic. This manifests in his recursive creation, the notions of intuition (the symbiotic creation), division '*par la pensée*' and distinct perception.

Descartes' thinking is not rigid since the idea of evolving systems is comprised in the possibility of 'divine revelations' (in the rule II of his *Rules for the direction of the mind*) that have to be 'assimilated' to existing knowledge by Cartesian Intuition and deduction.

Descartes' thinking is not unemotional, as the rule XII of his *Rules for the direction of the mind* insists on employing all possible human resources in conceiving an exploitable evolving system. From a pragmatic point of view, the emotions are hidden in our technological context in the notion of 'trust' and 'faith'. With respect to its large use, Newtonian conception is highly trusted since partial results are measurable in usual ways. However, Cartesian creation cannot be easily understood and measured (thus trusted) by an external observer requesting simple explanations in Newtonian terms and measures. Partial results in Cartesian creation are more-less informal 'chunks' possibly intertwined with other 'chunks' to be yet specified as it is written in XII rule of *Rules for the direction of the mind*. On the other hand, the notion of 'faith' is, in recursive Cartesian thinking, a *technical* term that expresses the conviction about the reasonable and realisable character of the goal and about the soundness and the appropriateness of the method employed for accessing to the goal.

We can here summarize Cartesian creativity representing the Cartesian Intuitionism in three points. Cartesian creativity

(a) focalises on the problem: {∀ specification formula ∃ framework in which the given specification formula has a solution}
(b) oscillates between the problems {∃ framework ∀ specification} and {∀ specification ∃ framework}
(c) considers the creativity process in its recursive cyclic version given by the scheme



where the arrow means "steers".

These three points give to Cartesian Intuitionism the feature of a combination of what is called essentialism and existentialism within the frame of logics by Girard in [50].

VI. CONSTRUCTIVE MATCHING METHODOLOGY

In this section we are going to

- illustrate some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and the difference between a Newtonian decision and Cartesian construction procedure;
- explain how the idea of evolving systems is actually performed in *CMM*;
- present an informal description of the basic constructor of *CMM*;
- present assessment and perspectives of *CMM*.

A. CMM *in the light of Cartesian Intuitionism*

The basic principle of Newtonian PS system is the use of a fixed set of specific strategies in order to solve the problems that are submitted to it. In case of failure, the user is requested to provide lemmas or axioms that lead to success.

The basic principle of Cartesian PS system is also the use of a specific strategy defined by the axioms, which themselves represent the whole system. But this is true only as long as the system meets no failure. In case of failure, we build a new PS system possibly with a new solving strategy.

We already illustrated such behaviour by building the pseudo-Peano system by replacing A3 by B3 and N by N3. If this kind of incomplete natural numbers is used to prove a theorem containing the term, say S(S(S(S(0)))), the 'synthesis' will fail. In a Newtonian paradigm, the user would be asked for a lemma specific to S(S(S(S(0)))) that enables a success. In such a case our paradigm would propose to modify the system of axioms by changing B3 and N3. We fully agree that, in this particular case, a human feels the needed modification as being trivial. In consequence, let us provide a more complex example that illustrates a situation where modifying system of axioms defining PS mechanism is not trivial.

In [8], a Newtonian system called Otter-Lambda is presented, together with several examples of its execution. We have chosen among them a formula

$$\forall\, a\ \forall n\ \{\ (S(0) < a \Rightarrow n < \exp(a,n))\ \} \qquad (*)$$

The Otter-Lambda system fails when the basic information relative to (*) is given as a recursive definition of the exponentiation function exp with respect to the second argument:

(A1) $\exp(u,0) = s(0)$
(A2) $\exp(u,S(v)) = \exp(u,v)*u$

of the addition and of the multiplication with respect to the first argument:

(A3) $0 + u = u$
(A4) $S(v) + u = S(v + u)$
(A5) $0 * u = 0$
(A6) $S(v) * u = (v * u) + u$

The definition of < is also recursive and given as:

(A7) $0 < y$, if $y \neq 0$
(A8) $S(v) < y$, if $v < y \ \&\ y \neq S(v)$

Since the Otter-Lambda system fails, it requests some help from its human user. In [8], the user is able to provide the following lemmas that enable Otter-Lambda to complete the proof of (*).

(A9) $not(u<v)$ or $(x*u < x*v)$ or $not(0 < x)$
(A10) $(x < y)$ or $(y \leq x)$
(A11) $not(y \leq x)$ or $not(x < y)$
(A12) $not(u < v)$ or not $(v \leq w)$ or $(u < v)$
(A13) $not(S(0) < z)$ or $not(0 < y)$ or $(S(y) \leq z*y)$
(A14) $0 + x = x$

We applied to the same problem our Cartesian paradigm, which does not suggest getting any user's help. The system determines n as the induction variable, since it occurs in recursive arguments of all the functions and predicates and the other possible candidate variable a occurs in the non-recursive first argument of the function exp, which would stop the evaluation process in an inductive proof. Nevertheless, our method notices at once a probable source of trouble: the predicate < is defined recursively with respect to its first argument, while, in (*), the induction variable n occurs also in second position of the predicate <. At this stage, the method could suggest the user to provide a definition of < with respect to both argument (this would actually fail), or with respect to the second argument (this would fail as well), or else, a non recursive definition (that would succeed). As already mentioned, our method is not expected to call on its user, and thus it will proceed by calling a custom-designed constructor named "Synthesis of Formal Specifications of Predicates". The initial results in developing this constructor are described in [49]. The symbiotic system *CMM* with this constructor included generates the following formal specification for predicate <:

$$x < y \Leftrightarrow \{\ \exists z\ y = S(x + z)\ \}.$$

With this new definition (*) is transformed into

$$\forall a\ \forall n\ \exists z\ \{\ (S(0) < a) \Rightarrow (\exp(a,n) = S(n + z))\ \}. \qquad (**)$$

Note that this last formula is a specification formula by introducing the existentially quantified variable z. *CMM* is then able to prove it (without interaction with the user). *CMM* generates and proves autonomously the following lemmas (that are formal specifications for six auxiliary sub-routines of the program specified by (**)):

L1. $\forall\, a\ \forall n1\ \forall b\ \exists z1\ \{\ S(0) < a \Rightarrow (n1 + b)*a + a = SS(n1 + z1)\ \}$.
L2. $\forall\, a\ \forall b\ \exists z2\ \{\ S(0) < a \Rightarrow b*a + a = SS(z2)\ \}$.
L3. $\forall\, a\ \exists z7\ \{\ S(0) < a \Rightarrow a = SS(z7)\ \}$.
L4. $\forall\, a\ \forall m\ \forall d\ \exists z5\ \{\ S(0) < a \Rightarrow (m + d) + a = S(m + z5)\ \}$.
L5. $\forall\, a\ \forall d\ \exists z3\ \{\ S(0) < a \Rightarrow d + a = S(z3)\ \}$.
L6. $\forall\, a\ \exists z4\ \{\ S(0) < a \Rightarrow a = S(z4)\ \}$.

This example illustrates all three points (a), (b), (c) of Cartesian Intuitionism in that, when meeting failure, a need for a complementary constructor transforming a recursive definition of a predicate into a non-recursive equivalent is informally specified. Then, the successful formalized design of this constructor enlarges the power of *CMM* and thus modifies the whole *CMM* that is ready, when necessary, to be once again modified.

The basic constructor of *CMM* is presented in [2]. With respect to the notions introduced in this paper, we readapt that presentation in Section VI.C. The other constructors of *CMM* specified so far are described in our publications up to 2001 [38]. Some of these constructors were implemented in the system *Proofs Educed by Constructive Matching for Synthesis* (PRECOMAS) [36], [39]. With respect to the symbiotic character of the constructors and the need of treating the failure analysis by developing further constructors, we have interrupted the implementation of PRECOMAS in 1990 and focused on developing an epistemic justification (see [41]) hand in hand by reformulating our work in terms of this justification.

The example presented in this section helps us to illustrate some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and the difference between a Newtonian decision and Cartesian construction procedure.

First of all, the development of *CMM* is, in this stage, by-hand made. This is because we seek for a methodology, i.e., a conceptual capture of the all problems that are related to inductive theorem proving viewed as a construction procedure. We seek (by-hand) for all the constructors of the resulting system by the on-purpose justified Cartesian method called *Formal Creativity* and described in our book [41]. Classical Newtonian approaches focus on implementing procedures that are checked out with respect

to some benchmark formulas. The systems are considered as failing when they do not provide a decision in some time constraint. For instance, [53] refers to experiments in which timeout is set to 30 seconds. The failure of the system is in this sense unproductive for further research in inductive theorem proving. This is why the Newtonian research is very quick in producing implementations but slow in providing conceptual descriptions of the problems that could point out the directions in which the research has to be done. As we mention in the next section, our by-hand research allowed us to formulate already several major problems.

Second, instead of a modular system for which the properties of modules are formulated and proved independently of the whole system, the Cartesian approach allows us to consider the whole system as an axiomatic system for which, as for Peano's axioms, there can be only a pragmatic justification expressed somewhat unscientifically by the sentence: 'The justification of the system is obvious as it was conceived in such a way that it works'. However, this justification is scientifically valid when one looks at it from the point of view of Cartesian notion of Intuition obtained by (and representing itself) a 'luminous calculus' (see rule II in *Regulae ad directionem ingenii* [29] and Bacon's 'luminous experiment' referred to in *Novum Organum* [5]). Because of its powerful potential for generating new ideas (similarly to lateral thinking [23]), the term 'luminous' should thus become actual even today in all scientific research.

Third, since Cartesian Intuitionism justifies employing all possible human resources, *CMM* relies heavily on the idea of using machine learning (computational creativity) techniques whenever it will be appropriate.

Fourth, as we shall illustrate below, our approach generates multiple auxiliary procedures. This is not possible with second order unification that is able, as in rippling ([52], [15]), to generate auxiliary procedures on one level only (i.e., during the execution, the unification does not generate further auxiliary procedures) and only with already defined functions.

B. Conceptual oscillation of CMM

As we suggested in Section II, we are interested in conceiving evolving systems. Such systems are conceived in oscillatory way. We call oscillatory a paradigm in which, to find an optimal result of a definition of a theory, we oscillate between both specifications of the problem

{∃solution ∀problem} and {∀problem ∃solution}

More exactly, our paradigm oscillates between a Newtonian formulation of PS and a Cartesian formulation of the same problem. It is clear that this purpose seems very ambitious when one forgets the preliminary restrictions (not considering efficiency of synthesized programs, proofs by structural induction only, specifications formulae expressed as conjunctions of atomic formulae and even more restrictions that may come out in a further elaboration). These restrictions do not make the problem trivial; they only enable to focus on the core of the problem that we must specify and solve at first.

In practice, this oscillation is performed in the following way. For a given specification formula, we attempt to perform a constructive proof relying on the results already achieved by *CMM*. In other words, we start to solve the problem having in mind the specification '∃solution ∀problem', where the solution is the *CMM* and the problem is the given specification formula. If the power of the *CMM* is not sufficient to prove the given specification formula, by a failure analysis we try to conceptualize the problems met as methods rather than heuristics. In other words, we solve the problem by focusing on the problem '∀problem ∃solution' and then by a suitable process of conceptualization similar to hypothetico-deductive method we try to come back to the specification '∃solution ∀problem', where the solution is now the extended *CMM*. This is why this paradigm is more the one of a mathematician trying to build a new theory-technology rather than that of a programmer focusing on obtaining efficient programs.

In this way, we have conceptualized many new methods in inductive theorem proving for specification formulas, for instance: implicative generalization, predicate synthesis from formal specification, synthesis of formal specifications of predicates, introduction of universally quantified induction hypotheses whenever appropriate, a particular evaluation tool and a particular equation solving tool. We explain this conceptual richness of inspirations of *CMM* proofs by the basic method for constructing atomic formulas '*CM*-formula construction' that has been introduced in [33] and the most complete presentation of which can be found in [40]. At present we are working on a general algorithmic presentation. In contrast to the basic methods in Newtonian paradigms that rely on simplification and rewriting, our *CM*-formula construction is a constructive method and thus it is very suitable for generating missing lemmas (see Section VI.A) and even axioms when the given data are incomplete as it is illustrated in [48]. *CMM* is even suitable for proving purely universally quantified theorems even if the proofs are generally more complicated, since the basic method is construction and not simplification. The advantage lies however in the fact that, during a proof of a universally quantified formula, a formula containing existential quantifiers can be generated, which replaces the problem of unification in the framework of PS and thus it seems to be conceptually more powerful.

C. CM-*formula construction*

Formulation

In the following, for simplicity, let us suppose that the formula to be proven has two arguments, that is to say that we need to prove that $F(t_1,t_2)$ is true, where F is the given theorem. We introduce a new type of argument in the atomic formula, which has to be proven true. We call them **pivotal arguments**, since the focus on them allows reducing what is usually called the search space of the proof. These arguments are denoted by ξ (or ξ' etc.) in the following. The pivotal argument replaces, in the first step, in a purely syntactical way, one of the arguments of the given formula. The first problem is thus choosing which of the arguments will be replaced by a pivotal argument ξ.

In the first step, let us suppose that we have chosen to work with $F(t_1,\xi)$. In an artificial, but custom-made manner,

we state C = {ξ | $F(t_1,\xi)$ is true}. Except the syntactical similarity with the formula to be proven, there is no semantic consideration in saying that $F(t_1,\xi)$ is true. It simply represents a 'quite-precise' purpose of trying to go from $F(t_1,\xi)$ to $F(t_1,t_2)$. We thus propose a 'detour' that will enable us to prove also the theorems that cannot be directly proven by the so-called simplification methods, i.e., without this 'detour'. In the second step, via the definition of F and those involved in the formulation of the term t_1, we look for the features shown by all the ξ such that $F(t_1,\xi)$ is true. Given the axioms defining F and the functions occurring in t_1, we are able to obtain a set C_1 expressing the conditions on the set { ξ } for which $F(t_1,\xi)$ is true. In other words, calling 'cond' these conditions and C_1 the set of the ξ such that cond(ξ) is true, we define C_1 by $C_1 = \{\xi \mid cond(\xi)\}$. We can also say that, with the help of the given axioms, we build a 'cond' such that the formula: $\forall \xi \in C_1$, $F(t_1,\xi)$ is true. In the third step, using the characteristics of C_1 obtained in the second step, the induction hypothesis is applied. Thus, we build a form of ξ such that $F(t_1,\xi)$ is related to $F(t_1,t_2)$ by using the induction hypothesis. For the sake of clarity, let us call ξ_C the result of applying the induction hypothesis to C_1 and C_2 so obtained is thus such that $F(t_1,\xi_C)$ is true. We are still left with a hard work to do: prove that t_2 belongs to C_2, i.e., to prove that ξ_C and t_2 can be made identical, i.e., that t_2 matches ξ_C. In the case of the success, this completes the proof. In the case of a failure, a new lemma $\xi_C = t_2$ with an appropriate quantification of the involved variables is generated. In some cases, an infinite sequence of lemmas may be generated. *CMM* is conceived in such a way that the obtained sequence is well-behaving (see [33]) in the sense that one can apply a generalization technique to obtain a more general formula that has to be proved. This formula covers logically the infinite sequence of lemmas and thus it fills the gap that cannot be overcome by purely deductive formal approach to theorem proving.

The works in [39] and [40] give a detailed description of handling the pivotal argument in a rigorous framework. In [2], we illustrate *CM*-formula construction on a simple synthesis of a program for displaying the last element of a non-empty list. This is why we can afford illustrate an incomplete example, namely how *CM*-formula construction generates L1 for (**) from Section VI.A.

Example

The formula (**) reads

$$\forall a\ \forall n\ \exists z\ \{ (S(0) < a) \Rightarrow (exp(a,n) = S(n + z)) \}.$$

The lemma L1 is generated in course of the induction step for (**) and we shall thus focus only on this general case of inductive proof. With respect to the recursive analysis of the given definitions (see Section VI.A), the induction variable here is n. It varies over natural numbers, and so, in the induction step, n = s(n1) for some natural number n1. We shall denote by sf the Skolem function corresponding to the existentially quantified variable z in this formula, i.e., z = sf(n,a).

In the induction step for (**), the method assumes a > S(0) and, since n is represented by S(n1), the induction hypothesis is (see [16])

$$\exists\ e\ exp(a,n1) = S(n1+e). \quad (A)$$

In this induction hypothesis,

$$e = sf(n1,a). \quad (B)$$

Assuming S(0) < a, the goal is to prove

$$z\ exp(a,S(n1)) = S(S(n1)+z). \quad (C)$$

Here, z = sf(S(n1),a). Since the term S(S(n1)+z) contains an existentially quantified variable, namely z, this term becomes the pivotal argument ξ. In the first step, ξ syntactically replaces the term S(S(n1)+z). The method gets an artificially built set

$$C = \{\xi \mid exp(a,S(n1)) = \xi \text{ is true }\}.$$

In the second step, the term exp(a,S(n1)) is evaluated using the axiom (A2). C changes to

$$C_1 = \{\xi \mid exp(a,n1)*a = \xi \text{ is true }\}.$$

In the third step, C_1 becomes semantically related to (**) by the application of the induction hypothesis. By the application of the induction hypothesis the method obtains

$$C_2 = \{\xi_C \mid S(n1+e)*a = \xi_C \text{ is true }\}.$$

This, by the application of (A6) gives

$$C_3 = \{\xi_C \mid (n1+e)*a + a = \xi_C \text{ is true }\}.$$

In the fourth step, the method has to check whether the second term, i.e., S(S(n1+z)), belongs to C_3. This leads to the problem of solving the equation

$$\exists\ z\ (n1+e)*a + a = S(S(n1+z)). \quad (D)$$

This equation cannot by solved by *CM*-term transformer (presented in [35]) and thus the method generates a new lemma.

Since we reserve the name e for existentially quantified variables coming from induction hypotheses, we rename e to b and thus the lemma noted in Section VI.A as L1 is generated, i.e.,

$$\forall\ a\ \forall n1\ \forall b\ \exists z1\ \{ S(0) < a \Rightarrow (n1 + b)*a + a = SS(n1 + z1) \}.$$

Let us denote by sf1 the Skolem function for z1, i.e., z1 = sf1(n1,b,a). By (D) we thus obtain the relation between sf and sf1, namely z in (D) is sf(S(n1),a) = sf1(n1,e,a), which, by (B), gives the partial program

$$sf(S(n1),a) = sf1(n1,sf(n1,a),a), \text{ if } a > S(0). \quad (E)$$

The method is the called recursively to prove L1 and all the lemmas that are generated.

This example illustrates well that *CM*-formula construction is an artificial, custom-made method. It is also useful as a suggestion to use PS in the role of a powerful 'unification' tool. For rather complex problems solved by *CMM* the reader can consult the already mentioned [43] but also [37], [40] and [42].

D. Assessment and perspectives of CMM

The stage relative to the procedure of demonstration was elaborated in all our publications until 2000 [38]. An experimental system called *PRECOMAS* (Proofs Educed by Constructive Matching for Synthesis) showing the soundness of the *CM*-formula construction was implemented in the 90s [36].

The stage relative to the specification of the intermediate lemmas is now in a good shape. It concerns also the scientific domain known as 'computational creativity' [46], [47].

The stage that concerns the clear and distinct perception (in the Cartesian sense) of the targeted strategic recursive axiomatization has begun in the article [44]. It must be improved and pursued by an adequate formalization of different fundamental interrelated problems that are met in the oscillatory design of the recursive systems, namely

- one - multiple (part - whole)
- static - dynamic (permanence - change)
- finite - infinite (visible - invisible)
- complete - incomplete (rigor - creativity).

In Program Synthesis, the problem between a whole and its parts is expressed as a strong and special interdependence between the diverse parts of the system, because a part or the whole can itself assume the failure cases and the weaknesses of the other parts. For example, the failure of a resolution of an equation can call in a recursive way the system for help (as we have illustrated above). Or, the deductive parts of the system can call inductive parts, and vice versa. This particular interdependence is described by Descartes as "the distinction, which is made by the thought" (distinction '*par la pensée*') presented above as "the ability of thinking as isolated, one of many mutually dependant features."

The problem of the oscillation between a static representation and a dynamic representation appears in the process of search and creation of the structures and the mechanisms of the control of proofs. This process oscillates between an already partially formalized shape and an informal shape of a given mechanism (see rule XII in *Regulae ad directionem ingenii* [29]). As we said above, the definitive demarcation that consists in fixing a final version of the mechanism is only made at the end of development of the whole system (i.e., by the Cartesian Intuition).

The problem of the regulation of the finite and the infinite appears in PS especially by the fact that an infinite visible variety of possible formal specifications must be managed by finite invisible structures. In other words, the final system of PS has to represent a finite solution of the infinite problem 'to think of everything at the same time'. So, for this problem, Ackermann's function in an oscillatory version models in a curiously proper way the solution that we envisage for this problem.

The problem of the oscillation between completeness and incompleteness is described in an informal way by the notion of pulsation that allows a controlled oscillation between rigor and creativity. In a concrete way, the *CM*-formula construction allows such a controlled oscillation and has influences on all the *CMM*.

These four fundamental problems are stemming from our perception of Cartesian Intuitionism. They appear as ideas of directions to be developed and to be formalized. These tasks will continue in our future work.

These problems are not, however, the only topics we shall deal with. In near future we intend to describe how the principles behind *CM*-formula construction apply in the design of evolving systems in general and in the evolving recursive *CMM* in particular. We have tackled this problem in an informal way in our book [41].

The power of *CMM* was illustrated on a number of interesting problems such as n-queens [34], the quotient and the rest of two numbers [32], a problem in robotics [45] and more recently the construction of a definition of Ackermann's function with respect to the second variable [43]. This last illustration is important because it shows the capacity of *CMM* to find another form of defining axioms, the final version of which is not known beforehand.

VII. ADVANTAGES AND DRAWBACKS

A Newtonian paradigm has the enormous advantage of being fully accepted and respected in the scientific community. As far as Program Synthesis is concerned, it allows bringing quickly highly user-dependent implementations. Its main drawback is however that it provides no clear future orientations of the research on inductive theorem proving. This manifests by a long pause in Newtonian research starting in ninetieth and followed by resurgence around 2010 [66], [67], [63], [58]. These new approaches deviate from the original PS problem, which is that of a user-independent strategy for proving theorems, by introducing a library of efficient templates suitable for one kind of problems or by identifying interesting classes of algorithms and by capturing as much generic algorithm design knowledge as possible in one place. Their contribution is practically very useful in the short term perspective but, in the long term one, it represents the work on building libraries for semantic classes of programs and a need for big data handling. This is an economically useful orientation. However, from the point of view of scientific curiosity, it misses the (reasonable) ambition of Cartesian Paradigm.

In this paper, we have illustrated that Cartesian paradigm is suited for generating a sequence of missing sub-routines. That is not yet possible in simplifications approaches.

The advantage of Cartesian Paradigm lies in its long-term vision of evolving (though disruptive) theorem proving systems. However, this long-term and disruptive perspective is not easily accessible, and makes it somewhat unattractive for researchers seeking quick gratification.

In short, Cartesian paradigm is an advantageous paradigm since it has

- a solid epistemic justification (this somewhat smoothens up its disruptive character);

and it enables:

- accepting Gödel's results in proactive way;
- considering PS as a problem of a developing a technology rather than a procedure of decision;
- introducing the idea of creating complex evolving systems as a complement to the largely accepted idea of observing and manipulating such systems (e.g., by Machine Learning, Knowledge Discovery, Data Mining and so on);
- allows placing PS in the context of creating evolving, recursive and symbiotic systems;
- allows integrating human creativity directly into the systems to be conceived.

The main drawbacks of Cartesian paradigm are the following:

- consideration of PS problem as a problem of a disruptive technology is not yet widespread;
- lack of availability for formations teaching to think in terms of evolving, recursive and symbiotic systems;
- creation of such systems is slow and difficult to evaluate by external observers;
- people used to linear conception of systems are disturbed by necessity to conceive at first mentally all the 'informal chunks' (i.e., constructors) of such systems before the actual implementation starts;
- necessity of collaborations between PS and several non-deductive methods such as they exist in Machine Learning, Data Mining, Knowledge Discovery and other domains.

The difficulty of PS in general confirms that we cannot expect a rapid development of powerful general purpose oriented industrial systems. Nevertheless, both paradigms have an important place in contemporary research.

VIII. CONCLUSION

In this paper, we have formulated two fundamental questions, namely whether the logical limits of Gödel's results can be 'overcome' by a pragmatic reformulation of the PS problem and whether there can be a custom-designed theorem prover for PS. The paper justifies our positive answers to these questions by putting forward the foundations for Newtonian and Cartesian systemic paradigms and by indicating the necessity of their synergy.

In contrast to Newtonian theoretical metrics of evaluation of PS systems, the paper suggests the metrics of robustness and conceptual symbiotic expressed by the measure of Cartesian Intuition.

This paper presents Cartesian and Newtonian paradigms in PS to a larger extent than our publications [1] and [2], namely by

- mentioning the main orientation of recent works on PS in Newtonian paradigm;
- comparing this orientation with our Cartesian approach
- thorough describing the epistemological background for the Cartesian Intuitionism;
- illustrating
  - some consequences of adopting Cartesian Intuitionism as epistemological justification of the conception of a recursive system and
  - the difference between a Newtonian decision and Cartesian construction procedure;
- presenting an expansion of the experiment presented in [1];
- illustrating that Cartesian Intuitionism can be looked upon as a 'generator of new ideas' not only in the form of missing axioms and lemmas in theorem proving process but also in the form of notions proper to custom-made creation of evolving symbiotic systems.

So far, the Newtonian paradigm has been very successful in producing systems that request human help as soon as some non-trivial 'creativity' is needed in order to provide a lemma or a heuristic not already included in the system library. Since one of our ultimate goals is modeling some form of mathematical systemic creativity by building a computer simulation of these creative steps, we had to adopt a new perspective, the one of Cartesian Intuitionism.

Cartesian Paradigm is disruptive not only by its evolving, symbiotic and recursive character but also because it brings an unusual action-oriented perspective to interpreting Gödel's results.

The Cartesian Paradigm faces more obstacles than the Newtonian one because of its complexity and because neither a superficial external observation (due to the presence of the symbiotic thinking in Cartesian Intuition) nor the sequential transmission (due to the use of recursion) nor a rigid formal perception (due to its evolving character) are suited to the appreciation of the work made in this recursive way. One of our goals in this paper was a call-to-action for tearing down these artificial obstacles immanent within the realm of the Newtonian paradigm. One of our goals was also to stress out the complementary and highly non-competing character of both paradigms.

ACKNOWLEDGMENT

I would like to express my warmest thanks to Michèle Sebag, my research group director at L.R.I., and Yves Kodratoff who helped me to express the ideas presented in this paper. Thanks to Veronique Benzaken for her moral support. The feedback provided by Dieter Hutter and the comments of referees of this Journal as well as of the ones of COGNITIVE 2013 and ICONS 2014 contributed to improve the quality of this paper.

REFERENCES

[1] M. Franova, "A Cartesian methodology for an autonomous program synthesis system," in M.Jäntti, and G. Weckman, Eds., Proc. of ICONS 2014, The Ninth International Conference on Systems; ISBN: 978-1-61208-319-3, pp. 22-27, 2014.

[2] M. Franova, "Cartesian Intuitionism for program synthesis," in. S. Shimizu, and T. Bosomaier, Eds., Cognitive 2013, The Fifth International Conference on Advanced Cognitive Technologies and Applications; ISBN: 978-1-61208-273-8, pp. 102-107, 2013.

[3] A. Asperti, C. S. Coen, E. Tassi, and S. Zacchiroli, "User interaction with the Matita Proof Assistant," Journal of Automated Reasoning, August 2007, Volume 39, Issue 2, pp. 109-139, August 2007.

[4] G. Bachelard, The Formation of the Scientific Mind: A Contribution to a Psychoanalysis of Objective Knowledge. Clinamen Press Ltd., 2001.

[5] F. Bacon, Novum Organum. P.U.F, 1986.

[6] D. A. Basin and T. Walsh, "A calculus for and termination of rippling," JAR, Volume 16, Issue 1-2, pp. 147-180, March 1996.

[7] B. Beckert, R. Hähnle, and P. H. Schmitt, Eds., Verification of Object-Oriented Software. The KeY Approach. Lecture

Notes in Computer Science, Volume 4334, Springer-Verlag, 2007.

[8] M. Beeson, "Mathematical induction in Otter-Lambda," Journal of Automated Reasoning, Volume 36, Issue 4, pp. 311-344, April 2006.

[9] Y. Bertot and P. Casteran, Interactive Theorem Proving And Program Development - Coq'art: The Calculus Of Inductive Constructions. Springer-Verlag, 2004.

[10] W. Bibel, "On syntax-directed, semantic-supported program synthesis," Artificial Intelligence 14, pp. 243-261, 1980.

[11] S. Biundo and F. Zboray, "Automated induction proofs using methods of program synthesis," Computers and Artificial Intelligence, 3, No. 6, pp. 473-481, 1984.

[12] R. S. Boyer and J S. Moore, A Computational Logic Handbook. Academic Press, Inc., 1988.

[13] A. Bundy, "The automation of proof by mathematical induction," in A. Robinson and A. Voronkov, Eds., Handbook of Automated Reasoning, Volume I; North-Holland, pp. 845-912, 2001.

[14] A. Bundy, F. Van Harnelen, C. Horn, and A. Smaill, "The Oyster–Clam system," in Stickel, M.E. Eds. 10th International Conference on Automated Deduction, Lecture Notes in Artificial Intelligence, Volume 449, Springer, pp. 647–648, 1990.

[15] A. Bundy, D. Basin, D. Hutter, and A. Ireland, Rippling: Meta-Level Guidance for Mathematical Reasoning; Cambridge University Press, 2005.

[16] R. M. Burstall, "Proving properties of programs by structural induction," Computer J., Volume 12, Issue 1, pp. 41-48, 1969.

[17] R. Bodik and B. Jobstmann, "Algorithmic program synthesis: introduction," in International Journal on Software Tools for Technology Transfer, Volume 15, Issue 5-6, pp. 397-411, October 2013.

[18] J. Chazarain and S. Muller, "Automated synthesis of recursive programs from a "forall" "exists" logical specification," Journal of Automated Reasoning, Volume 21, Issue 2, pp. 233-275, October 1998.

[19] K. Claessen, M. Johansson, D. Rosén, and N. Smallbone, "Automating inductive proofs using theory exploration," Automated Deduction – CADE-24, Lecture Notes in Computer Science, Volume 7898, pp. 392-406, 2013.

[20] R. L. Constable, Implementing Mathematics with the Nuprl Proof Development System. Prentice-Hall, Inc., Englewood Clifs, New Jersey, 1986.

[21] T. Coquand and G. Huet, "Constructions: A higher order proof system for mechanizing mathematics," in B. Buchberger, Eds., EUROCAL'85, European Conference on Computer Algebra, Proceedings Volume 1: Invited Lectures; April, Springer-Verlag, Linz, Austria, pp. 151-185, 1985.

[22] A. Damasio, Descartes' Error: Emotion, Reason and the Human Brain. Vintage, 2006.

[23] E. de Bono, Serious Creativity: Using the Power of Lateral Thinking to Create New Ideas. HarperCollins, 1992.

[24] L. De Moura and N. Bjorner, "Satisfiability modulo theories: an appetizer," Formal Methods: Foundations and Applications; Lecture Notes in Computer Science, Volume 5902, pp. 23-36, 2009.

[25] N. Dershowitz and U.S. Reddy, "Deductive and inductive synthesis of equational programs," JSC Volume 15, Nos. 5 and 6, pp. 463-466.

[26] R. Descartes, Oeuvres philosophiques (3 vol.), Ed. F. Alquié, T. 1; Classiques Garnier, Bordas, 1988.

[27] R. Descartes, "Discours de la méthode pour bien conduire sa raison et chercher la vérité dans les sciences," in R. Descartes, Oeuvres philosophiques (3 vol.). Edition de F. Alquié, T. 1; Classiques Garnier, Bordas, pp. 567-650, 1988.

[28] R. Descartes, "Les principes de la philosophie," in R. Descartes, Oeuvres philosophiques (3 vol.). Edition de F. Alquié. T. 3; Classiques Garnier, Bordas, pp. 87-525, 1989.

[29] R. Descartes, "Regulae ad directionem ingenii," in R. Descartes, Oeuvres complètes (11 vol.). Edition Adam et Tannery. T. 10; Vrin, Paris, pp. 359-469, 1996.

[30] L. Dixon and J. Fleuriot, "IsaPlanner: a prototype proof planner in Isabelle," in F. Baader, Eds., CADE-19, LNAI 2741, pp. 279–283, 2003.

[31] J. Endrullis, H. Geuvers, J. G. Simonsen, and H. Zantema, "Levels of undecidability in rewriting," Information and Computation archive, Volume 209, Issue 2, pp. 227-245, February 2011.

[32] M. Franova, "Program synthesis and constructive proofs obtained by Beth's tableaux," in R. Trappl, Eds., Cybernetics and System Research 2; North-Holland, Amsterdam, pp. 715-720, 1984.

[33] M. Franova, "CM-strategy : A methodology for inductive theorem proving or constructive well-generalized proofs," in A. K. Joshi, Eds., Proceedings of the Ninth International Joint Conference on Artificial Intelligence; August, Los Angeles, pp. 1214-1220, 1985.

[34] M. Franova, "An implementation of program synthesis from formal specifications," in Y. Kodratoff, Eds., Proceedings of the 8th European Conference on Artificial Intelligence; August 1-5, Pitman, London, United Kingdom, pp. 559-564, 1988.

[35] M. Franova, Fundamentals of a new methodology for program synthesis from formal specifications: CM-construction of atomic formulae. Thesis, Université Paris-Sud, November, Orsay, France, 1988.

[36] M. Franova, "PRECOMAS 0.3 user guide," Rapport de Recherche No.524, L.R.I., Université de Paris-Sud, Orsay, France, October, 1989.

[37] M. Franova, "A constructive proof for prime factorization theorem: A result of putting it together in Constructive Matching methodology," Rapport de Recherche No.780, L.R.I., Université de Paris-Sud, Orsay, France, October, 1992.

[38] https://www.lri.fr/~mf/index.html 2014.11.11

[39] M. Franova, "PRECOMAS - An implementation of Constructive Matching Methodology," Proceedings of ISSAC'90, ACM, New York, 1990, 16-23.

[40] M. Franova, "Constructive Matching methodology and automatic plan-construction revisited," Rapport de Recherche No. 874, L.R.I., Univ. de Paris-Sud, Orsay, France, November, 1993.

[41] M. Franova, Formal Creativity: method and practice – conception of complex 'computerized' systems and epistemological patent (Créativité Formelle: méthode et pratique - conception des systèmes "informatiques" complexes et brevet épistémologique). Publibook, 2008.

[42] M. Franova, "A Construction of several definitions recursive over the variable under the exponent for the exponent function," Rapport de Recherche No.1519, L.R.I., Université de Paris-Sud, Orsay, France, June, 2009.

[43] M. Franova, "A construction of a definition recursive with respect to the second variable for the Ackermann's function," Rapport de Recherche No.1511, L.R.I., Université de Paris-Sud, Orsay, France, 2009.

[44] M. Franova, "The role of recursion in and for scientific creativity," in R. Trappl, Eds., Cybernetics and Systems 2010; Proc. of the Twentieth European Meeting on Cybernetics and System research, Austrian Society for Cybernetic Studies, pp. 573-578, 2010.

[45] M. Franova and Y. Kodratoff, "How to clear a block with Constructive Matching methodology," in J. Mylopoulos, R. Reiter, Eds., Proceedings of the Twelfth International Joint

Conference on Artificial Intelligence, IJCAI'91; Morgan Kaufmann, pp. 232-337, 1991.

[46] M. Franova and Y. Kodratoff, "On Computational Creativity: 'inventing' theorem proofs," Rauch J., Ras Z.W., Berka P., Eloma T. Eds., Foundations of Intelligent Systems, 18th International Symposium, ISMIS 2009, LNAI 5722, September, Springer, pp. 573-581, 2009.

[47] M. Franova and Y. Kodratoff, "Two examples of computational creativity: ILP multiple predicate synthesis and the 'assets' in theorem proving," in J. Koronacki, Z. W. Ras, S.T. Wierzchon and J. Kacprzyk, Eds., Advances in Machine Learning II: Dedicated to the Memory of Professor Ryszard S. Michalski; Springer-Verlag, pp. 155-174, 2010.

[48] M. Franova and M. Kooli, "Recursion manipulation for robotics: why and how?," in R. Trappl, Eds., Cybernetics and Systems '98; proc. of the Fourteenth Meeting on Cybernetics and Systems Research, Austrian Society for Cybernetic Studies, Vienna, Austria, pp. 836-841, 1998.

[49] M. Franova and L. Popelinsky, "Synthesis of formal specifications of predicates: why and how?," in R. Trappl, Eds., Cybernetics and Systems 2000; proc. of the Fifteenth European Meeting on Cybernetics and Systems Research, Volume II, Austrian Society for Cybernetics Studies, pp. 739-744, 2000.

[50] J. Y. Girard, Le Point Aveugle I - Cours de Logique - Vers la Perfection. Hermann, 2006.

[51] K. Gödel, "Some metamathematical results on completeness and consistency, On formally undecidable propositions of Principia Mathematica and related systems I, and On completeness and consistency," in J. van Heijenoort: From Frege to Godel, A source book in mathematical logic, 1879-1931; Harvard University Press, Cambridge, Massachusets, pp. 592-618, 1967.

[52] A. Ireland, "The use of planning critics in mechanizing inductive proofs," in Logic Programming and Automated Reasoning, Lecture Notes in Computer Science, Volume 624, pp. 178-189, 1992.

[53] M. Johansson, L. Dixon, and A. Bundy, "Case-Analysis for rippling and inductive proof," Interactive Theorem Proving, Lecture Notes in Computer Science, Volume 6172, pp. 291-306, 2010.

[54] D. Kapur, "An overview of Rewrite Rule Laboratory (RRL)," J. Comput. Math. Appl. 29(2), pp. 91–114, 1995.

[55] Y. Korukhova, "An approach to automatic deductive synthesis of functional programs," Annals of Mathematics and Artificial Intelligence, Volume 50, Issue 3-4, pp. 255-271, August 2007.

[56] J. L. Le Moigne, La théorie du système général, théorie de la modélisation. P.U.F, 1984.

[57] Z. Manna and R.Waldinger, "A deductive approach to program synthesis," ACM Transactions on Programming Languages and Systems, Volume 2, Issue 1, January, pp. 90-121, 1980.

[58] S. Nedunuri, D.R. Smith, and W.R. Cook, "Theory and techniques for synthesizing efficient breadth-first search algorithms," in FM 2012: Formal Methods, Lecture Notes in Computer Science, Volume 7436, pp. 308-325, 2012.

[59] C. Paulin-Mohring and B. Werner, "Synthesis of ML programs in the system Coq," Journal of Symbolic Computation; Volume 15, Issues 5–6, pp. 607–640, May–June 1993.

[60] L. C. Paulson, "The foundation of a generic theorem prover," Journal of Automated Reasoning, Volume 5, Issue 3, pp. 363-397, September 1989.

[61] B. Pientka and Ch. Kreitz, "Instantiation of existentially quantified variables in inductive specification proofs," Artificial Intelligence and Symbolic Computation, Lecture Notes in Computer Science, Volume 1476, pp. 247-258, 1998.

[62] K. Popper, The logic of scientific discovery. Harper, 1968.

[63] Y. Pu, R. Bodik, and S. Srivastava, "Synthesis of first-order dynamic programming algorithms," in OOPSLA '11: Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications, pp. 83-98, 2011.

[64] D. R. Smith, "Top-down synthesis of simple divide and conquer algorithm," Artificial Intelligence, Volume 27, Issue 1, pp. 43-96, 1985.

[65] W. Sonnex, S. Drossopulou, and S. Eisenbach, "Zeno: an automated prover for properties of recursive data structures," in Proceedings of TACAS, Springer, pp. 407-421, 2012.

[66] S. Srivastava, S. Gulwani, and J. S. Foster, "From program verification to program synthesis", in M. V. Hermenegildo and J. Palsberg, Eds., Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, pp. 313-326, 2010.

[67] M. Vechev, E. Yahav, and G. Yorsh, "Abstraction-guided synthesis of synchronization," POPL '10: Proceedings of the 37th annual ACM SIGPLAN-SIGACT Symposium on Principles of programming languages, pp. 327-338, 2010.

A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems

Parastoo Delgoshaei
Department of Civil and Environmental Engineering
University of Maryland
College Park, MD 20742, USA
Email: parastoo@umd.edu

Mark A. Austin
Department of Civil and Environmental Engineering
University of Maryland
College Park, MD 20742, USA
Email: austin@isr.umd.edu

Amanda J. Pertzborn
Energy and Environment Division
National Institute of Standards and Technology (NIST)
Gaithersburg, MD 20899, USA
Email: amanda.pertzborn@nist.gov

***Abstract*—This paper describes a new semantic framework for model-based systems engineering, requirements traceability, and system simulation and assessment of cyber-physical systems (CPSs). When fully developed this environment will support the organization and integration of hierarchies of physical and software components, and perform analysis on their discrete and continuous behavior. Results of computational analysis will work alongside domain ontologies for decision making and rule checking procedures. To support the modeling and simulation of physical system behavior, and integration of the physical and cyber domains, we introduce Whistle, a new scripting language where physical units are embedded within the basic data types, matrices, and method interfaces to external object-oriented software packages. The capabilities of Whistle are demonstrated through a series of progressively complicated applications.**

Keywords-Cyber-Physical System; Semantic Modeling; Simulation Environment; Software Design Pattern; Rule Checking.

I. Introduction

Problem Statement. This paper is concerned with the development of procedures and software for the model-based systems engineering, integration, simulation and performance-assessment of cyber-physical systems (CPS). It builds upon our previous work [1] on semantic platforms for requirements traceability and system assessment. As illustrated in Figure 1, the distinguishing feature of CPS is a coupling of physical and cyber systems, with the cyber affecting the physical and vice versa. In a typical CPS application, embedded computers and networks will monitor and control physical processes, usually with feedback. The basic design requirement is that software and communications technologies will work together to deliver functionality that is correct and works with no errors. Unfortunately, present-day design procedures are inadequate for the design of modern CPS systems. A key problem is that today we do not have a mature science to support systems engineering of high-confidence cyber-physical systems assembled from subsystems having multiple physics (e.g., chemical, mechanical, electrical) [2], [3]. Design space exploration and trade studies are also difficult to conduct because decision variables span parametric, logical, and dependency relationship types. Components are often required to serve multiple functions – as such, cause-and-effect mechanisms are no longer localized and obvious. System relationships can reach laterally across systems hierarchies and/or intertwined network structures.

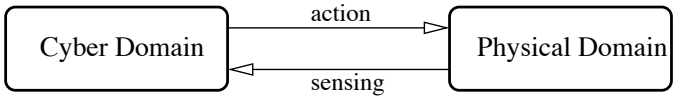


Figure 1. Interaction of cyber and physical domains in CPS.

In order for cyber-physical design procedures to proceed in a rational way we need mechanisms to easily combine abstractions from multiple physics and field equations (e.g., solids, fluids, heat, electromagnetics, chemistry) into sets of coupled equations that model the system. Components may be discrete (e.g., rigid body elements, control actuation elements, software logic), or continuous (e.g., differential equations for fluid flow). The challenge in developing accurate models of CPS behavior is complicated by differences in the underlying operation and data-stream flows associated with cyber and physical components. Whereas physical systems tend to have behavior that is continuous and associated with flows having physical quantities, cyber operates on discrete logic. To address these limitations, new computer programs and languages are required to address the challenges of distributed, complex CPSs. Their capabilities need to include establishing feedback loops between physical processes and computational units involving robust analysis, decision making mechanisms, dynamic modeling, knowledge of sensors and actuators, and computer networks. In a step toward creating this long-term goal, we are working on the development of a computational infrastructure where domain specific ontologies and rule checking routines operate hand-in-hand with a new scripting language introduced here as Whistle. This new language employs object-oriented design principles and software design patterns as a pathway to addressing challenging design questions.

Model-based Systems Engineering. Model-based systems engineering (MBSE) development is an approach to systems-level development in which the focus and primary artifacts of development are models, as opposed to documents. Our research methodology is driven by a need to achieve high levels of productivity in system development. We believe that high levels of productivity in system development can be achieved through the use of high-level visual abstractions coupled with lower-level (mathematical) abstractions suitable for formal systems analysis. The high-level abstractions provide

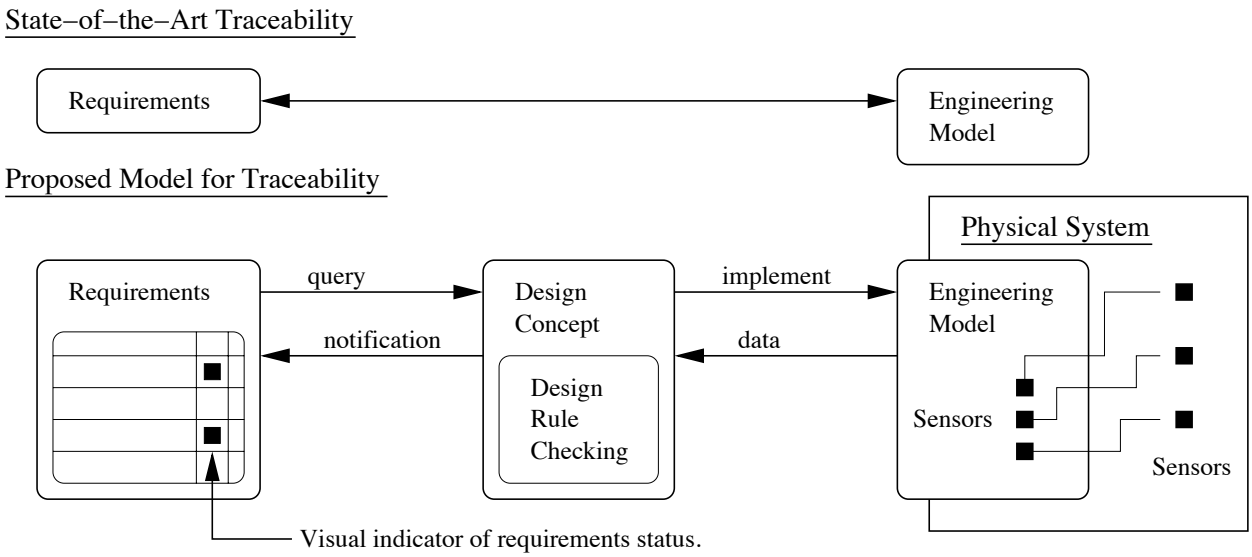


Figure 2. Schematics for: (top) state-of-the-art traceability, and (bottom) proposed model for ontology-enabled traceability for systems design and management.

a "big picture" summary of the system under development and highlight the major components, their connectivity, and performance. The lower-level abstractions are suitable for formal systems analysis – for example, verification of component interface compatibilities and/or assessment of system performance through the use of simulation methods. The former one is achieved through semantic web technologies, i.e., with domain specific ontologies. On the other hand, detailed simulation analysis can be performed by scripting language, or other analysis packages that are compatible with scripting language.

A tenet of our work is that methodologies for strategic approaches to design will employ semantic descriptions of application domains, and use ontologies and rule-based reasoning to enable validation of requirements, automated synthesis of potentially good design solutions, and communication (or mappings) among multiple disciplines [4][5][6]. A key element of required capability is an ability to identify and manage requirements during the early phases of the system design process, where errors are cheapest and easiest to correct. The systems architecture for state-of-the-art requirements traceability and the proposed platform model is shown in the upper and lower sections of Figure 2. In state-of-the-art traceability mechanisms, design requirements are connected directly to design solutions (i.e., objects in the engineering model). Our contention is that an alternative and potentially better approach is to satisfy a requirement by asking the basic question: What design concept (or group of design concepts) should I apply to satisfy a requirement? Design solutions are the instantiation/implementation of these concepts. The proposed architecture is a platform because it contains collections of domain-specific ontologies and design rules that will be reusable across applications. In the lower half of Figure 2, the textual requirements, ontology, and engineering models provide distinct views of a design: (1) Requirements are a statement of "what is required." (2) Engineering models are a statement of "how the required functionality and performance might be achieved," and (3) Ontologies are a statement of "concepts justifying a tentative design solution." During design, mathematical and logical rules are derived from textual requirements which, in turn, are connected to elements in an engineering model. Evaluation of requirements can include checks for satisfaction of system functionality and performance, as well as identification of conflicts in requirements themselves. A key benefit of our approach is that design rule checking can be applied at the earliest stage possible – as long as sufficient data is available for the evaluation of rules, rule checking can commence; the textual requirements and engineering models need not be complete. During the system operation, key questions to be answered are: What other concepts are involved when a change occurs in the sensing model? What requirement(s) might be violated when those concepts are involved in the change? To understand the inevitable conflicts and opportunities to conduct trade space studies, it is important to be able to trace back and understand cause-and-effect relationships between changes at system-component level and their affect on stakeholder requirements. Present-day systems engineering methodologies and tools, including those associated with SysML [7] are not designed to handle projects in this way.

Scope and Objectives. This paper describes a new approach to requirements traceability, simulation, and system assessment through the use of semantic platforms coupled with a component-based language where physical quantities (not just numbers) are deeply embedded in the language design and execution. The rationale for providing cyber with this capability is simple: if the cyber has an enhanced ability to represent the physical world in which it is embedded, then it will be in a better position to make decisions that are appropriate and correct.

Our test-bed application area and driver for this research is performance-based modeling and design of energy-efficient building environments. Modern buildings contain a variety of intertwined networks for the hierarchical arrangement of spaces (e.g., buildings have floors, floors contain rooms, and so forth), for fixed circulatory systems, e.g., power and heating, ventilation, and air conditioning (HVAC), for dynamic circulatory systems, e.g., air and water flows, and for wired and wire-

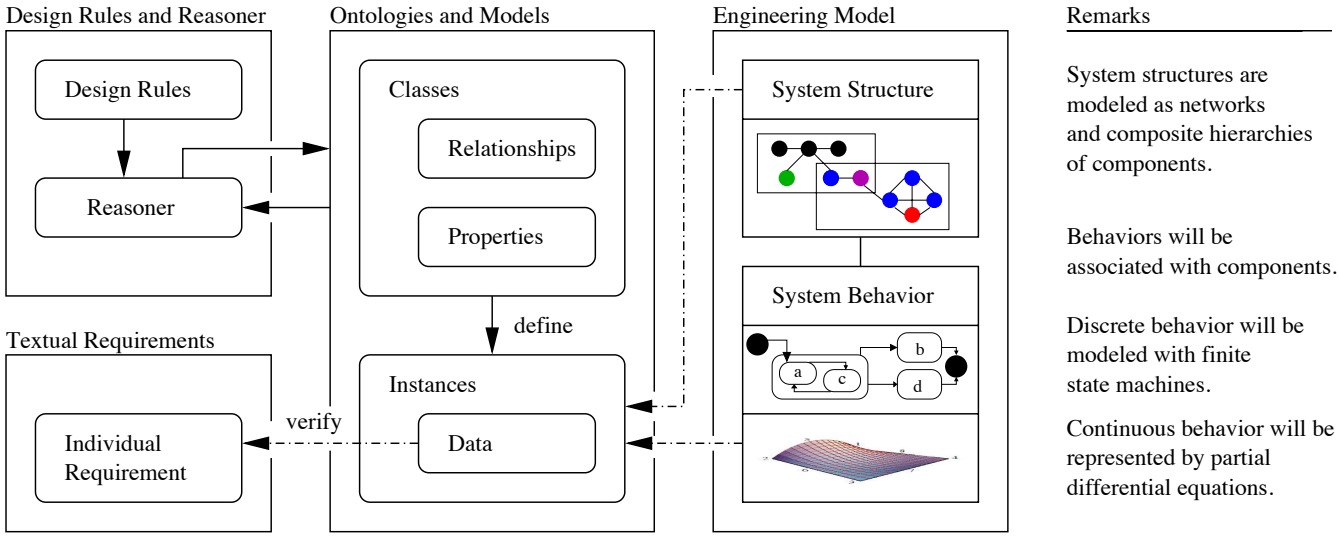


Figure 3. Framework for implementation of ontology-enabled traceability and design assessment.

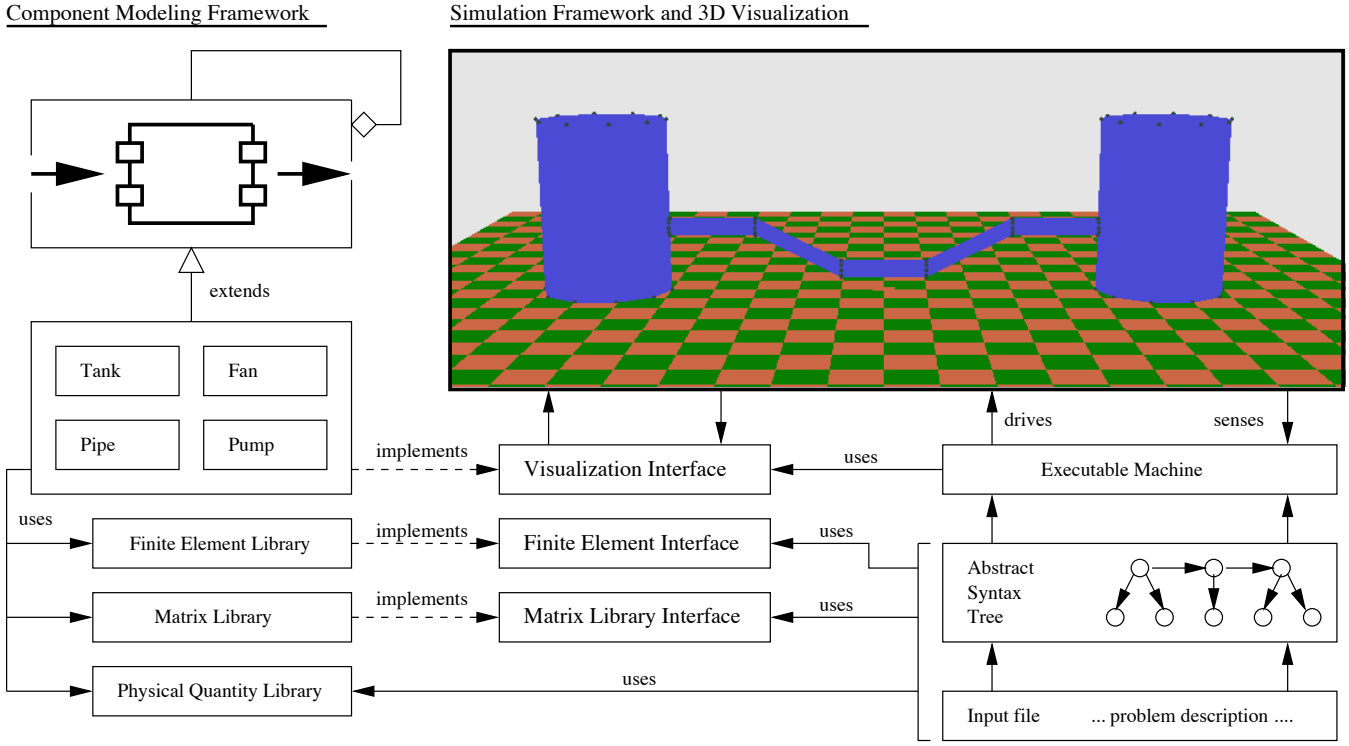


Figure 4. Architecture for modeling HVAC systems as networks of connected components, and using finite element solution procedures for computing and visualizing time-history behavior.

less communications. While there is a desire for each network to operate as independently as possible, in practice the need for new forms of functionality will drive components from different network types to connect in a variety of ways. Within the building simulation community state-of-the-art dynamic simulation is defined by Modelica, and steady-state simulation by DOE-2 and eQuest. From a CPS perspective, the time-history analysis and control of building system performance is complicated by the need to model combinations of discrete (e.g., control) and continuous behaviors (e.g., the physics of fluid dynamics). Predictions of dynamic behavior correspond to the solution of nonlinear differential algebraic equations (e.g., for water, air, and thermal flow) coupled to discrete equations (e.g., resulting from cyber decisions).

To facilitate and support this vision, we are currently working toward the platform infrastructure proposed by Figures 3 and 4. Figure 3 pulls together the different pieces of the proposed architecture shown in Figure 2. On the left-hand side the textual requirements are defined in terms of mathematical and logical rule expressions for design rule checking. Figure 4 highlights the software infrastructure for modeling systems that are part cyber and part physical. To deal with the complexity of

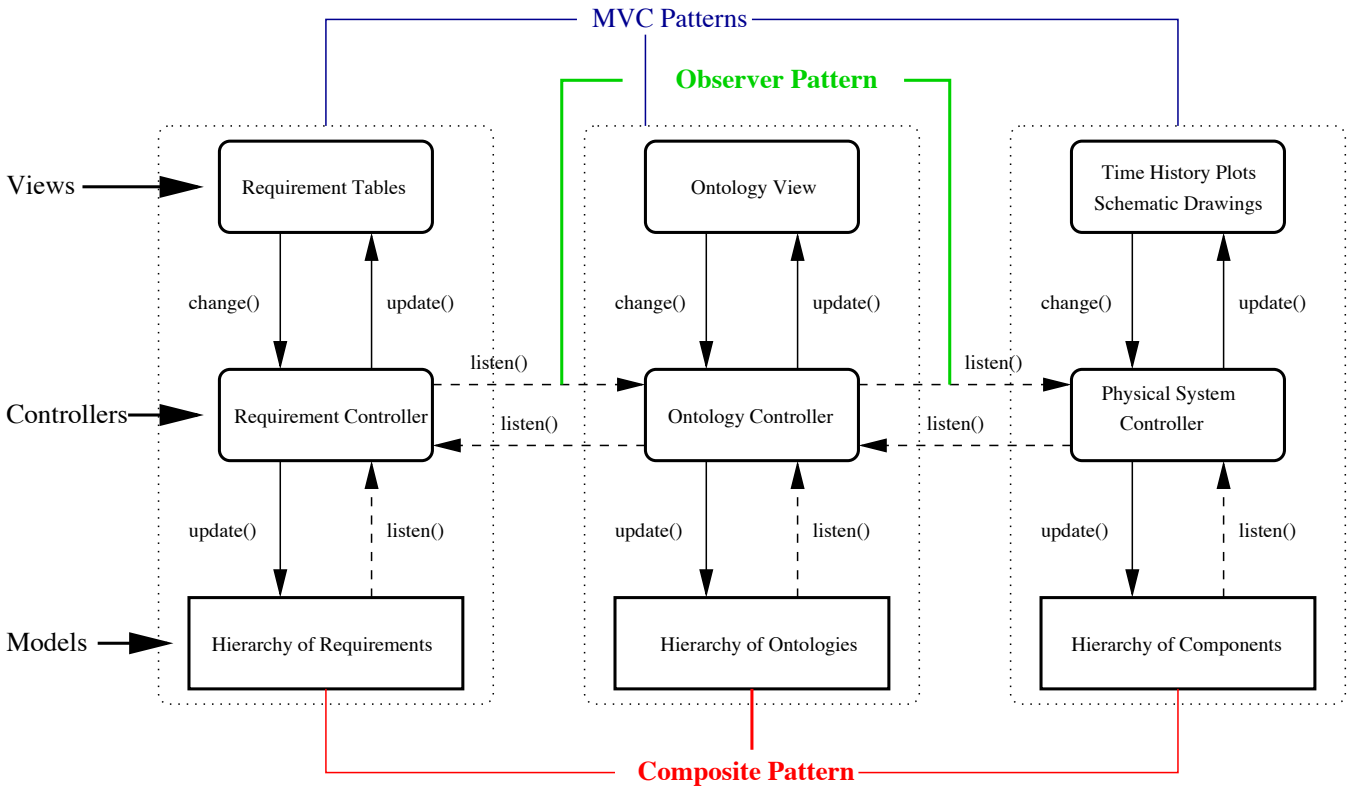


Figure 5. Software architecture for ontology-enabled traceability, annotated with model-view-controller, observer and composite-hierarchy design patterns.

building systems, which are defined by large numbers of physical and abstract components, we are proposing that models be organized into composite hierarchies, as shown on the top left-hand side of Figure 4. Specific component types will simply implement the composite hierarchy interface. To accommodate mixtures of discrete and continuous behavior, we are proposing that the software architecture implement a series of generic interfaces to software libraries for matrix computations, finite element analysis, and two- and three-dimensional visualization. This element is shown along the bottom of Figure 4. Finally, we need a capability for components to communicate across hierarchies, and we are proposing this be accomplished with listener mechanisms (e.g., a controller component might listen for data from a collection of sensor components). This is a work in progress. Looking ahead, our plans are to build a series of progressively capable software prototypes, with each iteration of development employing a combination of executable statecharts for the behavior modeling of HVAC components, and eventually finite element procedures for the computation of behaviors over continuous physical domains (e.g., fluid flow in a pipe network) [8][9][10].

This paper begins with a description of the semantic platform infrastructure and our use of software design patterns [11] (e.g., networks of model, view, controllers), software libraries and languages for semantic applications development using OWL [12] and Jena [13]. Section III describes related work. Section IV describes the design and features of Whistle, a scripting language we are developing to support the implementation of abstractions shown in Figures 3 and 4. A series of progressively complicated case study problems is presented in Section V.

II. Semantic Platform Infrastructure

Software Systems Architecture. Figure 5 represents the software architecture for ontology-enabled traceability and physical systems simulation, annotated with our use of model-view-controller, observer, and composite hierarchy software design patterns. Software design patterns are defined as general repeatable solutions to common software design problems; designers customize these templates to suit the design requirements. The model-view-controller (MVC) pattern is an architectural pattern with three components of model, view, and controller. This pattern is widely used in graphical user interface (GUI) applications. The observer design pattern defines a one-to-many relationship between objects. An observer component registers itself to a subject of interest and will be notified when an event occurs. An observer can register to different observable components or be removed when the interest no longer exists. The composite design pattern is used to describe groups of objects whose natural organizational structure is a hierarchy (e.g., a building contains floors; floors contain rooms; rooms contain desks and chairs). For composite hierarchies that represent spatial systems, algorithms can be developed to systematically traverse the hierarchy and process it according to a pre-defined purpose (e.g., display the contents of a hierarchy of coordinate systems; query to see if a point is inside a particular object). Another key benefit is model flexibility. Suppose, for example, that an engineer is working with a simple model of a building consisting of an air-handling unit and rooms defined by walls and doors and windows inside walls. If the room model is adjusted to a different orientation, then all of the subsystem elements (i.e., the walls, doors and windows) will be automatically re-positioned.

We employ a combination of MVC, observer, and composite hierarchy design patterns to synthesize dependency and data flow relationships between the requirements, ontology and engineering model work spaces, and a modified version of MVC where the controller serves as a mediator between multiple models and views. The latter can also be found in Apple Cocoa [14]. The requirements, ontology, and the physical system models are each represented as MVC nodes. Inside a MVC node, messages are distributed between the controller, views and models. Then, the observer design pattern is used to connect controller elements at each MVC node to other points of interest, thereby enabling traceability and flows of data across the system architecture. The controller registers with the model to be notified of a change in a property, and then updates views following a change in a model property. In practical terms, an end-user interacts with the views and makes changes to the model by passing data through the controller. Views pass the change queries to the controller and the controller updates the relevant models.

The composite hierarchy design pattern is used to organize the entities within each workspace. For the requirements model, this implies definition of compound requirements containing other sub-requirements. For the ontology models this implies that far-reaching ontology might be assembled from collections of ontologies describing specific domains. For example, an ontology for building systems might contain a mechanical systems ontology, among others. Finally, physical system models are created as hierarchies of components. Notice that the ontology controller is listening to the physical system controller and vice versa. This mechanism means that as a system is operating or is being simulated, changes in the system state will be reported to the ontology controller and will be updated in the data stored (individuals) in the ontology model. Looking the other way, an update to the value of a component attribute in the physical system model will trigger rule checking in the ontology workspace and possibly a change in the satisfaction of system requirements. For both scenarios, views will be updated upon a change in their models. The requirement controller listens to the ontology controller. This connection is the traceability thread back to the requirements. The requirements view will highlight the relevant requirement when the associated rule in the ontology is triggered.

Modeling and Reasoning with Ontologies. Textual requirements are connected to the ontology model and logical and mathematical design rules, and from there to the engineering model. Ontology models encompass the design concepts (ontology classes) in a domain, as well as the relationships among them. Classes are qualified with properties (c.f., attributes in classes) to represent the consequence of constraint and design rule evaluations. Examples of valid relationships are: containment, composition, uses, and ”Is Kind of”. These classes are place holders for the data extracted from the engineering model. Individuals are the object counterpart of classes, with data and object property relationships leading to the resource description framework -(RDF) graph infrastructure. Each instance of an individual holds a specific set of values obtained from the engineering model.

Rules serve the purpose of constraining the system operation and/or system design. They provide the mechanisms for early design verification, and ensure the intended behavior is achieved at all times during system operation. We are currently working with reasoners provided in the Jena API. A reasoner works with the RDF graph infrastructure and sets of user-defined rules to evaluate and further refine the RDF graph. Rule engines are triggered in response to any changes to the ontological model. This process assures that the model is consistent with respect to the existing rules. Traceability from ontologies to requirements is captured via implementation of the listeners that are notified as a result of change in the semantic model.

In a departure from past work, we are exploring the feasibility of creating built-in functions to capture and evaluate performance criteria, i.e., energy efficiency of the HVAC system. A second potential use of built-in functions is as an interface to packages that provide system improvements through optimization and performance related queries. We note that a rule-based approach to problem solving is particularly beneficial when the application logic is dynamic (i.e., where a change in a policy needs to be immediately reflected throughout the application) and rules are imposed on the system by external entities [15][16]. Both of these conditions apply to the design and management of engineering systems.

III. Related Work

An important facet of our work is use of Semantic Web technologies [17] as both system models and mechanisms to derive system behavior. While the vast majority of Semantic Web literature has used ontologies to define system structure alone, this is slowly changing. Derler and co-workers [18] explain, for example, how ontologies along with hybrid system modeling and simulation and concurrent models of computation can help us better address the challenges of modeling cyber-physical systems (CPSs). These challenges emerge from the inherited heterogeneity, concurrency, and sensitivity to timing of such systems. Domain specific ontologies are used to strengthen modularity, and to combine the model of system functionality with system architecture. As a case in point, the Building Service Performance Project proposes use of ontologies and rules sets to enhance modularity and perform cross-domain information exchange and representation [19]. Koelle and Strijland are investigating the design and implementation of a software tool to support semantic-driven architecture with application of rules for security assurance of large systems in air navigation [20].

For the cyber side of the CPS problem, visual modeling languages such as the Unified Modeling Language (UML) and SysML provide weak semantic support for MBSE. This leads us to consider languages and tools for MBSE that have stronger semantics. Consider, for example, the possibility of conceptual modeling through the use of ontologies and constraints represented as rules. In the physical domain, some modeling languages and modeling frameworks are developed to address the physical modeling and analysis of complex physical systems. Two well known examples are Modelica [21] and Ptolemy II [22]. Modelica offers strong physical modeling capabilities and features to be utilized in component based modeling. Physical equations are embedded inside components and components are connected together via ports. Some frameworks such as Open Modelica have been developed

to support graphical block diagram modeling with Modelica. Ptolemy studies modeling and simulation of concurrent real-time systems with actor-based designs. Actors are software components that communicate via message sending. A model is a network of interconnected actors. Moreover, directors implement a model of computation in this framework and can be attached to different layers of the model. For example, discrete-events (DE), data-flow (SDF), and 3-D visualization are some of the directions supported in Ptolemy [23]. The challenges for CPS design are greater because we need both the cyber and physical models to interact with each other, and at this time the bi-directional link connecting physical (continuous) operations to computational (discrete) operations is missing. Ongoing work is trying not only to cover this gap, but also take a step toward tieing the governing rules in the domain-specific ontologies to the textual requirements [24]. The work by Simko [25] uses CyPhyML, Hybrid Bond Graphs and ESMoL to formally describe the structure and behavior of CPSs. However, deductive reasoning is lacking in this work.

IV. Whistle Scripting Language

This section introduces Whistle, a new scripting language where physical units are deeply embedded within the basic data types, matrices, branching and looping constructs, and method interfaces to external object-oriented software packages. Whistle builds upon ideas prototyped in Aladdin [26][27][28] a scripting environment for the matrix and finite element analysis of engineering systems.

Language Design and Implementation. Scripting languages [29][30][31] are designed for rapid, high-level solutions to software problems, ease of use, and flexibility in gluing application components together. They facilitate this process by being weakly typed and interpreted at run time. Weakly typed means that few restrictions are placed on how information can be used a priori – the meaning and correctness of information is largely determined by the program at run time. And since much of the code needed to solve a problem using a system programming language is due to the language being typed, broadly speaking, weakly typed scripting languages require less code to accomplish a task [32]. Whistle is tiny in the sense that it uses only a small number of data types (e.g., physical quantities, matrices of physical quantities, booleans and strings). Features of the language that facilitate the specification of problem solutions include: (1) liberal use of comment statements (as with C and Java, c-style and in-line comment statements are supported), (2) consistent use of function names and function arguments, (3) use of physical units in the problem description, and (4) consistent use of variables, matrices, and looping and branching structures to control the flow of program logic.

Whistle is implemented entirely in Java. We use the tools JFlex (the Fast Scanner Generator for Java) [33] and BYACC/J (an extension of Berkeley YACC for Java) [34] to handle the parsing and lexical analysis of tokens and statements, Java Collections for the symbol table, and a variety of tree structure representations of the abstract syntax tree. A good introduction to symbol tables and abstract syntax tree representations can be found in the compilers and interpreters text by Mak [35].

Definition and Management of Physical Quantities. A physical quantity is a measure of some quantifiable aspect of the modeled world. In Whistle, basic engineering quantities such as length, mass, and force, are defined by a numerical value (number itself) plus physical units. Figure 6 is a subset of units presented in the Unit Conversion Guide [36], and shows the primary base units, supplementary units, and derived units that occur in engineering mechanics and structural analysis. The four basic units needed for engineering analysis are: length unit L; mass unit M; time unit t; and temperature unit T. Planar angles are represented by the supplementary base unit rad. Derived units are expressed algebraically in terms of base and supplementary units by means of multiplication and division, namely:

$$\text{units} = k \cdot L^{\alpha} M^{\beta} t^{\gamma} T^{\delta} \cdot \text{rad}^{\epsilon} \quad (1)$$

where $\alpha, \beta, \gamma, \delta$ and ϵ are exponents, and k is the scale factor. Numbers are simply non-dimensional quantities represented by the family of zero exponents $[\alpha, \beta, \gamma, \delta, \epsilon] = [0, 0, 0, 0, 0]$. The four basic units play the primary role in determining dimensional consistency of units in physical quantity and matrix operations. Because a radian represents the ratio of two distances (i.e., distance around the perimeter of a circle divided by its radius), most software implementations deal with radians as if they were dimensionless entities. Whistle departs from this trend by explicitly representing radians, and employing a special set of rules for their manipulation during physical quantity and matrix operations.

The scripting language libraries provide facilities for dynamic allocation of units (in both the US and SI systems), units copying, consistency checking and simplification, and units printing. Operations for units conversion are provided. In an effort to keep the scripting language usage and implementation as simple as possible, all physical quantities are stored as floating point numbers with double precision accuracy, plus units. Floating point numbers are viewed as physical quantities without units. There are no integer data types in Whistle.

Physical Quantity Arithmetic. Whistle supports the construction and evaluation of physical quantity expressions involving arithmetic, relational, and logical operators. The integration of units into the scripting language provides a powerful check for the dimensional consistency of formulas. A detailed summary may be found in Tables I and II. Suppose, for example, that we want to compute the force needed to move 1 kg over a distance of 10 m in 2 seconds. The fragment of code:

```
mass     = 1 kg;
distance = 10 m;
dt       = 2 sec;

force01 = mass*distance/dt^2;
print "*** Required force = ", force01;
```

demonstrates the procedure for defining the physical quantity variables mass (kg), distance (m) and dt (sec), and computing the required force. The output is:

```
*** Required force = [ 2.500, N]
```

Whistle provides a small library of built-in constants (e.g., Pi) and functions (e.g., Max(), Min(), Sqrt()) for the evaluation

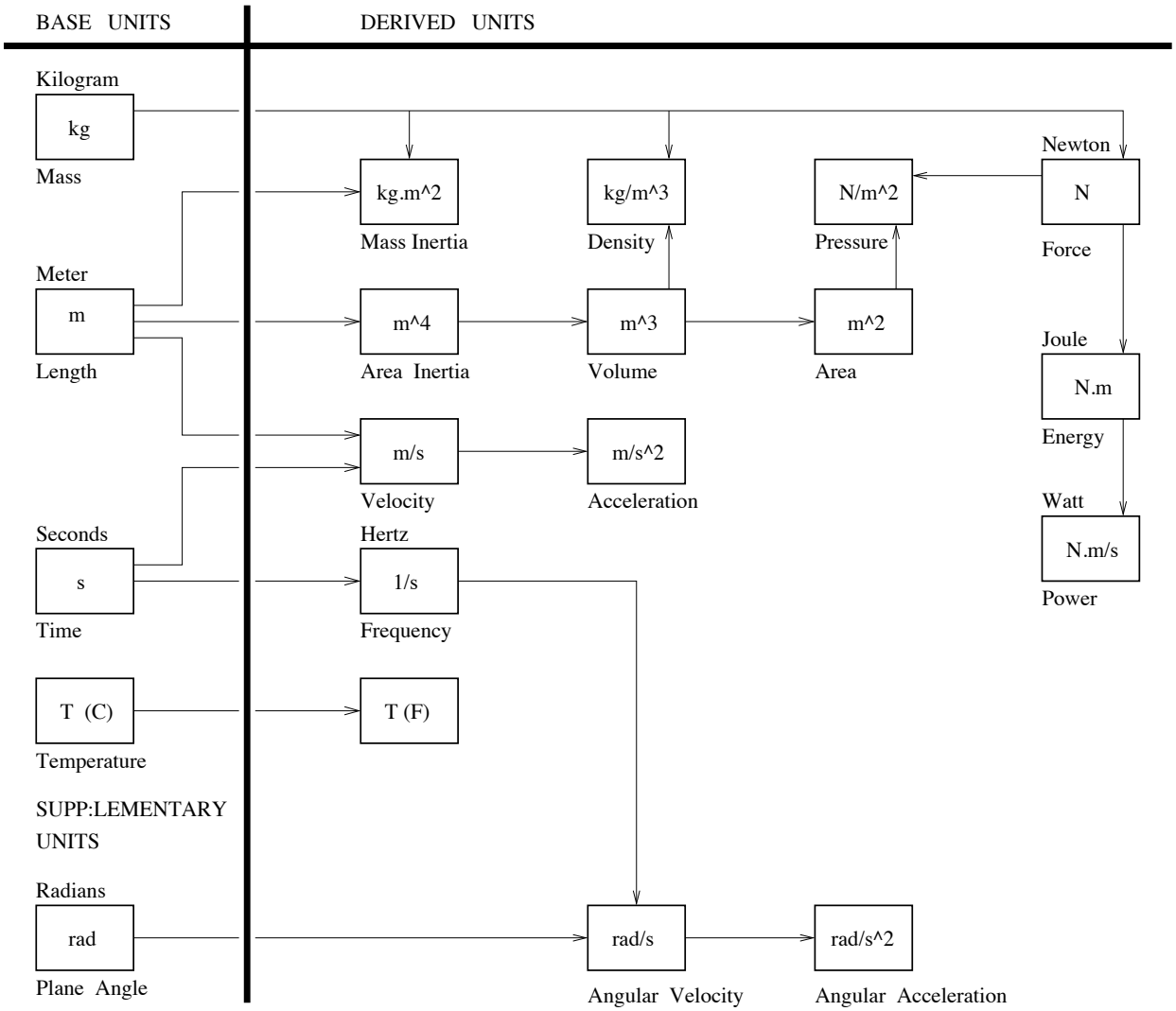


Figure 6. Primary base and derived units commonly found in engineering mechanics.

TABLE I. UNITS ARITHMETIC IN ARITHMETIC OPERATIONS

Description	Expression	Scale Factor	Unit Exponents
Addition	$q_1 + q_2$	k_1	$[\alpha_1, \beta_1, \gamma_1, \delta_1, \epsilon_1]$
Subtraction	$q_1 - q_2$	k_1	$[\alpha_1, \beta_1, \gamma_1, \delta_1, \epsilon_1]$
Multiplication	$q_1 * q_2$	$k_1 \cdot k_2$	$[\alpha_1 + \alpha_2, \beta_1 + \beta_2, \gamma_1 + \gamma_2, \delta_1 + \delta_2, \epsilon_1 + \epsilon_2]$
Division	q_1/q_2	k_1/k_2	$[\alpha_1 - \alpha_2, \beta_1 - \beta_2, \gamma_1 - \gamma_2, \delta_1 - \delta_2, \epsilon_1 - \epsilon_2]$
Exponential	$q_1 \hat{} q_2$	$k_1^{N\dagger}$	$[N\alpha_1, N\beta_1, N\gamma_1, N\delta_1, N\epsilon_1]^{\dagger}$

TABLE II. EXPRESSIONS INVOLVING RELATIONAL AND LOGICAL OPERATORS. A UNITS CONSISTENCY CHECK IS MADE BEFORE THE OPERATION PROCEEDS, AND THE RESULT OF THE OPERATION IS EITHER TRUE (1) OR FALSE (0). HERE WE ASSUME `x = 2 in` AND `y = 2 ft`.

Operator	Description	Example	Result
$<$	less than	$x < y$	true
$>$	greater than	$x > y$	false
$<=$	less than or equal to	$x <= y$	true
$>=$	greater than or equal to	$x >= y$	false
$==$	identically equal to	$x == y$	false
$!=$	not equal to	$x != y$	true
&&	logical and	$(x < y)$ && $(x <= y)$	true
$\|\|$	logical or	$(y < x) \ \| \ (x <= y)$	true
!	logical not	$!y$	false

of arithmetic expressions involving physical quantities. For example, the expressions:

```
print "Compute: Abs ( -2 cm ) --> ",
               Abs ( -2 cm );
print "Compute: Min ( 2 cm,   3 cm ) --> ",
               Min ( 2 cm,  3 cm );
print "Compute: Max ( 2 cm,   3 cm ) --> ",
               Max ( 2 cm,  3 cm );
```

generate the output:

```
Compute: Abs ( -2 cm ) --> [ 0.02000, m]
Compute: Min ( 2 cm,   3 cm ) --> [ 2.000, cm]
Compute: Max ( 2 cm,   3 cm ) --> [ 3.000, cm]
```

Relational and Logical Expressions. Whistle provides support for the representation and evaluation of relational expressions involving the "and operator" (&&), the "or operator" (||), and physical quantities. Consider, for example, the pair of lengths:

```
x = 10 cm; y = 20 cm;
```

The ensemble of expressions:

```
print "z01 = x <= 15 cm && y > x --> ",
             x <= 15 cm && y > x;
print "z02 = x <= 15 cm && y < x --> ",
             x <= 15 cm && y < x;
print "z03 = x <= 15 cm || y > x --> ",
             x <= 15 cm || y < x;
```

generates the output:

```
z01 = x <= 15 cm && y > x --> true
z02 = x <= 15 cm && y < x --> false
z03 = x <= 15 cm ||y > x --> true
```

Program Control. Program control is the basic mechanism in programming languages for using the outcome of logical and relational expressions to guide the pathway of a program execution. Whistle supports the "if" and "if-else" branching constructs, and the "while" and "for" looping constructs, with logical and relational operations being computed on physical quantities. The fragment of code:

```
x = 0 cm;
while ( x <= 10 cm ) {
   print "*** x = ", x;
   if ( x <= 5 cm ) {
      x = x + 1 cm;
   } else {
      x = x + 2 cm;
   }
}
```

generates the output:

```
*** x = [ 0.000, cm]
*** x = [ 1.000, cm]
*** x = [ 2.000, cm]
*** x = [ 3.000, cm]
*** x = [ 4.000, cm]
*** x = [ 5.000, cm]
*** x = [ 6.000, cm]
*** x = [ 8.000, cm]
*** x = [ 10.00, cm]
```

and demonstrates the basic functionality of a while loop and if-else branching construct working together.

Matrix Data Structure. Figure 7 shows the high-level layout of memory for the matrix data structure.

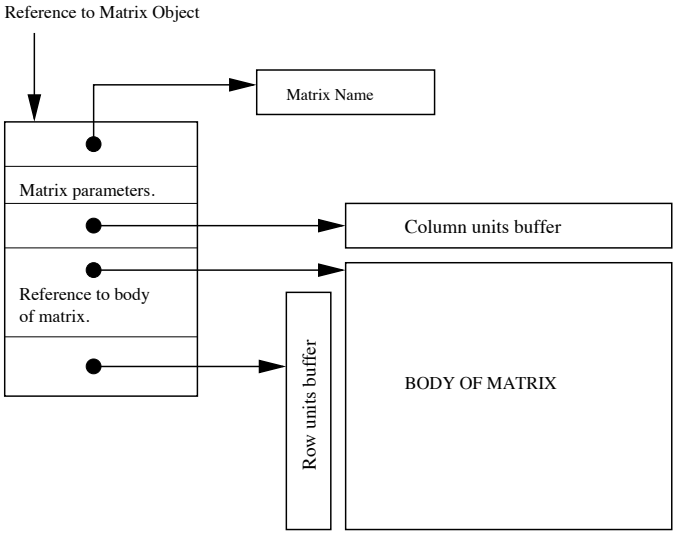


Figure 7. Layout of memory in matrix data structure.

Memory is provided for a character string containing the matrix name, two integers for the number of matrix rows and columns, as well as the matrix body. The matrix element units are stored in two one-dimensional arrays of type `Dimension`. One array stores the column units, and a second array the row units. The units for matrix element at row `i` and column `j` are simply the product of the `i-th` element of the row units buffer and the `j-th` element of column units buffer. Our use of row and column units matrices means that this model does not support the representation of matrices of quantities having arbitrary units. For most engineering applications, however, matrices are simply a compact and efficient way of describing families of equations of motion and equilibrium, and collections of data.

Engineering considerations dictate that the terms within an equation be dimensionally consistent. Similarly, consistency of dimensions in large collections of engineering data also must hold. In practical terms, the assumptions made by this model not only have minimal impact on our ability to solve engineering problems with matrices, but requires much less memory than individual storage of units for all matrix elements. Whistle performs dimensional consistency checks (and possible conversion of units types) before proceeding with all matrix operations. All that is required is examination of the row and column matrix units – there is no need to examine consistency of units at the matrix element level.

Matrix Operations. We are building computational support for standard matrix operations (e.g., addition, subtraction, multiplication, solution of linear equations) on physical quantities. For example, the fragment of code:

```
Force    = [ 2 N, 3 N, 4 N ];
Distance = [ 1 m; 2 m; 3 m ];
Work     = Force*Distance;
```

is a simple calculation for the work done by a force moving through a prescribed distance. The output is as follows:

```
Matrix: Force
row/col                  1            2            3
        units            N            N            N
   1           2.00000e+00  3.00000e+00  4.00000e+00

Matrix: Distance
row/col                  1
        units            m
   1           1.00000e+00
   2           2.00000e+00
   3           3.00000e+00

Matrix: Work
row/col                  1
        units          Jou
   1           2.00000e+01
```

Notice that the computation of units for the work done is automatically handled.

Java Bytecode Components. Early versions of the scripting environment [27] were essentially closed and came with a small set of built-in functions (e.g., Max(x,y), Abs (x), Sqrt (x)). Now, users can import references to compiled Java classes accessible in the JVM (Java Virtual Machine), and under certain restrictions, the methods of those classes can become part of the scripting environment. As we will soon see in the case study examples in section V, scripting statements of the form:

```
import className;
```

will dynamically load `className` into the scripting environment at runtime. When a class is loaded, all of the classes it references are loaded too. This class loading pattern happens recursively, until all classes needed are loaded.

This capability means that end-users can use the scripting language to glue computation components together and export heavy-duty computations to external mechanisms, such as Java libraries, or any other libraries to which Java can interface. Because our work has been driven by the simulation needs of energy efficient buildings, we initially had in mind that these classes would represent physical components in the building. However, from a scripting language perspective, whether or not the component represents a physical entity is irrelevant. As such, and as we will see in the case study examples below, components can also be defined for plotting, data modeling, executable statechart behaviors or, in fact, any modeling abstraction that uses physical quantity interfaces.

V. Case Study Problems

We now demonstrate the capabilities of Whistle by working step by step through five progressively complicated case study problems.

Case Study 1: Parsing a Simple Assignment Statement. The computational platform parses problem specifications into an abstract syntax tree, and then executes the statements by traversing the syntax tree in a well-defined manner. To see how this process works in practice, let's begin by working step by step through the details of processing the assignment statement:

```
x = 2 in;
```

Figure 8 shows the parse tree for this statement.

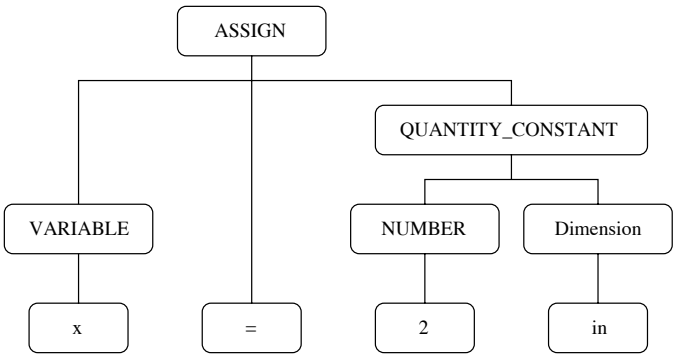


Figure 8. Parse tree for x = 2 in.

The interpreter parses and stores the character sequence "2 in" as the physical quantity two inches. Notice how `2` juxtaposed with `in` implies multiplication; we have hard-coded this interpretation into the scripting language because `2 in` is more customary and easier to read than `2 * in`. This quantity is discarded once the statement has finished executing.

The abstract syntax tree is as follows:

```
Starting PrintAbstractSyntaxTree() ...
======================================== ...

<COMPOUND>
   <ASSIGN>
       <VARIABLE id="x" level="0" />
       <QUANTITY_CONSTANT value="[ 2.000, in]" />
   </ASSIGN>
</COMPOUND>

======================================== ...
Finishing PrintAbstractSyntaxTree() ...
```

Compound statements allow for the modeling of sequences of individual statements. The assignment is defined by two parts, a variable having an identification "x" and a quantity constant having the value 2.0 in.

Internally, the quantity constant is automatically converted to its metric counterpart. Table III shows the name and value of variable "x" as well as details of the units type, scale factor and exponent values.

Case Study 2: Hierarchy of Water Tank Models. The purpose of this example is to see how modules of Java code

```
---------------------------
QUANTITY NAME AND VALUE
---------------------------
Quantity Name  : x
Quantity Value : 0.0508 (m)
-------------------------------------------
UNITS
-------------------------------------------
Units Name      : "in"     Length Exponent : 1
Units Type      : US       Mass Exponent   : 0
Scale Factor    : 0.0254   Time Exponent   : 0
                           Temp Exponent   : 0
                           Radian Exponent : 0
-------------------------------------------
```

TABLE III. SYMBOL TABLE STORAGE FOR QUANTITY x = 2 IN.

can be imported into the scripting language environment and become part of the admissible syntax.

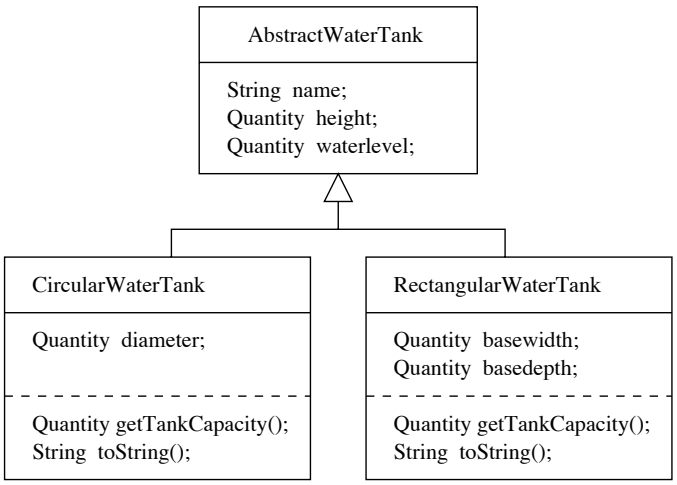


Figure 9. Water tank class hierarchy, annotated with a partial list of variables and methods.

Figure 9 shows a simple class hierarchy for the modeling of water tank components. The AbstractWaterTank class defines concepts and parameters common to all water tanks (e.g., name, waterlevel, height of the tank). The classes RectangularWaterTank and CircularWaterTank add details relevant to tanks with rectangular and circular base areas, respectively. For example, circular water tanks are defined by their diameter. Rectangular water tanks are defined by the parameters basewidth and basedepth. Geometry specific methods are written to compute tank capacities, and so forth.

Now let us assume that the source code for these classes has been compiled in a Java bytecode and references to their specifications are accessible in the JVM (Java Virtual Machine). The fragment of code:

```
import whistle.component.hvac.CircularWaterTank;
```

makes all of the public methods in `CircularWaterTank` and `AbstractWaterTank` available to the library of terms acceptable to the scripting language environment. A circular water tank component with diameter `2 m` and height `2 m` is created by writing:

```
tank01 = CircularWaterTank();
tank01.setDiameter( 2.0 m );
tank01.setHeight( 2 m );
```

The variable `tank01` references an object of type CircularWaterTank stored within the JVM. In a departure from standard programming and scripting languages, which support exchange of basic data types (e.g., float, double) and references to objects in method calls, our philosophy is that participating java classes will work with quantities, matrices of quantities, booleans and strings. Thus, in order to compute and see the tank capacity, we can write:

```
capacity = tank01.getTankCapacity();
print "*** Capacity is: ", capacity;
```

The output is as follows:

```
*** Capacity is: [ 6.283, m^3]
```

Case Study 3: Visualization of Pump Model Data. Pumps (a fan is a pump that moves a gas) are a type of turbomachinery that are generally modeled using empirical data because models based deductively upon first principles of physics can only represent generalized, idealized behavior, not actual specific behavior. Pump performance is difficult to predict because it requires understanding the complex interaction between the pump and the fluid: the shape of the impeller blades, the friction between the blades and the fluid at different temperatures, pressures, and impeller speeds, the details of the pipes and valves upstream and downstream of the pump all have an effect on the pump performance. Manufacturers of pumps create performance curves based on measurements of pumps. The curves show head (pressure), brake horse power, and efficiency as a function of flow rate for a given impeller diameter. The performance of the same pump design with a different impeller diameter, different rotational speed, or different fluid can be calculated from a set of performance curves using the similarity laws. These curves can be used to produce a curve of dimensionless head versus dimensionless flow rate that is more generally useful for incorporation into a modeling program [37], [38].

While the principal purpose of component modeling is for the representation of entities in the physical world, from a scripting perspective, the concept of components extends to services designed to support the analysis and visualization of CPS. To this end, we are in the process of developing data model and visualization components. Figure 10 shows a plot of pump performance data for a size 3, drawthrough 9 inch, BCMpress Fan. Note that the y-axis is dimensionless pressure, where the pressure head is normalized by $\rho * D^2 * N^2$, where ρ is density, D is impeller diameter, and N is rotational speed (rpm). The x-axis is dimensionless flow, where the flow rate

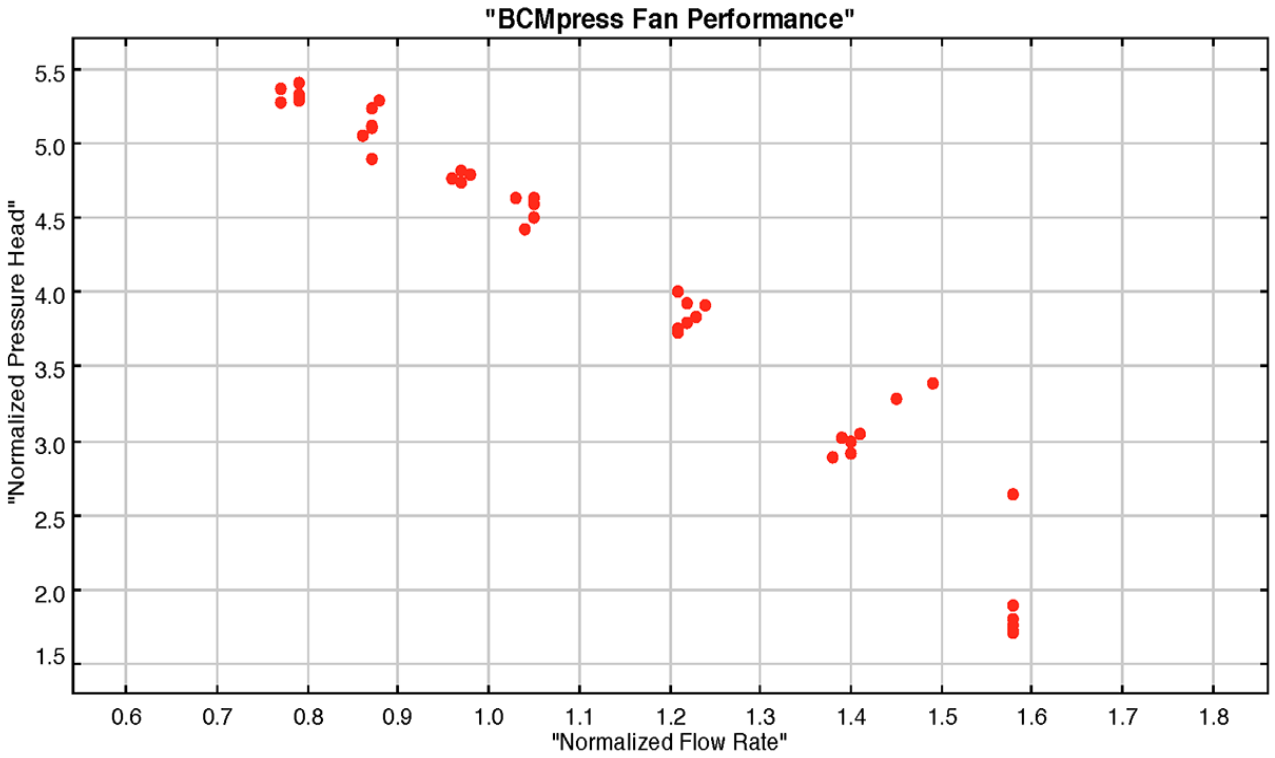


Figure 10. Dimensionless pressure as a function of dimensionless flow of a pump as calculated from standard manufacturer pump curves.

is normalized by $\rho * D^3 * N$. These normalizations are based on idealizations known as "fan laws".

The scripting language specification employs data model and plot components, i.e.,

```
// Import data model from XML file ....

data01 = DataModel();
data01.getData( "pumpModel.xml" );

// Plot pressure head vs discharge rate ...

plot01 = PtPlot();
plot01.setSize( 600, 700 );
plot01.setTitle( "BCMpress Fan Performance");
plot01.setXLabel("Dimensionless Flow Rate Q");
plot01.setYLabel("Dimensionless Pressure Head");

// Transfer data model to plot component ...

c01 = data01.getCurve ("level01");
nsteps = c01.getNoPoints();
for (i = 0; i < nsteps; i = i + 1) {
    plot01.addPoint( c01.getX(i), c01.getY(i) );
}

plot01.display();
```

DataModel() is an experimental component for the storage and management of data models, and their import/export in an xml format. The PtPlot() component is an interface to the PtPlot visualization package distributed with PtolemyII [23].

Case Study 4: Oscillatory Flow between Two Tanks. The language supports the representation of differential equations in their discrete form, and solution via numerical integration techniques.

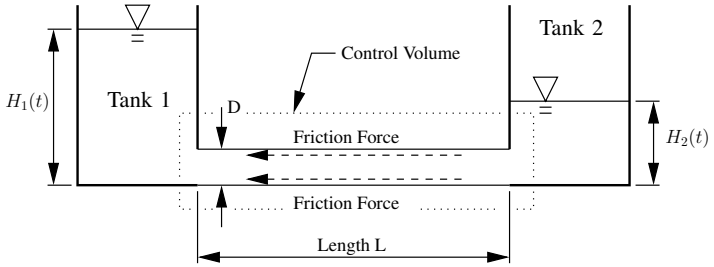


Figure 11. Summary of forces acting on a pipe element connecting two tanks.

Consider, for example, the problem of computing the oscillatory flow of fluid between two tanks as illustrated in Figure 11. Let v(t) and Q(t) be the velocity (m/sec) and flowrate (m^3/sec) in the pipe, measured positive when the flow is from Tank 1 to Tank 2. For a pipe cross section, A_p, and tank cross-section areas A_1 and A_2, conservation of mass implies:

$$Q(t) = A_p v(t) = -A_1 \frac{dH_1(t)}{dt} = A_2 \frac{dH_2(t)}{dt}. \qquad (2)$$

When water depths $H_1(t) \neq H_2(t)$, this "head" differential will cause fluid to flow through the pipe. Transient behavior of the fluid flow is obtained from the equations of momentum balance in the horizontal direction of the control volume, i.e.,

$$\left[\frac{dv(t)}{dt}\right] + \left[\frac{f_1}{2D}\right] v(t)|v(t)| = \left[\frac{g}{L}\right] [H_1(t) - H_2(t)]. \quad (3)$$

Notice that each term in equation (3) has units of acceleration, and that damping forces work to reduce and overall amplitude of accelerations. Damping forces are proportional to pipe roughness and inversely proportional to pipe diameter. The time-history response is computed by creating discrete forms of equations (2) and (3), and systematically integrating the first-order equations of motion with Euler integration. First, the update for momentum balance is given by:

$$v(t + dt) = v(t) + \left[\frac{dv(t)}{dt}\right] dt. \quad (4)$$

Updates in the water depth for each tank are given by:

$$H_1(t + dt) = H_1(t) - \left[\frac{A_p}{A_1}\right] v(t)dt. \quad (5)$$

and

$$H_2(t + dt) = H_2(t) + \left[\frac{A_p}{A_2}\right] v(t)dt. \quad (6)$$

If the tank and pipe components are defined as follows:

```
// Define tank and pipe components ....

tank01 = RectangularWaterTank();
tank01.setName("Tank 01");
tank01.setHeight( 10 m );
tank01.setBaseWidth( 3 m );
tank01.setBaseDepth( 5 m );
tank01.setWaterLevel( 5 m );

tank02 = RectangularWaterTank();
tank02.setName("Tank 02");
tank02.setHeight( 5 m );
tank02.setBaseWidth( 2.0 m );
tank02.setBaseDepth( 2.5 m );
tank02.setWaterLevel( 1 m );

pipe01 = Pipe();
pipe01.setLength(  5.0 m );
pipe01.setRadius( 10.0 cm );
pipe01.setRoughness( 0.005 );
```

then the script:

```
velFluid  = pRough/(4*pRadius)*velOld*Abs(velOld)*dt;
velUpdate = g/pLength*( h01Old - h02Old )*dt;
velNew    = velOld + velUpdate - velFluid;
```

shows the essential details of computing the fluid velocity update with Euler integration. During the executable phases of simulation (right-hand side of Figure 4), the runtime interpreter checks for dimensional consistency of terms in statements before proceeding with their evaluation. Figures 13 and 14 are plots of the tank water levels (m) versus time (sec), and volumetric flow rate (m^3/sec) versus time (sec), respectively.

Case Study 5: Tank with Water Supply and Shut-off Valve. This example, adapted from Turns [39], illustrates the steady and transient states of mass conservation and control volume of a tank with a shut-off valve and water supply system.

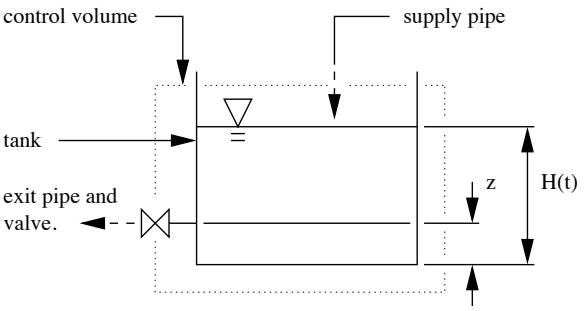


Figure 12. Front elevation of tank, supply pipe, and exit pipe and valve.

The system behavior corresponds to four states as follows: (I) The tank is empty, (II) The tank is being filled to a depth of 1 m, (III) The shut-off valve is opened and the water level is decreasing, (IV) The water level in the tank reaches a steady state and does not change. Based on conservation of mass for an unsteady filling process, we obtain the change in water level from equation (7),

$$\left[\frac{dH(t)}{dt}\right] \rho A_t = \rho v_1 A_1, \quad (7)$$

where $H(t)$ is water height in the tank in (m), ρ is water density and is equal to 997 (kg/m^3), A_t is cross-section area of the tank in (m^2), A_1 is cross-section area of supply pipe in (m^2), v_1 is average velocity of inlet water in (m/sec). When the water height is 1 m, the shut-off valve opens and the height of water in the tank will be updated based on equations:

$$\left[\frac{dH(t)}{dt}\right] \rho A_t = \dot{m}_1 - \dot{m}_2, \quad (8)$$

where $\dot{m}_1$ and $\dot{m}_2$ are the instantaneous mass flow of inlet and outlet pipes in (kg/s):

$$\dot{m}_2 = \rho v_2 A_2, \quad (9)$$

where A_2 is the cross-section area of the outlet pipe in (m^2):

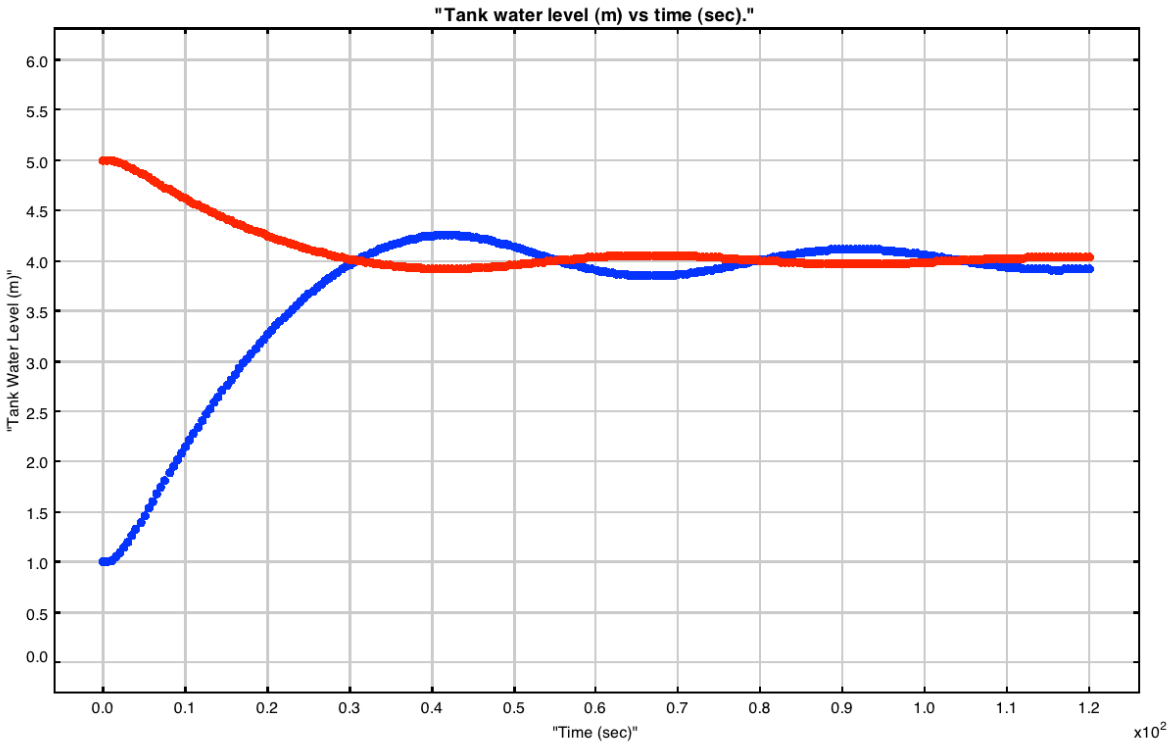


Figure 13. Tank water levels (m) versus time (sec).

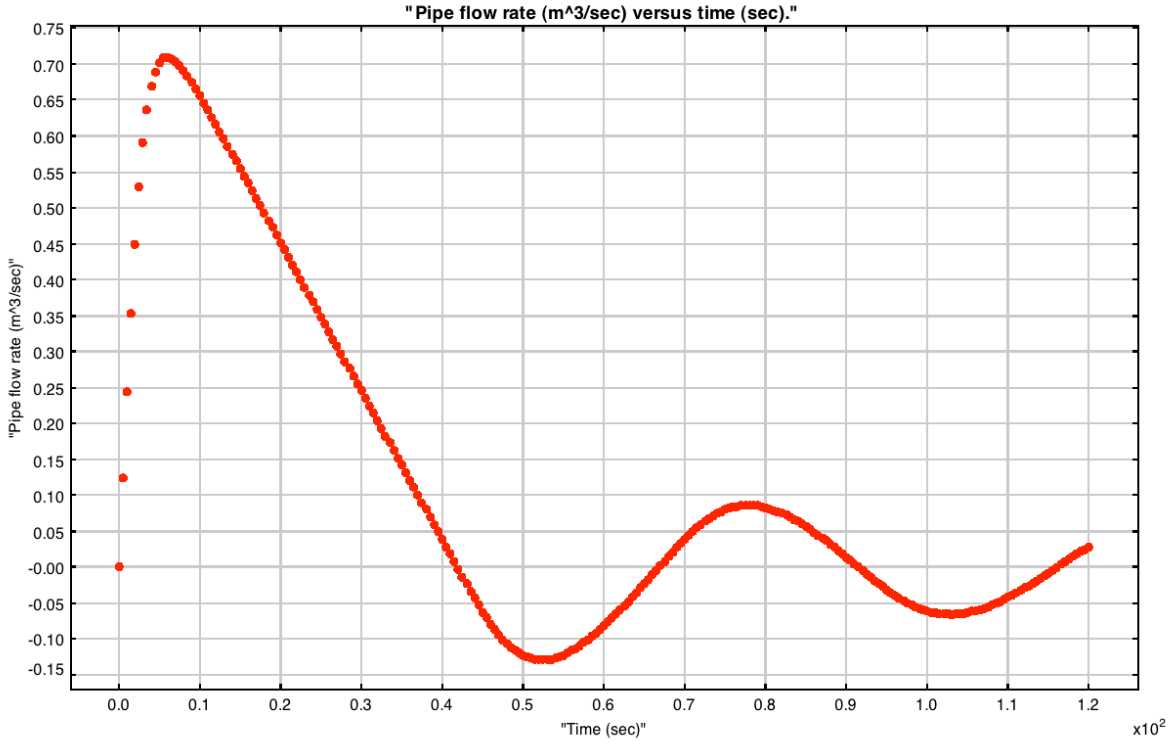


Figure 14. Volumetric flow rate (m^3/sec) versus time (sec).

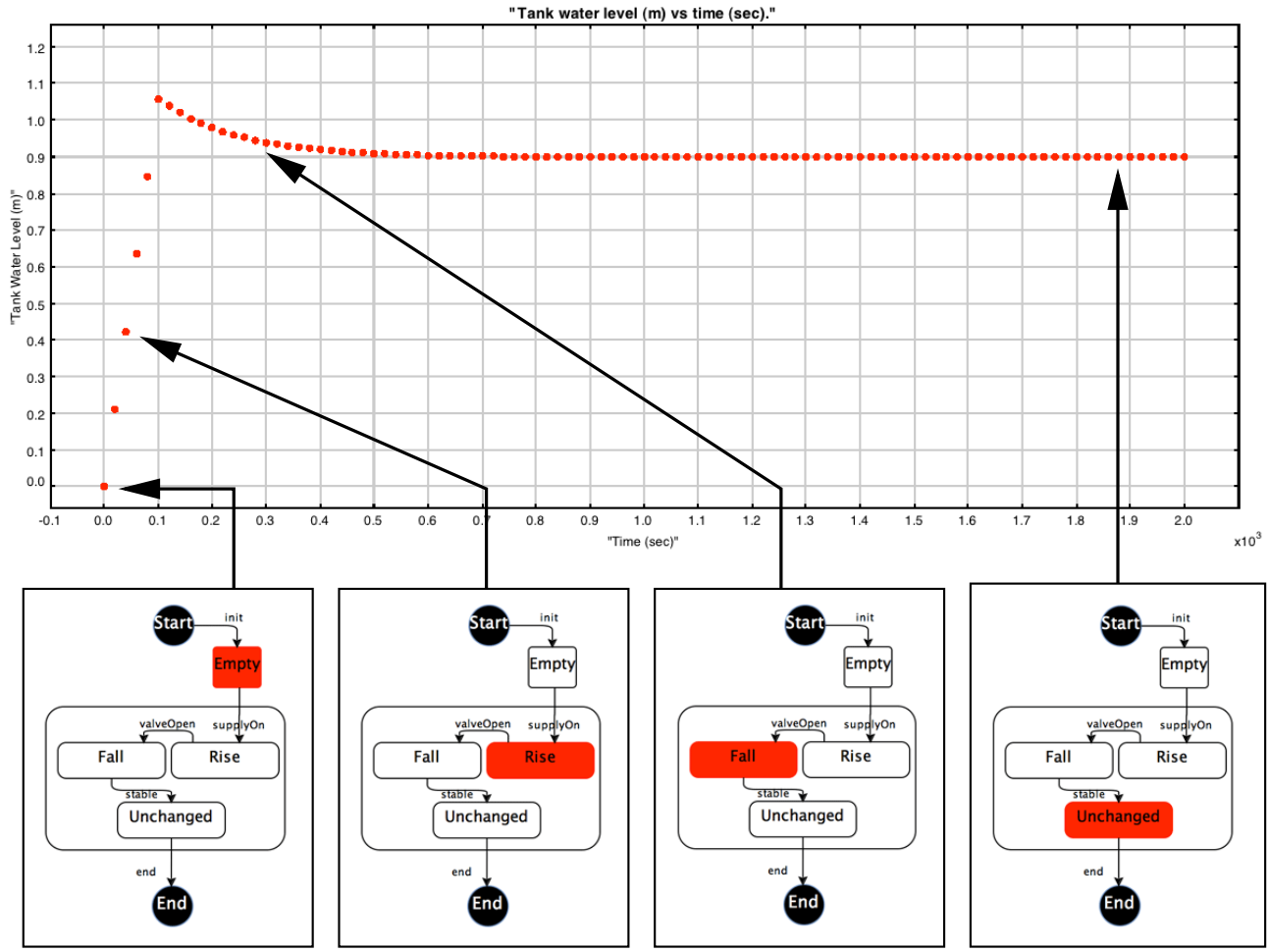


Figure 15. Time-history response for a tank having a water supply and shut-off valve. Upper plot: tank water level (m) versus time (sec). Lower plot: discrete statechart behaviors at various points in the time-history response.

$$\dot{m}_1 = \rho v_1 A_1, \tag{10}$$

$$v_2(t) = 0.85\sqrt{g\,(H(t) - z)}, \tag{11}$$

where $v(t)$ is outlet velocity in (m/s) and z is the location of the shut-off valve in (m). In order to mimic the physical equations, we used the scripting language to model components of the tank, supply, and exit pipes with their associated parameters.

The fragment of script below illustrates the essential details of defining the circular water tank and pipe components:

```
// Define tank and pipe components ....

tank01 = CircularWaterTank();
tank01.setName("Tank 01");
tank01.setDiameter( 1*0.15 m);

// Define supply pipe ....

pipe01 = Pipe();
pipe01.setRadius( 10.0 mm );
```

The heart of the time-history simulation is a looping construct that contains two cases (or discrete states) for physical behavior:

```
// Case 1: Water level is below 1 m:

DepthUpdate =  pipe1Velocity * pipe1Area*dt / tankArea;
DepthNew    =  DepthOld + DepthUpdate;
response01 [i][0] = i * dt;
response01 [i][1] = DepthNew;
DepthOld = DepthNew;

// Case 2: Water level is above 1 m:

massFRSupplyPipe = rho*pipe1Velocity * pipe1Area;

velocityExit   = 0.85*Sqrt(g*(DepthOld - 0.1 m));
massFRExitPipe = rho* velocityExit*pipe02.getArea();

massFlowRateCV = massFRSupplyPipe - massFRExitPipe;

dHeight  = massFRCV/(rho*tankArea)*dt;
DepthNew = DepthOld + dHeight;
response01 [i][0] = i * dt;
```

```
response01 [i][1] = DepthNew;
DepthOld = DepthNew;
```

Figure 15 shows the time-history response of the water level in the tank as it transitions from an empty tank to steady state where the water level remains unchanged t height of 0.9 m. In order to visualize the discrete behavior of this system, we employ our previously developed executable statechart package [10]. This package is capable of modeling and implementation for event-driven behavior with finite state machines. It supports modeling for: (1) Simple, hierarchical and concurrent states, start and final states, (2) History and deep-history pseudostates in hierarchical states, (3) Fork and join pseudostates for concurrent states, (4) Segmented transitions using junction points, and (5) Events, guards and actions for transitions. Visualization of the statechart behaviors is supported through use of mxGraphics in our code. The MVC design pattern (see Section II) is used to make views come alive as models transition through a sequence of states. The abbreviated script:

```
import whistle.statechart.TankStatechart;

....
statechart = TankStatechart();
statechart.startStatechart();
statechart.TransitionEvent(init);

if( DepthOld >= 1 m ){
  statechart.TransitionEvent(valveOpen);
  ....
}
....
```

shows how a statechart element for the water tank is created in an input file developed by the scripting language, and how the language is capable of triggering an event to the statechart when the water level exceeds 1 m. The bottom level of Figure 15 shows how different regions of continuous behavior correspond to the discrete states in the tank statechart.

VI. Conclusion

The purposes of this paper have been two-fold: (1) to describe a semantic platform infrastructure for the model-based systems engineering of cyber-physical systems, and (2) to describe a new and novel scripting language called Whistle. Both efforts are a work in progress. The proposed semantic platform infrastructure will enhance systems engineering practice by lowering validation costs (through rule checking early in design) and providing support for performance assessment during system operation. Our focus in this paper has been to describe the basic features of Whistle, and to show how it can be used to simulate the behavior of a variety of systems characterized by fluid flows and simple control.

Our plans for the future are to conduct research in scripting language design and computational modeling so that Whistle provides the CPS modeling infrastructure and systems integration glue needed to implement the vision of Figures 3 through 5. We envision cyber-physical systems having behaviors that are both distributed and concurrent, and defined by mixtures of local- and global- rule-based control. For the time-history behavior modeling and control of energy-efficient buildings, the finite element method is attractive because problem solutions (e.g., spatial distributions of temperature and pressure in large enclosures) can be formulated from first principles of engineering such as momentum balance. Solution procedures need to be robust, scalable, and extensible to energy-balance calculations. We will design a family of component model interfaces (see the left-hand side of Figure 4), extend them for the implementation of a build components library (e.g., tanks, pipes, valves) and where needed, participate in finite element analysis, actuation, and control. In order for modeling procedures to be efficient we need mechanisms that take advantage of the natural hierarchy of physical systems. Engineers should be provided with the capability to position sensors inside water tanks, and then connect tanks together with networks of pipes and pumps. At the same time, we also need a capability for components to communicate across hierarchies, and we are proposing this be accomplished with listener mechanisms (e.g., a controller component might listen for data from a collection of sensor components and then depending on the water level reading, take an appropriate action). The keys to making this work are software interfaces designed to support a multitude of system viewpoints (e.g., a visualization interface for 2D- and 3D- visualization, a finite element interface for the description of element-level behaviors cast in a matrix format, a communications interface for sensor to controller communication) and Whistle's feature to import and work with references to compiled bytecodes in the Java Virtual Machine. Whistle will act as the glue for systems integration and access to procedures for simulation, visualization and system assessment.

Acknowledgment

The work reported here is part of a US National Institute of Standards and Technology (NIST) funded program dedicated to the development of standards for CPS design, modeling, construction, verification and validation.

References

[1] P. Delgoshaei, M.A. Austin, and D.A. Veronica, "A semantic platform infrastructure for requirements traceability and system assessment," The Ninth International Conference on Systems (ICONS 2014), February 2014, pp. 215-219, ISBN: 978-1-61208-319-3.

[2] NIST. "Strategic R&D opportunities for 21st Cyber-physical systems: connecting computer and information systems with the physical world," National Institute of Science and Technology(NIST), Gaithersburg, MD, USA, 2013.

[3] J. Wing, "Cyber-physical systems research challenges," National Workshop on High-Confidence Automotive Cyber-Physical Systems, Troy, MI, USA, 2008.

[4] M.A. Austin,V. Mayank, and N. Shmunis, "Ontology-based validation of connectivity relationships in a home theater system," The 21st International Journal of Intelligent Systems, Vol. 21, No. 10, pp. 1111–1125, October 2006.

[5] M.A. Austin, V. Mayank, and N. Shmunis, "PaladinRM: graph-based visualization of requirements organized for team-based design," Systems Engineering: The Journal of the International Council on Systems Engineering, Vol. 9, No. 2, pp. 129–145, May 2006.

[6] N. Nassar and M.A. Austin, "Model-based systems engineering design and trade-off analysis with RDF graphs," 11th Annual Conference on Systems Engineering Research (CSER 2013), Georgia Institute of Technology, Atlanta, GA, March 19-22, 2013, pp. 216–225, doi:10.1016/j.procs.

[7] S. Fridenthal, A. Moore, and R. Steiner, "A practical guide to SysML," MK-OMG, 2008.

[8] P. Delgoshaei and M.A. Austin, "Software design patterns for ontology-enabled traceability," Conference on Systems Engineering Research (CSER 2011), Redondo Beach, Los Angeles, April 15-16, 2011.

[9] P. Delgoshaei and M.A. Austin, "Software patterns for traceability of requirements to finite-state machine behavior: application to rail transit systems design and management," The 22nd Annual International Symposium of The International Council on Systems Engineering (INCOSE 2012), Rome, Italy, 2012, pp. 2141-2155, DOI: 10.1002/j.2334-5837.2012.tb01463.x.

[10] P. Delgoshaei, "Software patterns for traceability of requirements to state machine behavior," M.S. Thesis in Systems Engineering, University of Maryland, College Park, MD 20742, November 2012.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: elements of reusable object-oriented software," Addison-Wesley Professional Computing Series, 1995.

[12] OWL w3 See http://www.w3.org/TR/owl-features/ (Accessed, February 2004).

[13] 2013. *Apache Jena*, accessible at: http://www.jena.apache.org.

[14] R. Eckstein, "Java SE application design with MVC," Sun Microsystems, 2007. For more information, see http://www.oracle.com/technetwork/articles/javase/index-142890 (Accessed, November 2014).

[15] M.H. Qusay, "Getting started with the Java rule engine API (JSR 94): toward rule-based applications," Sun Microsystems, 2005. For more information, see http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html (Accessed, March 10, 2008).

[16] G. Rudolf, "Some guidelines for deciding whether to use a rules engine," 2003, Sandia National Labs. For more information see http://herzberg.ca.sandia.gov/guidelines.shtml (Accessed, March 10, 2008).

[17] T. Berners-Lee, J. Hendler, and O. Lassa, "The semantic web," Scientific American, pp. 35–43, May 2001.

[18] P. Derler, E.A Lee, and A.S. Sangiovanni-Vincentelli, "Modeling cyberphysical systems," Proceedings of the IEEE, 100, January 2012.

[19] D. Macpherson and M. Raymond, "Ontology across building, emergency, and energy standards," The Building Service Performance Project, Ontology Summit, 2009.

[20] R. Koelle and W. Strijland, "Semantic driven security assurance for system engineering in SESAR/NextGen," In Integrated Communications, Navigation and Surveillance Conference (ICNS), 2013, pp. k2-1–k2-12.

[21] P. Fritzson, "Principles of Object-Oriented modeling and simulation with Modelica 2.1," Wiley-IEEE Press, 2003.

[22] E.A. Lee, "Finite state machines and models in Ptolemy II," Technical report, EECS Department, University of California, Berkeley, 2009. For more information, see http://ptolemy.eecs.berkeley.edu/ptolemyII (Accessed, August 1, 2014).

[23] C. Brooks, E.A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng, "Heterogeneous concurrent modeling and design in Java (volume 1: introduction to Ptolemy II)," Technical Report ECB/EECS-2008-28, Department Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, April 2008.

[24] J. Lin, S. Sedigh, and A. Miller, "A semantic agent framework for cyber-physical systems," Semantic agent systems studies in computational intelligence, Vol. 344, pp. 189-213, 2011.

[25] G. Simko, D. Lindecker, T. Levendovszky, S. Neema, and J. Sztipanovits, "Specification of cyber-physical components with formal semantics integration and composition," The 16th ACM-IEEE International Conference on Model Driven Engineering Languages and Systems, 2013.

[26] M.A. Austin, X.G. Chen, and W.J. Lin, "ALADDIN: A computational toolkit for interactive engineering matrix and finite element analysis," Technical Research Report TR 95-74, Institute for Systems Research, College Park, MD 20742, August 1995.

[27] M.A. Austin, W.J. Lin, and X.G. Chen, "Structural matrix computations with units," Journal of Computing in Civil Engineering, ASCE, Vol.14, No. 3, pp. 174–182, July 2000.

[28] M.A. Austin, "Matrix and finite element stack machines for structural engineering computations with units," Advances in Engineering Software, Vol. 37, No. 8, pp. 544–559, August 2006.

[29] J.K. Osterhout, "Tcl and the Tk Toolkit," Addison-Wesley Professional Computing Series, Reading, MA 01867, 1994.

[30] L. Wall, T. Christiansen, and R. Schwartz, "Programming Perl," O'Reilly and Associates, Sebastopol, CA 95472, 2nd edition, 1993.

[31] R.L. Schwartz, T. Phoenix, and B.D Foy, "Learning Perl," O'Reilly and Associates, Sebastopol, CA 95472, 4th edition, July 2005.

[32] J. Ousterhout, "Scripting: higher level programming for the 21st century," IEEE Computer Magazine, March 1998.

[33] JFlex -The fast scanner generator for Java: See http://jflex.de/, (Accessed: August 1, 2013).

[34] Berkeley Yacc: See http://invisible-island.net/byacc/, (Accessed: August 1, 2013).

[35] R. Mak, "Writing compilers and interpreters: a software engineering approach (Third Edition)," Wiley Publishing Inc, 2009.

[36] "Unit Conversion Guide," "Fuels and petrochemical division of AICHE," 1990.

[37] F.M. White, "Fluid mechanics (4th Edition)," McGraw-Hill, 1999.

[38] J. Wright, "Building performance simulation for design and optimization," chapter HVAC systems performance and prediction, pp. 312–340, Spon Press (an imprint of Taylor & Francis), London and New York, 2010.

[39] S.R. Turns, "Thermal-fluid sciences: an integrated approach," Cambridge University Press, 2006.

Safety by Construction: Well-behaved Scalable Systems

Peter Ochsenschläger* and Roland Rieke*†
*Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
†Philipps-Universität Marburg, Germany
Email: peter-ochsenschlaeger@t-online.de, roland.rieke@sit.fraunhofer.de

***Abstract*—This paper presents a formal framework that provides construction principles for *well-behaved scalable systems*, such that starting with a prototype system satisfying a desired safety property result in a scalable system satisfying a corresponding safety property, called *scalable safety property*. With respect to different aspects of scalability, the focus of this work is on *property preserving structural scalability*. At that, we consider systems composed of a varying set of individual components where individual components of the same type behave in the same manner, which is characteristic for the type. The respective properties can rely on specific component types and a specific number of individual components but not on the specific individuality of the components. Well-behaved scalable systems are characterised by those systems, which fulfil such a kind of property if already one prototype system (depending on the property) fulfils that property. Sufficient conditions to specify a certain kind of basic well-behaved scalable systems are given and it is shown, how to construct more complex systems by the composition of several synchronisation conditions. Scalable safety properties can be used to express privacy policies as well as security and dependability requirements. It is demonstrated, how the parameterised problem of verifying such a property is reduced to a finite state problem for well-behaved scalable systems. The formal framework for well-behaved scalable systems is developed in terms of prefix closed formal languages and alphabetic language homomorphisms.**

Keywords-uniformly parameterised systems, monotonic parameterised systems, behaviour-abstraction, self-similarity of behaviour, privacy policies, scalable safety properties.

I. Introduction

This article is based on [1], where the concept of well-behaved scalable systems has been introduced. It is extended by extensive proofs of the theorems and the definition of *scalable safety properties* as well as their verification for well-behaved scalable systems. This is illustrated by a complex example, where several synchronisation conditions are composed.

Scalability is a desirable property of systems. However, the term scalability is often not clearly defined and thus it is difficult to characterise and understand systems with respect to their scalability properties [2]. In [3], four aspects of scalability are considered, i.e., load scalability, space scalability, space-time scalability, and structural scalability. In this paper, we focus on *structural scalability*, which is "the ability of a system to expand in a chosen dimension without major modifications to its architecture" [3]. Examples of systems that need to be highly scalable comprise grid computing architectures and cloud computing platforms [4], [5]. Usually, such systems consist of few different types of components and for each such type a varying set of individual components exists. Component types can be defined in such a granularity that individual components of the same type behave in the same manner, which is characteristic for the type. For example, a client-server system that is scalable consists of the component types *client* and *server* and several sets of individual clients as well as several sets of individual servers. Let us now call a choice of sets of individual components an *admissible choice of individual component sets*, iff for each component type exactly one set of individual components of that type is chosen. Then, a "scalable system" can be considered as a family of systems, whose elements are systems composed of a specific admissible choice of individual component sets.

For safety critical systems as well as for business critical systems, assuring the correctness is imperative. Formally, the dynamic behaviour of a discrete system can be described by the set of its possible sequences of actions. This way to model the behaviour is important, because it enables the definition of safety requirements as well as the verification of such properties, because for these purposes sequences of actions of the system have to be considered [6], [7], [8]. For short, we often will use the term *system* instead of *systems behaviour* if it does not generate confusions. With this focus, scalable systems are families of system behaviours, which are indexed by admissible choices of individual component sets. We call such families *parameterised systems*. In this paper, we define *well-behaved scalable systems* as a special class of parameterised systems and develop construction principles for such systems. The main goal for this definition is to achieve that well-behaved scalable systems fulfil certain kind of safety properties if already one prototype system (depending on the property) fulfils that property (cf. Section IV). To this end, construction principles for well-behaved scalable systems are *design principles for*

verifiability [9]. We give an example that demonstrates the significance of self-similarity for verification purposes and show that for well-behaved scalable systems scalable safety properties can be verified by finite state methods.

The main content of the paper can basically be divided into three parts. Besides the basic definitions, the first part (Section III and Section IV) comprises a characterisation of the systems under consideration and their fundamental properties. The second part (Section V and Section VI, enriched by the appendix) provides the formal framework for the construction of well-behaved systems. The last part (Section VII) provides a generic verification scheme for scalable safety properties and presents an example for its application. Concluding remarks and further research directions are given in Section VIII.

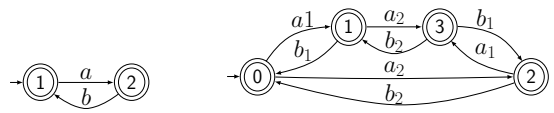
II. Related Work

Considering the behaviour-verification aspect, which is one of our motivations to formally define well-behaved scalable systems, there are some other approaches to be mentioned. An extension to the Murφ verifier to verify systems with replicated identical components through a new data type called RepetitiveID is presented in [10]. The verification is performed by explicit state enumeration in an abstract state space where states do not record the exact numbers of components. A typical application area of this tool are cache coherence protocols. The aim of [11] is an abstraction method through symmetry, which works also when using variables holding references to other processes. In [12], a methodology for constructing abstractions and refining them by analysing counter-examples is presented. The method combines abstraction, model-checking and deductive verification. A technique for automatic verification of parameterised systems based on process algebra CCS [13] and the logic modal mu-calculus [14] is presented in [15]. This technique views processes as property transformers and is based on computing the limit of a sequence of mu-calculus [14] formulas generated by these transformers. The above-mentioned approaches demonstrate that finite state methods combined with deductive methods can be applied to analyse parameterised systems. The approaches differ in varying amounts of user intervention and their range of application. A survey of approaches to combine model checking and theorem proving methods is given in [16]. Far reaching results in verifying parameterised systems by model checking of corresponding abstract systems are given in [17], [18]. It is well known that the general verification problem for parameterised systems is undecidable [19], [20]. To handle that problem, we present (a) a formal framework to specify parameterised systems in a restricted manner, and (b) construction principles for well-behaved scalable systems.

III. Characterisation of Scalable Systems

The behaviour L of a discrete system can be formally described by the set of its possible sequences of actions. Therefore, $L \subset \Sigma^*$ holds where Σ is the set of all actions of the system, and Σ^* (free monoid over Σ) is the set of all finite sequences of elements of Σ, including the empty sequence denoted by ε. This terminology originates from the theory of formal languages [21], where Σ is called the alphabet (not necessarily finite), the elements of Σ are called letters, the elements of Σ^* are referred to as words and the subsets of Σ^* as formal languages. Words can be composed: if u and v are words, then uv is also a word. This operation is called the *concatenation*; especially $\varepsilon u = u\varepsilon = u$. A word u is called a *prefix* of a word v if there is a word x such that $v = ux$. The set of all prefixes of a word u is denoted by $\mathrm{pre}(u)$; $\varepsilon \in \mathrm{pre}(u)$ holds for every word u. Formal languages, which describe system behaviour, have the characteristic that $\mathrm{pre}(u) \subset L$ holds for every word $u \in L$. Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages. Different formal models of the same system are partially ordered with respect to different levels of abstraction. Formally, abstractions are described by alphabetic language homomorphisms. These are mappings $h^* : \Sigma^* \longrightarrow \Sigma'^*$ with $h^*(xy) = h^*(x)h^*(y)$, $h^*(\varepsilon) = \varepsilon$ and $h^*(\Sigma) \subset \Sigma' \cup \{\varepsilon\}$. So, they are uniquely defined by corresponding mappings $h : \Sigma \longrightarrow \Sigma' \cup \{\varepsilon\}$. In the following, we denote both the mapping h and the homomorphism h^* by h. We consider a lot of alphabetic language homomorphisms. So, for simplicity we tacitly assume that a mapping between free monoids is an alphabetic language homomorphism if nothing contrary is stated. We now introduce a guiding example.

Example 1. *A server answers requests of a family of clients. The actions of the server are considered in the following. We assume with respect to each client that a request will be answered before a new request from this client is accepted. If the family of clients consists of only one client, then the automaton in Fig. 1(a) describes the system behaviour $S \subset \Sigma^*$, where $\Sigma = \{a, b\}$, the label a depicts the request, and b depicts the response.*



(a) Actions at a server with respect to a client (b) Two clients served concurrently by one server

Figure 1. Scalable client-server system

Example 2. *Fig. 1(b) now describes the system behaviour $S_{\{1,2\}} \subset \Sigma^*_{\{1,2\}}$ for two clients 1 and 2, under the*

assumption that the server handles the requests of different clients non-restricted concurrently.

For a parameter set I and $i \in I$ let $\Sigma_{\{i\}}$ denote pairwise disjoint copies of Σ. The elements of $\Sigma_{\{i\}}$ are denoted by a_i and $\Sigma_I := \bigcup_{i \in I} \Sigma_{\{i\}}$, where $\Sigma_{\{j\}} \cap \Sigma_{\{k\}} = \emptyset$ for $j \neq k$. The index i describes the bijection $a \leftrightarrow a_i$ for $a \in \Sigma$ and $a_i \in \Sigma_{\{i\}}$.

Example 3. *For $\emptyset \neq I \subset \mathbb{N}$ with finite I, let now $S_I \subset \Sigma_I^*$ denote the system behaviour with respect to the client set I. For each $i \in \mathbb{N}$ $S_{\{i\}}$ is isomorphic to S, and S_I consists of the non-restricted concurrent run of all $S_{\{i\}}$ with $i \in I$.*

It holds $S_{I'} \subset S_I$ for $I' \subset I$.

Let $\mathcal{I}_1$ denote the set of all finite non-empty subsets of $\mathbb{N}$ (the set of all possible clients). Then, the family $(S_I)_{I \in \mathcal{I}_1}$ is an example of a monotonic parameterised system.

If the example is extended to consider several servers, which are depicted by natural numbers, then, e.g.,

$$\mathcal{I}_2 := \{\mathring{I} \times \hat{I} \subset \mathbb{N} \times \mathbb{N} | \mathring{I} \neq \emptyset \neq \hat{I}, \text{ with } \mathring{I}, \hat{I} \text{ finite}\}$$

is a suitable parameter structure.

$\mathcal{I}_2$ used in the example above shows how the component structure of a system can be expressed by a parameter structure using Cartesian products of individual component sets. The following Definition 1 abstracts from the intuition of a component structure.

Definition 1 (parameter structure)**.** *Let N be a countable (infinite) set and $\emptyset \neq \mathcal{I} \subset \mathfrak{P}(N) \setminus \{\emptyset\}$. $\mathcal{I}$ is called a* parameter structure *based on N.*

For scalable systems it is obvious to assume that enlarging the individual component sets does not reduce the corresponding system behaviour. More precisely: let I and K be two arbitrary admissible choices of individual component sets, where each individual component set in I is a subset of the corresponding individual component set in K. If S_I and S_K are the corresponding systems' behaviours, then S_I is a subset of S_K. Families of systems with this property we call *monotonic parameterised systems*. The following definition formalises monotonic parameterised systems.

Definition 2 (monotonic parameterised system)**.** *Let $\mathcal{I}$ be a* parameter structure*. For each $I \in \mathcal{I}$ let $\mathcal{L}_I \subset \Sigma_I^*$ be a prefix closed language. If $\mathcal{L}_{I'} \subset \mathcal{L}_I$ for each $I, I' \in \mathcal{I}$ with $I' \subset I$, then $(\mathcal{L}_I)_{I \in \mathcal{I}}$ is a* monotonic parameterised system*.*

As we assume that individual components of the same type behave in the same manner, S_I and S_K are isomorphic (equal up to the names of the individual components), if I and K have the same cardinality. This property we call *uniform parameterisation*. With these notions we define *scalable systems* as *uniformly monotonic parameterised systems*. Monotonic parameterised systems, in which isomorphic subsets of parameter values describe isomorphic subsystems, we call *uniformly monotonic parameterised systems*.

Definition 3 (isomorphism structure)**.** *Let $\mathcal{I}$ be a parameter structure, $I, K \in \mathcal{I}$, and $\iota : I \to K$ a bijection, then let $\iota_K^I : \Sigma_I^* \to \Sigma_K^*$ the isomorphism defined by*

$$\iota_K^I(a_i) := a_{\iota(i)} \text{ for } a_i \in \Sigma_I.$$

For each $I, K \in \mathcal{I}$ let $\mathcal{B}(I,K) \subset K^I$ a set (possibly empty) of bijections. $\mathcal{B}_\mathcal{I} := (\mathcal{B}(I,K))_{(I,K) \in \mathcal{I} \times \mathcal{I}}$ is called an isomorphism structure *for $\mathcal{I}$.*

Definition 4 (scalable system)**.** *Let $(\mathcal{L}_I)_{I \in \mathcal{I}}$ a monotonic parameterised system and $\mathcal{B}_\mathcal{I} = (\mathcal{B}(I,K))_{(I,K) \in \mathcal{I} \times \mathcal{I}}$ an* isomorphism structure *for $\mathcal{I}$.*
$(\mathcal{L}_I)_{I \in \mathcal{I}}$ is called uniformly monotonic parameterised with respect to $\mathcal{B}_\mathcal{I}$ *iff*

$$\mathcal{L}_K = \iota_K^I(\mathcal{L}_I) \text{ for each } I, K \in \mathcal{I} \text{ and each } \iota \in \mathcal{B}(I,K).$$

Uniformly monotonic parameterised systems for short are called scalable systems*.*

Example 4. *Let $\mathcal{I} = \mathcal{I}_2$.*

$$\mathcal{B}^2(\mathring{I} \times \hat{I}, \mathring{K} \times \hat{K}) := \{\iota \in (\mathring{K} \times \hat{K})^{(\mathring{I} \times \hat{I})} \mid \text{it exist bijections } \mathring{\iota} : \mathring{I} \to \mathring{K} \text{ and } \hat{\iota} : \hat{I} \to \hat{K} \text{ with } \iota((r,s)) = (\mathring{\iota}(r), \hat{\iota}(s)) \text{ for each } (r,s) \in (\mathring{I} \times \hat{I})\}$$

for $\mathring{I} \times \hat{I} \in \mathcal{I}_2$ and $\mathring{K} \times \hat{K} \in \mathcal{I}_2$ defines an isomorphism structure *$\mathcal{B}^2_{\mathcal{I}_2}$.*

IV. Well-behaved Scalable Systems

To motivate our formalisation of *well-behaved*, we consider a typical security requirement of a scalable client-server system: Whenever two different clients cooperate with the same server then certain critical sections of the cooperation of one client with the server must not overlap with critical sections of the cooperation of the other client with the same server. If for example both clients want to use the same resource of the server for confidential purposes, then the allocation of the resource to one of the clients has to be completely separated from the allocation of this resource to the other client. More generally, the concurrent cooperation of one server with several clients has to be restricted by certain *synchronisation conditions* to prevent, for example, undesired race conditions.

According to this example, we focus on properties, which rely on specific component types and a specific number of individual components for these component types but not on the specific individuality of the individual components. Now, we want to achieve that a well

behaved scalable system fulfils such a kind of property if already one *prototype system* (depending on the property) fulfils that property. In our example, a prototype system consists of two specific clients and one specific server.

To formalise this desire, we consider arbitrary I and K as in the definition of monotonic parameterised system. Then we look at S_K from an abstracting point of view, where only actions corresponding to the individual components of I are considered. If the smaller subsystem S_I behaves like the abstracted view of S_K, then we call this property *self-similarity* or more precisely *self-similarity of scalable systems*, to distinguish our notion from geometric oriented notions [22] and organisational aspects [23] of self-similarity. In [7], it is shown that *self-similar uniformly monotonic parameterised systems* have the above desired property. Therefore, we define *well-behaved scalable systems* as self-similar uniformly monotonic parameterised systems. We now formally look at $\mathcal{L}_I$ from an abstracting point of view concerning a subset $I' \subset I$. The corresponding abstractions are formalised by the homomorphisms $\Pi^I_{I'} : \Sigma^*_I \to \Sigma^*_{I'}$.

Definition 5 (self-similar monotonic parameterised system)**.** *For $I' \subset I$ let $\Pi^I_{I'} : \Sigma^*_I \to \Sigma^*_{I'}$ with*

$$\Pi^I_{I'}(a_i) = \left\{ \begin{array}{rl} a_i \mid & a_i \in \Sigma_{I'} \\ \varepsilon \mid & a_i \in \Sigma_I \setminus \Sigma_{I'}. \end{array} \right.$$

A monotonic parameterised system $(\mathcal{L}_I)_{I\in\mathcal{I}}$ is called self-similar *iff $\Pi^I_{I'}(\mathcal{L}_I) = \mathcal{L}_{I'}$ for each $I, I' \in \mathcal{I}$ with $I' \subset I$.*

Definition 6 (well-behaved scalable system)**.** *Self-similar scalable systems for short are called* well-behaved scalable systems*.*

A fundamental construction principle for systems satisfying several constraints is intersection of system behaviours. This emphasises the importance of the following theorem.

Theorem 1 (intersection theorem)**.** *Let $\mathcal{I}$ be a parameter structure, $\mathcal{B}_\mathcal{I}$ an isomorphism structure for $\mathcal{I}$, and $T \neq \emptyset$.*

i) *Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ for each $t \in T$ be a monotonic parameterised system, then $(\bigcap_{t\in T} \mathcal{L}^t_I)_{I\in\mathcal{I}}$ is a monotonic parameterised system.*
ii) *Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ for each $t \in T$ be a scalable system with respect to $\mathcal{B}_\mathcal{I}$, then $(\bigcap_{t\in T} \mathcal{L}^t_I)_{I\in\mathcal{I}}$ is a scalable system with respect to $\mathcal{B}_\mathcal{I}$.*
iii) *Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ for each $t \in T$ be a self-similar monotonic parameterised system, then $(\bigcap_{t\in T} \mathcal{L}^t_I)_{I\in\mathcal{I}}$ is a self-similar monotonic parameterised system.*

Proof of Theorem 1 (i)–(iii):

Proof of (i): Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ a monotonic parameterised system for each $t \in T$, then $\mathcal{L}^t_{I'} \subset \mathcal{L}^t_I$ for $t \in T$, $I, I' \in \mathcal{I}$, and $I' \subset I$. This implies

$$\bigcap_{t\in T} \mathcal{L}^t_{I'} \subset \bigcap_{t\in T} \mathcal{L}^t_I,$$

and thus (i).

Proof of (ii): Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ an scalable system with respect to $(\mathcal{B}(I,K))_{(I,K)\in\mathcal{I}\times\mathcal{I}}$ for each $t \in T$, then $\iota^I_K(\mathcal{L}^t_I) = \mathcal{L}^t_K$ for $t \in T$, I, $K \in \mathcal{I}$, and $\iota \in \mathcal{B}(I,K)$.

Because all ι^I_K are isomorphisms,

$$\iota^I_K(\bigcap_{t\in T} \mathcal{L}^t_I) = \bigcap_{t\in T} \iota^I_K(\mathcal{L}^t_I) = \bigcap_{t\in T} \mathcal{L}^t_K,$$

which proves (ii).

Proof of (iii): Let $(\mathcal{L}^t_I)_{I\in\mathcal{I}}$ a self-similar monotonic parameterised system for each $t \in T$. For $I, I' \in \mathcal{I}$ with $I' \subset I$ holds

$$\Pi^I_{I'}(\bigcap_{t\in T} \mathcal{L}^t_I) \subset \bigcap_{t\in T} \Pi^I_{I'}(\mathcal{L}^t_I) = \bigcap_{t\in T} \mathcal{L}^t_{I'} \subset \bigcap_{t\in T} \mathcal{L}^t_I. \quad (1)$$

Because $\bigcap_{t\in T} \mathcal{L}^t_{I'} \subset \Sigma^*_{I'}$ holds

$$\Pi^I_{I'}(\bigcap_{t\in T} \mathcal{L}^t_{I'}) = \bigcap_{t\in T} \mathcal{L}^t_{I'}.$$

Together with the second inclusion from (1) it follows

$$\bigcap_{t\in T} \mathcal{L}^t_{I'} \subset \Pi^I_{I'}(\bigcap_{t\in T} \mathcal{L}^t_I).$$

Because of the first part of (1) now holds

$$\Pi^I_{I'}(\bigcap_{t\in T} \mathcal{L}^t_I) = \bigcap_{t\in T} \mathcal{L}^t_{I'},$$

which proves (iii). ■

Weak additional assumptions for well-behaved scalable systems imply that such systems are characterised by parametrisation of one well-defined minimal prototype system. More precisely:

Definition 7 (minimal prototype system)**.** *Let $\mathcal{I}$ be a parameter structure based on N. For $I \in \mathcal{I}$ and $n \in N$ let $\tau^I_n : \Sigma^*_I \to \Sigma^*$ the homomorphisms given by*

$$\tau^I_n(a_i) = \left\{ \begin{array}{rl} a \mid & a_i \in \Sigma_{I\cap\{n\}} \\ \varepsilon \mid & a_i \in \Sigma_{I\setminus\{n\}} \end{array} \right. .$$

For a singleton index set $\{n\}$, $\tau^{\{n\}}_n : \Sigma^*_{\{n\}} \to \Sigma^*$ is an isomorphism and for each $n \in I \in \mathcal{I}$ holds

$$\Pi^I_{\{n\}} = (\tau^{\{n\}}_n)^{-1} \circ \tau^I_n. \quad (2)$$

If now $(\mathcal{L}_I)_{I\in\mathcal{I}}$ is a well-behaved scalable system with respect to $(\mathcal{B}(I,K))_{(I,K)\in\mathcal{I}\times\mathcal{I}}$ with $\{n\} \in \mathcal{I}$ for $n \in I \in \mathcal{I}$ and $\mathcal{B}(I,K) \neq \emptyset$ for all singleton I and K, then because of (2) holds

$$\mathcal{L}_I \subset \bigcap_{n\in I} (\tau^I_n)^{-1}(L) \text{ for each } I \in \mathcal{I},$$

where $L = \tau_n^{\{n\}}(\mathcal{L}_{\{n\}})$ *for each* $n \in \bigcup_{I \in \mathcal{I}} I$.

L *is called the* minimal prototype system *of* $(\mathcal{L}_I)_{I \in \mathcal{I}}$.

Definition 8 (behaviour-family $(\dot{\mathcal{L}}(L)_I)_{I \in \mathcal{I}}$ generated by the minimal prototype system L and the parameter structure $\mathcal{I}$)**.** *Let* $\emptyset \neq L \subset \Sigma^*$ *be prefix closed,* $\mathcal{I}$ *a parameter structure, and*

$$\dot{\mathcal{L}}(L)_I := \bigcap_{i \in I} (\tau_i^I)^{-1}(L) \text{ for } I \in \mathcal{I}.$$

The systems $\dot{\mathcal{L}}(L)_I$ consist of the "non-restricted concurrent run" of all systems $(\tau_i^{\{i\}})^{-1}(L) \subset \Sigma_{\{i\}}^*$ with $i \in I$. Because $\tau_i^{\{i\}} : \Sigma_{\{i\}}^* \to \Sigma^*$ are isomorphisms, $(\tau_i^{\{i\}})^{-1}(L)$ are pairwise disjoint copies of L.

Theorem 2 (simplest well-behaved scalable systems)**.** $(\dot{\mathcal{L}}(L)_I)_{I \in \mathcal{I}}$ *is a well-behaved scalable system with respect to each isomorphism structure for* $\mathcal{I}$ *based on* N *and*

$$\dot{\mathcal{L}}(L)_I = \bigcap_{i \in N} (\tau_i^I)^{-1}(L) \text{ for each } I \in \mathcal{I}.$$

The proof of this theorem is given in the appendix.

V. Construction of Well-behaved Systems by Restriction of Concurrency

Now, we show how to construct well-behaved systems by restricting concurrency in the behaviour-family $\dot{\mathcal{L}}$. In Example 3, holds $S_I = \dot{\mathcal{L}}(S)_I$ for $I \in \mathcal{I}_1$. If, in the given example, the server needs specific resources for the processing of a request, then - on account of restricted resources - an non-restricted concurrent processing of requests is not possible. Thus, restrictions of concurrency in terms of synchronisation conditions are necessary. One possible but very strong restriction is the requirement that the server handles the requests of different clients in the same way as it handles the requests of a single client, namely, on the request follows the response and vice versa. This synchronisation condition can be formalised with the help of S and the homomorphisms Θ^I as shown in the following example.

Example 5. *Restriction of concurrency on account of restricted resources: one "task" after another. All behaviours with respect to* $i \in I$ influence *each other. Let*

$$\bar{S}_I := S_I \cap (\Theta^I)^{-1}(S) = \bigcap_{i \in I} (\tau_i^I)^{-1}(S) \cap (\Theta^I)^{-1}(S)$$

for $I \in \mathcal{I}_1$*, where generally, for each index set* I*,* $\Theta^I : \Sigma_I^* \to \Sigma^*$ *is defined by* $\Theta^I(a_i) := a$*, for* $i \in I$ *and* $a \in \Sigma$.

From the automaton in Fig. 1(b), it is evident that $\bar{S}_{\{1,2\}}$ will be accepted by the automaton in Fig. 2(a).

Given an arbitrary $I \in \mathcal{I}_1$, then $\bar{S}_I$ is accepted by an automaton with state set $\{0\} \cup I$ and state transition relation given by Fig. 2(b) for each $i \in I$.

(a) Automaton accepting $\bar{S}_{\{1,2\}}$ (b) Automaton accepting $\bar{S}_I$

Figure 2. Automata accepting $\bar{S}_{\{1,2\}}$ and $\bar{S}_I$

From this automaton, it is evident that $(\bar{S}_I)_{I \in \mathcal{I}_1}$ is a well-behaved scalable system, with respect to each isomorphism structure $\mathcal{B}_{\mathcal{I}_1}$ for $\mathcal{I}_1$.

Example 6. *A restriction of concurrency in the extended example where a family of servers is involved is more complicated than in the case of* $(\bar{S}_I)_{I \in \mathcal{I}_1}$*. The reason for that is that in the simple example the restriction of concurrency can be formalised by a restricting influence of the actions with respect to all parameter values (i.e., the entire* Σ_I*). When considering the restriction of concurrency in the extended example, the actions influence each other only with respect to the parameter values, which are bound to the same server.*

Let the first component of the elements from $\mathbb{N} \times \mathbb{N}$ *in the parameter structure* $\mathcal{I}_2$ *denote the server, then the actions from* $\Sigma_{\{r\} \times \hat{I}}$ *influence each other for given* $r \in \mathring{I}$ *with* $\mathring{I} \times \hat{I} \in \mathcal{I}$ *and thus restrict the concurrency.*

For the formalisation of this restriction of concurrency, we now consider the general case of monotonic parameterised systems $(\dot{\mathcal{L}}(L)_I)_{I \in \mathcal{I}}$. As already observed in (2), for each well-behaved scalable system $(\mathcal{L}_I)_{I \in \mathcal{I}}$ there exists (under weak preconditions) a system $(\dot{\mathcal{L}}(L)_I)_{I \in \mathcal{I}}$ with $\mathcal{L}_I \subset \dot{\mathcal{L}}(L)_I$ for each $I \in \mathcal{I}$, where $L = \tau_n^{\{n\}}(\mathcal{L}_{\{n\}})$ for each $n \in I \in \mathcal{I}$. Moreover, in context of Definition 8 it was observed that $\dot{\mathcal{L}}(L)_I$ consists of the non-restricted concurrent run of pairwise disjoint copies of L.

In conjunction, this shows that an adequate restriction of concurrency in $(\dot{\mathcal{L}}(L)_I)_{I \in \mathcal{I}}$ can lead to the construction of well-behaved scalable systems. Therefore, the restricting influence of actions with respect to specific parameter values described above shall now be formalised.

Definition 9 (influence structure)**.** *Let* $T \neq \emptyset$ *and* $\mathcal{I}$ *a parameter structure. For each* $I \in \mathcal{I}$ *and* $t \in T$ *a* sphere of influence *is specified by* $E(t, I) \subset I$*. The family*

$$\mathcal{E}_{\mathcal{I}} = (E(t, I))_{(t,I) \in T \times \mathcal{I}}$$

is called influence structure *for* $\mathcal{I}$ *indexed by* T.

The non-restricted concurrent run of the pairwise disjoint copies of L will now be restricted in the following way: For each $t \in T$ the runs of all copies k with $k \in E(t, I)$ influence each other independently of the specific values of $k \in E(t, I)$. With respect to our extended example (several servers) with $\mathcal{I}_2$, the spheres of influence $E(t, I)$

are generalisations of the sets $\{r\} \times \hat{I}$, where $I = \mathring{I} \times \hat{I}$ and $t = (r,s) \in \mathring{I} \times \hat{I}$.

Generally, for each $t \in T$ the intersection

$$\dot{\mathcal{L}}(L)_I \cap (\tau^I_{E(t,I)})^{-1}(V) \tag{3}$$

formalises the restriction of the non-restricted concurrent run of the copies of L within $\dot{\mathcal{L}}(L)_I$ by the mutual influence of each element of $E(t,I)$.

Definition 10 (behaviour of influence and influence homomorphisms)**.** *In* (3)*, the* behaviour of influence V *is a prefix closed language $V \subset \Sigma^*$, and for $I, I' \subset N$ the homomorphism $\tau^I_{I'} : \Sigma^*_I \to \Sigma^*$ is defined by:*

$$\tau^I_{I'}(a_i) = \begin{cases} a \mid & a_i \in \Sigma_{I \cap I'} \\ \varepsilon \mid & a_i \in \Sigma_{I \setminus I'} \end{cases} .$$

The homomorphisms $\tau^I_{E(t,I)}$ are called the influence homomorphisms *of $\mathcal{E}_\mathcal{I}$.*

Definition 11 (behaviour-family $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ generated by the minimal prototype system L, the influence structure $\mathcal{E}_\mathcal{I}$, and the behaviour of influence V)**.** *Because the restriction* (3) *shall hold for all $t \in T$, the restricted systems $\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I$ are defined by the prefix closed languages*

$$\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I := \dot{\mathcal{L}}(L)_I \cap \bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V) \text{ for } I \in \mathcal{I}.$$

Definition 11 shows how synchronisation requirements for the systems $\dot{\mathcal{L}}(L)_I$ can be formalised by influence structures and behaviour of influence in a very general manner. Since, similar to the well-behaved scalable systems $(\dot{\mathcal{L}}(L)_I)_{I\in\mathcal{I}}$, in the systems $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ each $\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_{\{i\}}$ shall be isomorphic to L for each $\{i\} \in \mathcal{I}$, $V \supset L$ has to be assumed. Therefore, in general we assume for systems $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ that $V \supset L \neq \emptyset$. Note that $\tau^I_{I'}$ are generalisations of τ^I_n and Θ^I, because

$$\tau^I_n = \tau^I_{\{n\}} \text{ and } \Theta^I = \tau^I_I = \tau^I_N$$

for each $I \subset N$ and $n \in N$.

Further requirements, which assure that $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ are well-behaved scalable systems, will now be given with respect to $\mathcal{E}_\mathcal{I}$, $\mathcal{B}_\mathcal{I}$, L and V. Assuming $T = N$ and $\varepsilon \in V$ the scalability property is assured by the following technical requirements for $\mathcal{E}_\mathcal{I}$ and $\mathcal{B}_\mathcal{I}$:

Theorem 3 (construction condition for scalable systems)**.** *Let $\mathcal{I}$ be a parameter structure based on N, $\mathcal{E}_\mathcal{I} = (E(n,I))_{(n,I)\in N\times\mathcal{I}}$ be an influence structure for $\mathcal{I}$, and let $\mathcal{B}_\mathcal{I} = (\mathcal{B}(I,I'))_{(I,I')\in\mathcal{I}\times\mathcal{I}}$ be an isomorphism structure for $\mathcal{I}$. Let $\varepsilon \in V \subset \Sigma^*$, for each $I \in \mathcal{I}$ and $n \in N$ let $E(n,I) = \emptyset$, or it exists an $i_n \in I$ with $E(n,I) = E(i_n,I)$, and for each $(I,I') \in \mathcal{I}\times\mathcal{I}, \iota \in \mathcal{B}(I,I')$ and $i \in I$ holds*

$$\iota(E(i,I)) = E(\iota(i),I').$$

Let $E(t,I') = E(t,I) \cap I'$ for each $t \in T$ and $I, I' \in \mathcal{I}, I' \subset I$. Then $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ is a scalable system with respect to $\mathcal{B}_\mathcal{I}$ and

$$\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I = \dot{\mathcal{L}}(L)_I \cap \bigcap_{n\in I} (\tau^I_{E(n,I)})^{-1}(V)).$$

The proof of this theorem is given in the appendix.

Example 7. *Let $\mathcal{I}$ be a parameter structure based on N, and for $I \in \mathcal{I}$ let $\bar{E}(i,I) := I$ for $i \in N$.*

$$\bar{\mathcal{E}}_\mathcal{I} := (\bar{E}(i,I))_{(i,I)\in N\times\mathcal{I}} \text{ satisfies the assumptions of Theorem 3 for each isomorphism structure } \mathcal{B}_\mathcal{I}. \tag{4}$$

It holds $(\Theta^I)^{-1}(V) = (\tau^I_{\bar{E}(i,I)})^{-1}(V)$ for each $i \in N, I \in \mathcal{I}$, and $V \subset \Sigma^*$.
Therefore, $\mathcal{L}(L,\bar{\mathcal{E}}_\mathcal{I},V)_I = \dot{\mathcal{L}}(L)_I \cap (\Theta^I)^{-1}(V)$ for $I \in \mathcal{I}$. Especially, $\bar{S}_I = \mathcal{L}(S,\bar{\mathcal{E}}_{\mathcal{I}_1},S)_I$ for each $I \in \mathcal{I}_1$.

Example 8. *For the parameter structure $\mathcal{I}_2$, and for $\mathring{I} \times \hat{I} \in \mathcal{I}_2$ let*

$$E^2((\mathring{n},\hat{n}),\mathring{I}\times\hat{I}) := \begin{cases} \{\mathring{n}\}\times\hat{I} \mid & \mathring{n} \in \mathring{I} \\ \emptyset \mid & \mathring{n} \in \mathbb{N} \setminus \mathring{I} \end{cases} .$$

$$\mathcal{E}^2_{\mathcal{I}_2} := (E^2((\mathring{n},\hat{n}),\mathring{I}\times\hat{I}))_{((\mathring{n},\hat{n}),\mathring{I}\times\hat{I})\in(\mathbb{N}\times\mathbb{N})\times\mathcal{I}_2} \tag{5}$$

satisfies the assumptions of Theorem 3 for the isomorphism structure $\mathcal{B}^2_{\mathcal{I}_2}$.
$(\mathcal{L}(S,\mathcal{E}^2_{\mathcal{I}_2},S)_I)_{I\in\mathcal{I}_2}$ is the formalisation of the extended example (several servers) with restricted concurrency.

In order to extend Theorem 3 with respect to self-similarity, an additional assumption is necessary. This is demonstrated by the following counter-example.

Example 9. *Let $G \subset \{\mathrm{a},\mathrm{b},\mathrm{c}\}^*$ the prefix closed language, which is accepted by the automaton Fig. 3(a). Let $H \subset \{\mathrm{a},\mathrm{b},\mathrm{c}\}^*$ the prefix closed language, which is accepted by the automaton in Fig. 3(b). It holds $\emptyset \neq G \subset H$ but $(\mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_I)_{I\in\mathcal{I}_1}$ is not self-similar, e.g.,*

$$\Pi^{\{1,2,3\}}_{\{2,3\}}(\mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_{\{1,2,3\}}) \neq (\mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_{\{2,3\}}$$

because

$$\mathrm{a_1b_1a_2a_3} \in \mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_{\{1,2,3\}},$$

and hence

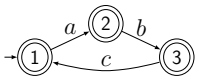
$$\mathrm{a_2a_3} \in \Pi^{\{1,2,3\}}_{\{2,3\}}(\mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_{\{1,2,3\}}),$$

but

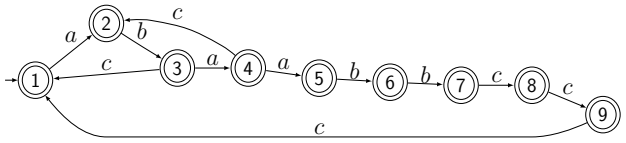
$$\mathrm{a_2a_3} \notin (\mathcal{L}(G,\bar{\mathcal{E}}_{\mathcal{I}_1},H)_{\{2,3\}}.$$

Definition 12 (closed under shuffle projection)**.** *Let $L, V \subset \Sigma^*$. V is closed under shuffle projection with respect to L, iff*

$$\Pi^{\mathbb{N}}_K[(\bigcap_{n\in\mathbb{N}} (\tau^{\mathbb{N}}_n)^{-1}(L)) \cap (\Theta^{\mathbb{N}})^{-1}(V)] \subset (\Theta^{\mathbb{N}})^{-1}(V) \tag{6}$$



(a) Automaton accepting G



(b) Automaton accepting H

Figure 3. Counterexample

for each subset $\emptyset \neq K \subset \mathbb{N}$. *We abbreviate this by* $\mathrm{SP}(L,V)$.

Remark 1. *It can be shown that in* $\mathrm{SP}(L,V)$ $\mathbb{N}$ *can be replaced by each countable infinite set.*

Remark 2. *If* L *and* V *are prefix closed with* $\emptyset \neq L \subset V$, *then it is easy to show that* $\mathrm{SP}(L,V)$ *follows from self-similarity of* $(\mathcal{L}(L,\bar{\mathcal{E}}_{\mathcal{I}_1},V)_I)_{I\in\mathcal{I}_1}$.

With Definition 12 we are now able to formulate our main result for constructing well-behaved scalable systems defined by a single synchronisation condition.

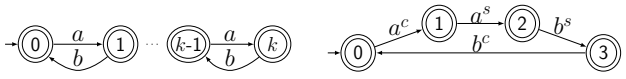
Theorem 4 (construction condition for well-behaved scalable systems)**.** *By the assumptions of Theorem 3 together with* $\mathrm{SP}(L,V)$

$$(\mathcal{L}(L,\mathcal{E}_{\mathcal{I}},V)_I)_{I\in\mathcal{I}}$$

is a well-behaved scalable system.

The proof of this theorem is given in the appendix.

Example 10. *For* $k \in \mathbb{N}$ *let the prefix closed language* $F_k \subset \{a,b\}^*$ *be defined by the automaton in Fig. 4(a).*



(a) Automaton for $F_k \subset \{a,b\}^*$ (b) One client, one server

Figure 4. Automata at different abstraction levels

With respect to Example 1, $F_1 = S$ *holds. It can be shown that* $\mathrm{SP}(S,F_k)$ *holds for each* $k \in \mathbb{N}$. *With Theorem 4 now, by* (4) *and* (5) *especially, the systems* $(\mathcal{L}(L,\bar{\mathcal{E}}_{\mathcal{I}_1},F_k)_I)_{I\in\mathcal{I}_1}$ *and* $(\mathcal{L}(L,\mathcal{E}^2_{\mathcal{I}_2},F_k)_I)_{I\in\mathcal{I}_2}$ *are uniformly monotonic parameterised and self-similar. These are the two cases of the guiding example where the concurrency of the execution of requests is bounded by* k.

Theorem 4 is the main result for constructing well-behaved scalable systems defined by a single synchronisation condition. The following section shows how this result together with the Intersection Theorem can be used for constructing more complex well-behaved scalable systems defined by the combination of several synchronisation conditions, as for example well-behaved scalable systems consisting of several component types.

VI. Well-behaved Scalable Systems Generated by a Family of Influence Structures

Up to now, the examples were considered at an abstraction level, which takes into account only the actions of the server (or the servers, depending on the choice of the parameter structure).

Example 11. *For a finer abstraction level, which additionally takes into account the actions of the clients, a finer alphabet, e.g.,* $\check{\Sigma} = \{a^c,b^c,a^s,b^s\}$ *and a prefix closed language* $\check{S} \subset \check{\Sigma}^*$ *is needed, which, e.g., is defined by the automaton in Fig. 4(b).*

In general, a finer relation for system specifications at different abstraction levels can be defined by alphabetic language homomorphisms.

Definition 13 (abstractions)**.** *In general, let* $\check{L} \subset \check{\Sigma}^*$ *and* $L \subset \Sigma^*$ *be prefix closed languages. We call* $\check{L}$ finer than L *or* L coarser than $\check{L}$ *iff an alphabetic homomorphism* $\nu : \check{\Sigma}^* \to \Sigma^*$ *exists with* $\nu(\check{L}) = L$.

For each parameter structure $\mathcal{I}$ and $I \in \mathcal{I}$ ν defines an homomorphism $\nu^I : \check{\Sigma}_I^* \to \Sigma_I^*$ by $\nu^I(a_i) := (\nu(a))_i$ for $a \in \check{\Sigma}$ and $i \in I$, where $(\varepsilon)_i := \varepsilon$.

Let now $\mathcal{E}_{\mathcal{I}}$ be an influence structure for $\mathcal{I}$ indexed by N, which is the base of $\mathcal{I}$, and let $\emptyset \neq L \subset V \subset \Sigma^*$ be prefix closed. $(\mathcal{L}(L,\mathcal{E}_{\mathcal{I}},V)_I)_{I\in\mathcal{I}}$ induces a restriction of the concurrency in $(\dot{\mathcal{L}}(\check{L})_I)_I$ by the intersections

$$\dot{\mathcal{L}}(\check{L})_I \cap (\nu^I)^{-1}[\bigcap_{t\in N}(\tau^I_{E(t,I)})^{-1}(V)] \text{ for each } I \in \mathcal{I}. \quad (7)$$

If $\check{\tau}^I_{I'} : \check{\Sigma}_I^* \to \check{\Sigma}^*$ is defined analogously to $\tau^I_{I'}$ for $I, I' \subset N$ by

$$\check{\tau}^I_{I'}(a_i) = \begin{cases} a \mid & a \in \check{\Sigma} \text{ and } i \in I \cap I' \\ \varepsilon \mid & a \in \check{\Sigma} \text{ and } i \in I \setminus I' \end{cases},$$

then holds $\tau^I_{I'} \circ \nu^I = \nu \circ \check{\tau}^I_{I'}$. From this it follows that

$$(\nu^I)^{-1}[\bigcap_{t\in N}(\tau^I_{E(t,I)})^{-1}(V)] = \bigcap_{t\in N}(\check{\tau}^I_{E(t,I)})^{-1}(\nu^{-1}(V))$$

and therewith

$$\dot{\mathcal{L}}(\check{L})_I \cap (\nu^I)^{-1}[\bigcap_{t\in N}(\tau^I_{E(t,I)})^{-1}(V)] = \mathcal{L}(\check{L},\mathcal{E}_{\mathcal{I}},\nu^{-1}(V))_I \quad (8)$$

for each $I \in \mathcal{I}$. Notice that $\emptyset \neq \check{L} \subset \nu^{-1}(V) \subset \check{\Sigma}^*$ is prefix closed. So if $(\mathcal{L}(L,\mathcal{E}_{\mathcal{I}},V)_I)_{I\in\mathcal{I}}$ fulfils the assumptions of Theorem 3, then this holds for $(\mathcal{L}(\check{L},\mathcal{E}_{\mathcal{I}},\nu^{-1}(V))_I)_{I\in\mathcal{I}}$ as well and the system

$$(\dot{\mathcal{L}}(\check{L})_I \cap (\nu^I)^{-1}[\bigcap_{t\in N}(\tau^I_{E(t,I)})^{-1}(V)])_{I\in\mathcal{I}}, \quad (9)$$

which is defined by the intersections (7), is a scalable system. The following general theorem can be used to prove self-similarity of such systems.

Theorem 5 (inverse abstraction theorem)**.** *Let $\varphi : \Sigma^* \to \Phi^*$ be an alphabetic homomorphism and $W, X \subset \Phi^*$, then*

$$\mathrm{SP}(W,X) \text{ implies } \mathrm{SP}(\varphi^{-1}(W), \varphi^{-1}(X)).$$

Proof of Theorem 5:
Let K be a non-empty set. Each alphabetic homomorphism $\varphi : \Sigma^* \to \Phi^*$ defines a homomorphism $\varphi^K : \Sigma_K^* \to \Phi_K^*$ by

$$\varphi^K(a_n) := (\varphi(a))_n \text{ for } a_n \in \Sigma_K, \text{ where } (\varepsilon)_n = \varepsilon.$$

If $\bar{\tau}_n^K : \Phi_K^* \to \Phi$ and $\bar{\Theta}^K : \Phi_K^* \to \Phi$ are defined analogous to τ_n^K and Θ^K, then

$$\varphi \circ \tau_n^K = \bar{\tau}_n^K \circ \varphi^K, \text{ and } \varphi \circ \Theta^K = \bar{\Theta}^K \circ \varphi^K. \tag{10}$$

Let now N be an infinite countable set. Because of (10), for $W, X \subset \Phi^*$

$$\begin{aligned}&(\bigcap_{n\in N} (\tau_n^N)^{-1}(\varphi^{-1}(W))) \cap (\Theta^N)^{-1}(\varphi^{-1}(X))\\ &= (\varphi^N)^{-1}[(\bigcap_{n\in N} (\bar{\tau}_n^N)^{-1}(W)) \cap (\bar{\Theta}^N)^{-1}(X)].\end{aligned} \tag{11}$$

Because of $\varphi^K(w) = \varphi^N(w)$ for $w \in \Sigma_K^* \subset \Sigma_N^*$ and $\emptyset \neq K \subset N$

$$(\varphi^K)^{-1}(Z) \subset (\varphi^N)^{-1}(Z) \text{ for } Z \subset \Phi_K^*. \tag{12}$$

If now $\mathrm{SP}(W,X)$, and

$$\Pi_K^N[(\varphi^N)^{-1}(Y)] = (\varphi^K)^{-1}(\bar{\Pi}_K^N[Y]) \tag{13}$$

for $Y \subset \Phi_N^*$ and $\emptyset \neq K \subset N$, where $\bar{\Pi}_K^N : \Phi_N^* \to \Phi_K^*$ is defined analogous to Π_K^N, then follows (with (10) - (13))

$$\begin{aligned}&\Pi_K^N[(\bigcap_{n\in N} (\tau_n^N)^{-1}(\varphi^{-1}(W))) \cap (\Theta^N)^{-1}(\varphi^{-1}(X))]\\ &= (\varphi^K)^{-1}(\bar{\Pi}_K^N[(\bigcap_{n\in N} (\bar{\tau}_n^N)^{-1}(W)) \cap (\bar{\Theta}^N)^{-1}(X)])\\ &\subset (\varphi^K)^{-1}((\bar{\Theta}^N)^{-1}(X)) \subset (\varphi^N)^{-1}((\bar{\Theta}^N)^{-1}(X))\\ &= (\Theta^N)^{-1}(\varphi^{-1}(X)).\end{aligned} \tag{14}$$

With (14)

$$\mathrm{SP}(\varphi^{-1}(W), \varphi^{-1}(X)) \text{ follows from } \mathrm{SP}(W,X), \tag{15}$$

if (13) holds.

It remains to show (13). For the proof of (13) it is sufficient to prove

$$\Pi_K^N((\varphi^N)^{-1}(y)) = (\varphi^K)^{-1}(\bar{\Pi}_K^N(y)) \tag{16}$$

for each $y \in \Phi_N^*$, because of

$$\Pi_K^N((\varphi^N)^{-1}(Y) = \bigcup_{y\in Y} \Pi_K^N((\varphi^N)^{-1}(y))$$

and

$$(\varphi^K)^{-1}(\bar{\Pi}_K^N(Y)) = \bigcup_{y\in Y} (\varphi^K)^{-1}(\bar{\Pi}_K^N(y)).$$

Here, for $f : A \to B$ and $b \in B$ we use the convention

$$f^{-1}(b) = f^{-1}(\{b\}).$$

With $Y = \{y\}$ (16) is also necessary for (13), and so it is equivalent to (13).

Definition 14 ((general) projection)**.** *For arbitrary alphabets Δ and Δ' with $\Delta' \subset \Delta$ general projections $\pi_{\Delta'}^{\Delta} : \Delta^* \to \Delta'^*$ are defined by*

$$\pi_{\Delta'}^{\Delta}(a) := \begin{cases} a \mid & a \in \Delta' \\ \varepsilon \mid & a \in \Delta \setminus \Delta' \end{cases}. \tag{17}$$

In this terminology the projections

$$\Pi_K^N : \Sigma_N^* \to \Sigma_K^* \text{ and } \bar{\Pi}_K^N : \Phi_N^* \to \Phi_K^*$$

considered until now are special cases, which we call *parameter-projections*. It holds

$$\Pi_K^N = \pi_{\Sigma_K}^{\Sigma_N} \text{ and } \bar{\Pi}_K^N = \pi_{\Phi_K}^{\Phi_N}. \tag{18}$$

Because of the different notations, in general we just use the term *projection* for both cases.

We now consider the equation (16) for the special case, where $\varphi : \Sigma^* \to \Phi^*$ is a projection, that is $\varphi = \pi_{\Phi}^{\Sigma}$ with $\Phi \subset \Sigma$. In this case also $\varphi^N : \Sigma_N^* \to \Phi_N^*$ is a projection, with

$$\varphi^N = \pi_{\Phi_N}^{\Sigma_N}. \tag{19}$$

Lemma 1 (projection-lemma)**.**
Let Δ be an alphabet, $\Delta' \subset \Delta$, $\Gamma \subset \Delta$ and $\Gamma' = \Delta' \cap \Gamma$, then

$$\pi_{\Delta'}^{\Delta}((\pi_{\Gamma}^{\Delta})^{-1}(y)) = (\pi_{\Gamma'}^{\Delta'})^{-1}(\pi_{\Delta'}^{\Delta}(y))$$

for each $y \in \Gamma^*$.

Proof: Let $y \in \Gamma^*$. We show

$$\pi_{\Gamma'}^{\Delta'}(\pi_{\Delta'}^{\Delta}(z)) = \pi_{\Delta'}^{\Delta}(y) \text{ for each } z \in (\pi_{\Gamma}^{\Delta})^{-1}(y) \tag{20}$$

and we show that

$$\begin{aligned}&\text{for each } u \in (\pi_{\Gamma'}^{\Delta'})^{-1}(\pi_{\Delta'}^{\Delta}(y)) \text{ there exists a}\\ &v \in (\pi_{\Gamma}^{\Delta})^{-1}(y) \text{ such that } \pi_{\Delta'}^{\Delta}(v) = u.\end{aligned} \tag{21}$$

From (20) it follows that

$$\pi_{\Delta'}^{\Delta}((\pi_{\Gamma}^{\Delta})^{-1}(y)) \subset (\pi_{\Gamma'}^{\Delta'})^{-1}(\pi_{\Delta'}^{\Delta}(y))$$

and from (21) it follows that

$$(\pi_{\Gamma'}^{\Delta'})^{-1}(\pi_{\Delta'}^{\Delta}(y)) \subset \pi_{\Delta'}^{\Delta}((\pi_{\Gamma}^{\Delta})^{-1}(y)),$$

which in turn proves Lemma 1.

Proof of (20): By definition of π_{Γ}^{Δ}, $\pi_{\Gamma'}^{\Delta'}$ and $\pi_{\Delta'}^{\Delta}$ follows

$$\pi_{\Gamma'}^{\Delta'}(\pi_{\Delta'}^{\Delta}(z)) = \pi_{\Delta'}^{\Delta}(\pi_{\Gamma}^{\Delta}(z))$$

for each $z \in \Delta^*$ and therewith (20).

Proof of (21) by induction on $y \in \Gamma^*$:

Induction base. Let $y = \varepsilon$, then $u \in (\Delta' \setminus \Gamma')^*$ for each $u \in (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(y))$. From this follows

$$\pi^{\Delta}_{\Delta'}(v) = u \text{ with } v := u \in (\pi^{\Delta}_{\Gamma})^{-1}(\varepsilon).$$

Induction step. Let $y = \mathring{y}\hat{y}$ with $\mathring{y} \in \Gamma^*$ and $\hat{y} \in \Gamma$.
Case 1: $\hat{y} \in \Gamma \setminus \Gamma' = \Gamma \cap (\Delta \setminus \Delta')$
Then

$$(\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(y)) = (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(\mathring{y})).$$

By induction hypothesis then for each $u \in (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(y))$ it exists $\mathring{v} \in (\pi^{\Delta}_{\Gamma})^{-1}(\mathring{y})$ such that $\pi^{\Delta}_{\Delta'}(\mathring{v}) = u$.
With $v := \mathring{v}\hat{y}$ holds $\pi^{\Delta}_{\Gamma}(\mathring{v}\hat{y}) = \mathring{y}\hat{y} = y$ and hence

$$v \in (\pi^{\Delta}_{\Gamma})^{-1}(y) \text{ and } \pi^{\Delta}_{\Delta'}(v) = \pi^{\Delta}_{\Delta'}(\mathring{v}) = u.$$

Case 2: $\hat{y} \in \Gamma' \subset \Delta'$
Then $\pi^{\Delta}_{\Delta'}(y) = \pi^{\Delta}_{\Delta'}(\mathring{y})\hat{y}$. Therefore, each $u \in (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(y))$ can be departed into $u = \mathring{u}\hat{y}\hat{u}$ with $\mathring{u} \in (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Delta}_{\Delta'}(\mathring{y}))$ and $\hat{u} \in (\Delta' \setminus \Gamma')^*$.
By induction hypothesis then exists $\mathring{v} \in (\pi^{\Delta}_{\Gamma})^{-1}(\mathring{y})$ such that $\pi^{\Delta}_{\Delta'}(\mathring{v}) = \mathring{u}$.
With $v := \mathring{v}\hat{y}\hat{u}$ holds $\pi^{\Delta}_{\Gamma}(\mathring{v}\hat{y}\hat{u}) = \mathring{y}\hat{y} = y$ and hence

$$v \in (\pi^{\Delta}_{\Gamma})^{-1}(y) \text{ and } \pi^{\Delta}_{\Delta'}(v) = \pi^{\Delta}_{\Delta'}(\mathring{v})\hat{y}\hat{u} = \mathring{u}\hat{y}\hat{u} = u.$$

This completes the proof of (21). ■

For $y \in \Gamma^*$ holds

$$\pi^{\Delta}_{\Delta'}(y) = \pi^{\Gamma}_{\Delta' \cap \Gamma}(y) = \pi^{\Gamma}_{\Gamma'}(y).$$

Therewith, from Lemma 1 follows

$$\pi^{\Delta}_{\Delta'}((\pi^{\Delta}_{\Gamma})^{-1}(y)) = (\pi^{\Delta'}_{\Gamma'})^{-1}(\pi^{\Gamma}_{\Gamma'}(y)) \text{ for each } y \in \Gamma^*. \tag{22}$$

For $\emptyset \neq K \subset N, \Phi \subset \Sigma, \Delta := \Sigma_N, \Delta' := \Sigma_K$, and $\Gamma := \Phi_N$ holds $\Gamma' = \Delta' \cap \Gamma = \Phi_K$.

Assuming $\varphi = \pi^{\Sigma}_{\Phi}$, which implies $\varphi^K = \pi^{\Sigma_K}_{\Phi_K}$, then from (22) (with (18) and (19)), follows

$$\Pi^N_K((\varphi^N)^{-1}(y)) = (\varphi^K)^{-1}(\bar{\Pi}^N_K(y))$$

for $y \in \Phi^*_N$, and so (16). With this,

premise (13) is fulfilled for (15), when φ is a projection, (23)

which proves Theorem 5 for projections.

Definition 15 (strictly alphabetic homomorphism)**.** *Let Σ, Φ alphabets, and $\varphi : \Sigma^* \to \Phi^*$ a homomorphism. Then φ is called* alphabetic*, if $\varphi(\Sigma) \subset \Phi \cup \{\varepsilon\}$, and φ is called* strictly alphabetic*, if $\varphi(\Sigma) \subset \Phi$.*

Each alphabetic homomorphism $\varphi : \Sigma^* \to \Phi^*$ is the composition of a projection with a strictly alphabetic homomorphism, more precisely,

$$\varphi = \varphi_S \circ \pi^{\Sigma}_{\varphi^{-1}(\Phi) \cap \Sigma}, \tag{24}$$

where $\varphi_S : (\varphi^{-1}(\Phi) \cap \Sigma)^* \to \Phi^*$ is the strictly alphabetic homomorphism defined by

$$\varphi_S(a) := \varphi(a) \text{ for } a \in \varphi^{-1}(\Phi) \cap \Sigma.$$

For $W, X \subset \Phi^*$ and $\varphi : \Sigma^* \to \Phi^*$ alphabetic (24) implies

$$\begin{aligned} \varphi^{-1}(W) =& (\pi^{\Sigma}_{\varphi^{-1}(\Phi) \cap \Sigma})^{-1}((\varphi_S)^{-1}(W)) \text{ and} \\ \varphi^{-1}(X) =& (\pi^{\Sigma}_{\varphi^{-1}(\Phi) \cap \Sigma})^{-1}((\varphi_S)^{-1}(X)). \end{aligned} \tag{25}$$

Now with (23) and (25) it remains to prove Theorem 5 for strictly alphabetic homomorphisms. This will be done by Lemma 2, which proves (16) for strictly alphabetic homomorphisms.

Lemma 2. *Let $\varphi : \Sigma^* \to \Phi^*$ be a strictly alphabetic homomorphism, then for all $y \in \Phi^*_N$ and $\emptyset \neq K \subset N$ holds*

$$\Pi^N_K((\varphi^N)^{-1}(y)) = (\varphi^K)^{-1}(\bar{\Pi}^N_K(y)).$$

Proof: Proof by induction on y.
Induction basis: $y = \varepsilon$
Because φ^N is strictly alphabetic

$$(\varphi^N)^{-1}(\varepsilon) = \{\varepsilon\} \text{ and so } \Pi^N_K((\varphi^N)^{-1}(\varepsilon)) = \{\varepsilon\}.$$

For the same reason

$$(\varphi^K)^{-1}(\bar{\Pi}^N_K(\varepsilon)) = (\varphi^K)^{-1}(\varepsilon) = \{\varepsilon\}.$$

Induction step: Let $y = y'a_t$ with $a_t \in \Phi_N$, where $a \in \Phi$ and $t \in N$. Because φ^N is alphabetic, it holds

$$(\varphi^N)^{-1}(y'a_t) = ((\varphi^N)^{-1}(y'))((\varphi^N)^{-1}(a_t)),$$

and so

$$\Pi^N_K((\varphi^N)^{-1}(y'a_t)) = \Pi^N_K((\varphi^N)^{-1}(y'))\Pi^N_K((\varphi^N)^{-1}(a_t)).$$

Also holds

$$(\varphi^K)^{-1}(\bar{\Pi}^N_K(y'a_t)) = (\varphi^K)^{-1}(\bar{\Pi}^N_K(y'))(\varphi^K)^{-1}(\bar{\Pi}^N_K(a_t)).$$

According to the induction hypothesis, it holds

$$\Pi^N_K((\varphi^N)^{-1}(y')) = (\varphi^K)^{-1}(\bar{\Pi}^N_K(y')).$$

Therefore, it remains to show

$$\Pi^N_K((\varphi^N)^{-1}(a_t)) = (\varphi^K)^{-1}(\bar{\Pi}^N_K(a_t)).$$

Case 1: $t \notin K$
Because φ^N is strictly alphabetic, it holds $(\varphi^N)^{-1}(a_t) \subset \Sigma_{\{t\}}$, so

$$\Pi^N_K((\varphi^N)^{-1}(a_t)) = \{\varepsilon\}.$$

Additionally holds $\bar{\Pi}^N_K(a_t) = \varepsilon$, and therewith

$$(\varphi^K)^{-1}(\bar{\Pi}^N_K(a_t)) = \{\varepsilon\},$$

because φ^K is strictly alphabetic.

Case 2: $t \in K$
Because φ^N is strictly alphabetic, it holds

$$(\varphi^N)^{-1}(a_t) = \{b_t \in \Sigma_{\{t\}} | \varphi(b) = a\},$$

and therewith

$$\Pi_K^N((\varphi^N)^{-1}(a_t)) = \{b_t \in \Sigma_{\{t\}} | \varphi(b) = a\}.$$

$\bar{\Pi}_K^N(a_t) = a_t$ and therewith

$$(\varphi^K)^{-1}(\bar{\Pi}_K^N(a_t)) = \{b_t \in \Sigma_{\{t\}} | \varphi(b) = a\},$$

because φ^K is strictly alphabetic. This completes the proof of Lemma 2. ∎

This completes the proof of Theorem 5. ∎

Generally, by (6), $\mathrm{SP}(\nu^{-1}(L), \nu^{-1}(V))$ implies $\mathrm{SP}(X, \nu^{-1}(V))$ for each $X \subset \nu^{-1}(L)$. Especially $\mathrm{SP}(\check{L}, \nu^{-1}(V))$ is implied by $\mathrm{SP}(L, V)$ on account of Theorem 5. So, by Theorem 5, if $(\mathcal{L}(L, \mathcal{E}_\mathcal{I}, V)_I)_{I \in \mathcal{I}}$ fulfils the assumptions of Theorem 4, then

$$(\mathcal{L}(\check{L}, \mathcal{E}_\mathcal{I}, \nu^{-1}(V))_I)_{I \in \mathcal{I}} = (\dot{\mathcal{L}}(\check{L})_I \cap (\nu^I)^{-1}[\bigcap_{t \in N} (\tau^I_{E(t,I)})^{-1}(V)])_{I \in \mathcal{I}} \quad (26)$$

is a well-behaved scalable system.

The intersections in (7) formalise restriction of concurrency in $(\dot{\mathcal{L}}(\check{L})_I)_{I \in \mathcal{I}}$ under *one specific aspect* (one specific synchronisation condition), which is given by ν, $\mathcal{E}_\mathcal{I}$, and V. Restriction of concurrency under *several aspects* (several synchronisation conditions) is formalised by the intersections

$$\dot{\mathcal{L}}(\check{L})_I \cap \bigcap_{r \in R} (\nu_r^I)^{-1}[\bigcap_{t \in N} (\tau^I_{E_r(t,I)})^{-1}(V_r)] \quad (27)$$

for each $I \in \mathcal{I}$ based on N, $R \neq \emptyset$ is the index set of the aspects. The family of aspects restricting concurrency is given by

- a family $(\nu_r)_{r \in R}$ of *alphabetic homomorphisms* $\nu_r : \check{\Sigma}^* \to \Sigma^{(r)*}$ for $r \in R$,
- a family $(\mathcal{E}^r_\mathcal{I})_{r \in R}$ of *influence structures* $\mathcal{E}^r_\mathcal{I} = (E_r(t,I))_{(t,I) \in N \times \mathcal{I}}$ indexed by N for $r \in R$, and
- a family $(V_r)_{r \in R}$ of *influence behaviours* $V_r \subset \Sigma^{(r)*}$ for $r \in R$.

From (8) it follows now

$$\dot{\mathcal{L}}(\check{L})_I \cap \bigcap_{r \in R} (\nu_r^I)^{-1}[\bigcap_{t \in N} (\tau^I_{E_r(t,I)})^{-1}(V_r)] = \bigcap_{r \in R} \mathcal{L}(\check{L}, \mathcal{E}^r_\mathcal{I}, \nu_r^{-1}(V_r))_I$$

for each $I \in \mathcal{I}$. Because of the intersection theorem, the uniform monotonic parameterisation and self-similarity of the system

$$(\dot{\mathcal{L}}(\check{L})_I \cap \bigcap_{r \in R} (\nu_r^I)^{-1}[\bigcap_{t \in N} (\tau^I_{E_r(t,I)})^{-1}(V_r)])_{I \in \mathcal{I}}$$

can be inferred from respective properties of the systems

$$(\mathcal{L}(\check{L}, \mathcal{E}^r_\mathcal{I}, \nu_r^{-1}(V_r))_I)_{I \in \mathcal{I}} \text{ for each } r \in R.$$

Using (9) and (26), this requires the verification of the assumptions of Theorem 4 for

$$(\mathcal{L}(\nu_r(\check{L}), \mathcal{E}^r_\mathcal{I}, V_r)_I)_{I \in \mathcal{I}} \text{ for each } r \in R. \quad (28)$$

If $\mathcal{I}$ is based on $N = \bigtimes_{k \in K} N_k$, where K is a finite set and each N_k is countable, then along the lines of $\mathcal{I}_2$, a parameter structure $\mathcal{I}_K$ can be defined for this domain. Such $\mathcal{I}_K$ fit for systems consisting of finitely many component types. Each subset $K' \subset K$ with $\emptyset \neq K' \neq K$ defines a bijection between N and $(\bigtimes_{k \in K'} N_k) \times (\bigtimes_{k \in K \setminus K'} N_k)$. By this bijection, for each of these K' an influence structure $\mathcal{E}^{K'}_{\mathcal{I}_K}$ is defined like $\mathcal{E}^2_{\mathcal{I}_2}$ that satisfies the assumptions of Theorem 3 with respect to an isomorphism structure $\mathcal{B}^K_{\mathcal{I}_K}$ defined like $\mathcal{E}^2_{\mathcal{I}_2}$.

VII. Scalable Safety Properties

We will now give an example that demonstrates the significance of self-similarity for verification purposes and then present a generic verification scheme for scalable safety properties.

Example 12. *We consider a system of servers, each of them managing a resource, and clients, which want to use these resources. We assume that as a means to enforce a given* privacy policy *a server has to manage its resource in such a way that no client may access this resource during it is in use by another client (*privacy requirement*). This may be required to ensure* anonymity *in such a way that clients and their actions on a resource cannot be linked by an observer.*

We formalise this system at an abstract level, where a client may perform the actions $\mathrm{a^c}$ (send a request), $\mathrm{b^c}$ (receive a permission) and $\mathrm{c^c}$ (send a free-message), and a server may perform the corresponding actions $\mathrm{a^s}$ (receive a request), $\mathrm{b^s}$ (send a permission) and $\mathrm{c^s}$ (receive a free-message). The automaton $\mathbb{L}$ depicted in Fig. 5 describes the cooperation of one client and one server.

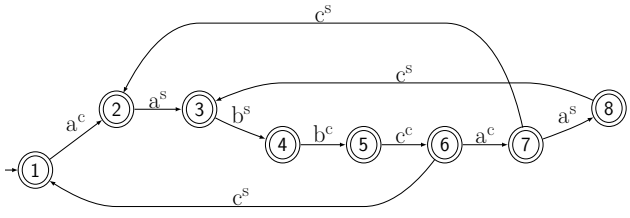


Figure 5. Automaton $\mathbb{L}$

We now formalise the parameterised cooperation $(\mathcal{C}_J)_{J \in \mathcal{I}}$ according to the description in Section VI.

$$\mathcal{C}_J = \dot{\mathcal{L}}(\check{L})_J \cap \bigcap_{r \in R} (\nu_r^J)^{-1}[\bigcap_{t \in N} (\tau^J_{E_r(t,J)})^{-1}(V_r)].$$

Because $(\mathcal{C}_J)_{J\in\mathcal{I}}$ involves several clients as well as several servers, let $\mathcal{I} := \mathcal{I}_2$, $N := \mathbb{N}\times\mathbb{N}$, and $\mathcal{B}_{\mathcal{I}} := \mathcal{B}^2_{\mathcal{I}_2}$, where the first component refers to the client and the second component refers to the server. Now $\check{L}$ is the prefix closed language that is accepted by the automaton $\mathbb{L}$.

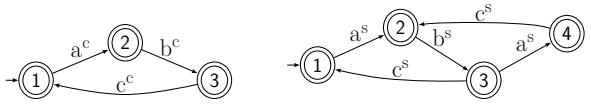
For the examined example we assume that both clients and servers are subject to constaints with respect to processing several cooperations. Thus, two aspects of constaints are considered, therefore: $R := \{c,s\}$, $\Sigma^{(c)} := \{\mathrm{a}^{\mathrm{c}},\mathrm{b}^{\mathrm{c}},\mathrm{c}^{\mathrm{c}}\}$, $\Sigma^{(s)} := \{\mathrm{a}^{\mathrm{s}},\mathrm{b}^{\mathrm{s}},\mathrm{c}^{\mathrm{s}}\}$, $\check{\Sigma} = \Sigma^{(c)}\dot\cup\Sigma^{(s)}$, $\nu_c : \check{\Sigma}^* \to \Sigma^{(c)*}$ with

$$\nu_c(x) := \begin{cases} x \mid & x \in \Sigma^{(c)} \\ \varepsilon \mid & x \in \Sigma^{(s)} \end{cases},$$

and $\nu_s : \check{\Sigma}^* \to \Sigma^{(s)*}$ with

$$\nu_s(x) := \begin{cases} x \mid & x \in \Sigma^{(s)} \\ \varepsilon \mid & x \in \Sigma^{(c)} \end{cases}.$$

$\nu_c(\check{L})$ and $\nu_s(\check{L})$ now describe the behaviour of a client respectively a server in the cooperation of a client with a server. $\nu_c(\check{L})$ and $\nu_s(\check{L})$ are accepted by the automata in Fig. 6(a) and Fig. 6(b).



(a) Automaton accepting $\nu_c(\check{L})$ (b) Automaton accepting $\nu_s(\check{L})$

Figure 6. Client and server behaviour in the cooperation

These automata show that in $\nu_c(\check{L})$ the "phase" $\mathrm{a}^{\mathrm{c}}\mathrm{b}^{\mathrm{c}}\mathrm{c}^{\mathrm{c}}$ can happen repeatedly and in $\nu_s(\check{L})$ two instances of the "phase" $\mathrm{a}^{\mathrm{s}}\mathrm{b}^{\mathrm{s}}\mathrm{c}^{\mathrm{s}}$ can run partly concurrently.

We now assume that this restriction of concurrency shall also hold for the parameterised system. This restriction is then given by the definitions $V_c := \nu_c(\check{L})$ and $V_s := \nu_s(\check{L})$ with an appropriate choice of influence structures.

Because for each client respectively server all cooperations with all servers respectively clients influence each other, let now according to Example 8, for $I\times K \in \mathcal{I}_2$ and $(i,k) \in \mathbb{N}\times\mathbb{N}$:

$$E^c((i,k),I\times K) := \begin{cases} \{i\}\times K \mid & i \in I \\ \emptyset \mid & i \in \mathbb{N}\setminus I \end{cases},$$

$$E^s((i,k),I\times K) := \begin{cases} I\times\{k\} \mid & k \in K \\ \emptyset \mid & k \in \mathbb{N}\setminus K \end{cases},$$

$$\mathcal{E}^c_{\mathcal{I}_2} := (E^c((i,k),I\times K))_{((i,k),I\times K)\in(\mathbb{N}\times\mathbb{N})\times\mathcal{I}_2}, \text{ and}$$

$$\mathcal{E}^s_{\mathcal{I}_2} := (E^s((i,k),I\times K))_{((i,k),I\times K)\in(\mathbb{N}\times\mathbb{N})\times\mathcal{I}_2}.$$

As in Example 8, both influence structures satisfy the assumptions of Theorem 3 for the isomorphism structure $\mathcal{B}^2_{\mathcal{I}_2}$. Therefore, $(\mathcal{L}(\nu_c(\check{L}),\mathcal{E}^c_{\mathcal{I}_2},\nu_c(\check{L}))_J)_{J\in\mathcal{I}_2}$ and $(\mathcal{L}(\nu_s(\check{L}),\mathcal{E}^s_{\mathcal{I}_2},\nu_s(\check{L}))_J)_{J\in\mathcal{I}_2}$ are scalable systems. Because of (28) now $(\mathcal{C}_J)_{J\in\mathcal{I}_2}$ is a well-behaved scalable system if $\mathrm{SP}(\nu_c(\check{L}),\nu_c(\check{L}))$ and $\mathrm{SP}(\nu_s(\check{L}),\nu_s(\check{L}))$ hold.

In [24], sufficient conditions are given for a property equivalent to $\mathrm{SP}(U,V)$. These can be proven for both examples. A comprehensive and more general method for verification of $\mathrm{SP}(U,V)$ is subject of a forthcoming paper.

Considering b^{c} as the begin action and c^{c} as the end action with respect to accessing a resource, the privacy requirement for each $\mathcal{C}_J$ with $J = I\times K \in \mathcal{I}_2$ can be formalised by the following condition (29).

Let $i,i' \in I$, $i \neq i'$, $k \in K$ and

$$\mu^{I\times K}_{<i,i',k>} : \Sigma^*_{I\times K} \to \{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)},\mathrm{b}^{\mathrm{c}}{}_{(i',k)}\}^* \text{ with}$$

$$\mu^{I\times K}_{<i,i',k>}(x) := \begin{cases} x \mid x \in \{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)},\mathrm{b}^{\mathrm{c}}{}_{(i',k)}\} \\ \varepsilon \mid x \in \Sigma_{I\times K}\setminus\{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)},\mathrm{b}^{\mathrm{c}}{}_{(i',k)}\}. \end{cases}$$

Condition: For each $i,i' \in I$, $i \neq i'$ and $k \in K$ holds

$$\mu^{I\times K}_{<i,i',k>}(\mathcal{C}_{I\times K}) \cap \Sigma^*_{\{i,i'\}\times\{k\}}\mathrm{b}^{\mathrm{c}}{}_{(i,k)}\mathrm{b}^{\mathrm{c}}{}_{(i',k)} = \emptyset. \quad (29)$$

For $i,i' \in I$, $i \neq i'$, and $k \in K$ let

$$\rho_{<i,i',k>} : \Sigma^*_{\{i,i'\}\times\{k\}} \to \{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)},\mathrm{b}^{\mathrm{c}}{}_{(i',k)}\}^*$$

be defined by

$$\rho_{<i,i',k>}(x) := \begin{cases} x \mid x \in \{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)},\mathrm{b}^{\mathrm{c}}{}_{(i',k)}\} \\ \varepsilon \mid x \in \Sigma_{\{i,i'\}\times\{k\}}\setminus\{\mathrm{b}^{\mathrm{c}}{}_{(i,k)},\mathrm{c}^{\mathrm{c}}{}_{(i,k)}, \\ \qquad\qquad \mathrm{b}^{\mathrm{c}}{}_{(i',k)}\} \end{cases},$$

then

$$\mu^{I\times K}_{<i,i',k>} = \rho_{<i,i',k>} \circ \Pi^{I\times K}_{\{i,i'\}\times\{k\}}.$$

Hence,

$$\mu^{I\times K}_{<i,i',k>}(\mathcal{C}_{I\times K}) = \rho_{<i,i',k>}(\mathcal{C}_{\{i,i'\}\times\{k\}}) \quad (30)$$

because $(\mathcal{C}_{I\times K})_{I\times K\in\mathcal{I}_2}$ is a well-behaved scalable system.

Let

$$\iota_{<i,i',k>} : \Sigma^*_{\{i,i'\}\times\{k\}} \to \Sigma^*_{\{1,2\}\times\{1\}}$$

be the isomorphism defined by

$$\iota_{<i,i',k>}(x) := \begin{cases} (\tau^{\{1\}\times\{1\}}_{(1,1)})^{-1}(\tau^{\{i\}\times\{k\}}_{(i,k)}(x)) \mid x \in \\ \qquad\qquad \Sigma_{\{i\}\times\{k\}} \\ (\tau^{\{2\}\times\{1\}}_{(2,1)})^{-1}(\tau^{\{i'\}\times\{k\}}_{(i',k)}(x)) \mid x \in \\ \qquad\qquad \Sigma_{\{i'\}\times\{k\}} \end{cases}.$$

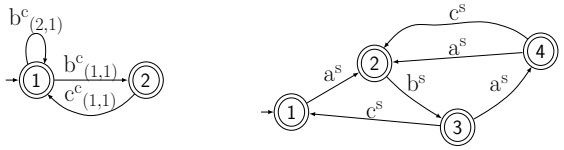
Then

$$\iota_{<i,i',k>} \in \{\iota^{\{i,i'\}\times\{k\}}_{\{1,2\}\times\{k\}} | \iota \in \mathcal{B}^2(\{i,i'\}\times\{k\},\{1,2\}\times\{1\})\}$$

(cf. Example 4), and therefore

$$\iota_{<i,i',k>}(\mathcal{C}_{\{i,i'\}\times\{k\}}) = \mathcal{C}_{\{1,2\}\times\{1\}}, \quad (31)$$

because $(\mathcal{C}_{I\times K})_{I\times K\in\mathcal{I}_2}$ is a scalable system.



(a) Minimal automaton of $\rho_{<1,2,1>}(\mathcal{C}_{\{1,2\}\times\{1\}})$ (b) Automaton accepting $\nu'_s(\check{L})$

Figure 8. Minimal automaton and counter example

Now, by (30), (31), and

$$\rho_{<i,i',k>} = \iota^{-1}_{<i,i',k>} \circ \rho_{<1,2,1>} \circ \iota_{<i,i',k>},$$

$\mathcal{C}_{I\times K}$ fulfils the privacy requirement (29) for each $I \times K \subset \mathcal{I}_2$ iff

$$\rho_{<1,2,1>}(\mathcal{C}_{\{1,2\}\times\{1\}}) \cap \Sigma^*_{\{1,2\}\times\{1\}} \mathrm{b^c}_{(1,1)} \mathrm{b^c}_{(2,1)} = \emptyset. \tag{32}$$

This can be verified by checking the automaton of $\mathcal{C}_{\{1,2\}\times\{1\}}$ that consists of 36 states (see Fig. 7). The actions of interest with regard to the privacy requirement, namely $\mathrm{b^c}$ and $\mathrm{c^c}$, are depicted by solid lines. For example, after the begin action $\mathrm{b^c}_{(1,1)}$ connecting states $7 \to 11$ a respective end action $\mathrm{c^c}_{(1,1)}$ is either directly possible (see $11 \to 15$) or after an intermediate action (see $11 \to 16$) or two intermediate actions (see $11 \to 16 \to 23$).

The minimal automaton of $\rho_{<1,2,1>}(\mathcal{C}_{\{1,2\}\times\{1\}})$ is shown in Fig. 8(a), which implies (32).

On the contrary, let $\mathcal{C}'_{I\times K}$ be defined as $\mathcal{C}_{I\times K}$ but with V'_s instead of V_s, where V'_s is defined by the automaton of Fig. 8(b). Then $(\mathcal{C}'_{I\times K})_{I\times K\in\mathcal{I}_2}$ is not self-similar because

$$\mathrm{a^c}_{(1,1)}\mathrm{a^c}_{(2,1)}\mathrm{a^c}_{(3,1)}\mathrm{a^s}_{(1,1)}\mathrm{b^s}_{(1,1)}\mathrm{a^s}_{(2,1)}\mathrm{a^s}_{(3,1)}\mathrm{b^s}_{(2,1)}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)} \in \mathcal{C}'_{\{1,2,3\}\times\{1\}},$$ and so

$$\mathrm{a^c}_{(1,1)}\mathrm{a^c}_{(2,1)}\mathrm{a^s}_{(1,1)}\mathrm{b^s}_{(1,1)}\mathrm{a^s}_{(2,1)}\mathrm{b^s}_{(2,1)}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)} \in \Pi^{\{1,2,3\}\{1\}}_{\{1,2\}\{1\}}(\mathcal{C}'_{\{1,2,3\}\times\{1\}})$$

but

$$\mathrm{a^c}_{(1,1)}\mathrm{a^c}_{(2,1)}\mathrm{a^s}_{(1,1)}\mathrm{b^s}_{(1,1)}\mathrm{a^s}_{(2,1)}\mathrm{b^s}_{(2,1)}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)} \notin \mathcal{C}'_{\{1,2\}\times\{1\}}.$$

The same action sequence shows that $\mathcal{C}'_{\{1,2,3\}\times\{1\}}$ does not fulfil the privacy requirement.

The privacy requirement of the example is a typical safety property [25]. These properties describe that "nothing forbidden happens". They can be formalised by a set $\mathcal{F}$ of forbidden action sequences. So a system $\mathcal{L}_J \subset \Sigma^*_J$ satisfies a safety property $\mathcal{F}_J \subset \Sigma^*_J$ iff $\mathcal{L}_J \cap \mathcal{F}_J = \emptyset$.

In our example, the privacy requirement (29) is formalised by

$$\begin{aligned}\mathcal{F}^p_{I\times K} &= \bigcup_{\substack{i,i'\in I, i\neq i'\\ k\in K}} (\mu^{I\times K}_{<i,i',k>})^{-1}(\Sigma^*_{\{i,i'\}\times\{k\}}\mathrm{b^c}_{(i,k)}\mathrm{b^c}_{(i',k)}) \\ &= \bigcup_{\substack{i,i'\in I, i\neq i'\\ k\in K}} (\Pi^{I\times K}_{\{i,i'\}\times\{k\}})^{-1}(\iota^{-1}_{<i,i',k>}[\rho^{-1}_{<1,2,1>} \\ &\qquad (\Sigma^*_{\{1,2\}\times\{1\}}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)})])\end{aligned}$$

because of

$$\mu^{I\times K}_{<i,i',k>} = \iota^{-1}_{<i,i',k>} \circ \rho_{<1,2,1>} \circ \iota_{<i,i',k>} \circ \Pi^{I\times K}_{\{i,i'\}\times\{k\}}$$

and

$$\begin{aligned}&\iota_{<i,i',k>}(\Sigma^*_{\{i,i'\}\times\{k\}}\mathrm{b^c}_{(i,k)}\mathrm{b^c}_{(i',k)}) \\ &\qquad = \Sigma^*_{\{1,2\}\times\{1\}}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)}.\end{aligned}$$

As

$$\begin{aligned}&\{(\{i,i'\}\times\{k\}, \iota^{-1}_{<i,i',k>}) \mid i,i'\in I, i\neq i', \text{ and } k\in K\} \\ &= \{(I'\times K', \iota^{\{1,2\}\times\{1\}}_{I'\times K'}) \mid I'\times K' \subset I\times K \text{ and} \\ &\qquad \iota \in \mathcal{B}^2(\{1,2\}\times\{1\}, I'\times K')\}\end{aligned}$$

it follows

$$\mathcal{F}^p_{I\times K} = \bigcup_{\substack{I'\times K'\subset I\times K\\ \iota\in\mathcal{B}^2(\{1,2\}\times\{1\}, I'\times K')}} (\Pi^{I\times K}_{I'\times K'})^{-1}(\iota^{\{1,2\}\times\{1\}}_{I'\times K'}(F^p)) \tag{33}$$

with

$$F^p := \rho^{-1}_{<1,2,1>}(\Sigma^*_{\{1,2\}\times\{1\}}\mathrm{b^c}_{(1,1)}\mathrm{b^c}_{(2,1)}).$$

The representation (33) can be generalised for arbitrary parameter structures $\mathcal{I}$ and corresponding isomorphism structures $\mathcal{B}_{\mathcal{I}} = (\mathcal{B}(J,J'))_{(J,J')\in\mathcal{I}\times\mathcal{I}}$:

Let $\bar{J} \in \mathcal{I}$ and $\bar{F} \subset \Sigma^*_{\bar{J}}$, then for each $J \in \mathcal{I}$ let

$$F^{\bar{F}}_J := \bigcup_{J'\in\mathcal{I}, J'\subset J, \iota\in\mathcal{B}(\bar{J},J')} (\Pi^J_{J'})^{-1}(\iota^{\bar{J}}_{J'}(\bar{F})). \tag{34}$$

Now by the same argument as in our privacy example, we get

Theorem 6. *Let $(\mathcal{L}_J)_{J\subset\mathcal{I}}$ be a well-behaved scalable system, and let $\bar{F} \subset \Sigma^*_{\bar{J}}$ with $\bar{J} \in \mathcal{I}$, then*

$$\mathcal{L}_J \cap \mathcal{F}^{\bar{F}}_J = \emptyset \text{ for each } J \in \mathcal{I} \text{ iff } \mathcal{L}_{\bar{J}} \cap \mathcal{F}^{\bar{F}}_{\bar{J}} = \emptyset. \tag{35}$$

If $\mathcal{L}_{\bar{J}}$ and $\bar{F}$ are regular subsets of $\Sigma^*_{\bar{J}}$, then (35) can be checked by finite state methods [21].

If $(\mathcal{L}_J)_{J\subset\mathcal{I}}$ is defined as in (27) the regularity of $\check{L}$ and of V_r for each $r \in R$ and finiteness of R and $\bar{J}$ implies regularity of $\mathcal{L}_{\bar{J}}$.

For finite sets $J, \bar{J} \in \mathcal{I}$ with $\#(J) < \#(\bar{J})$, where $\#$ denotes the cardinality of a set, holds $\mathcal{F}^{\bar{F}}_J = \emptyset$, because of $\mathcal{B}(\bar{J}, J') = \emptyset$ for each $J' \in J$ with $J' \in \mathcal{I}$. Therefore,

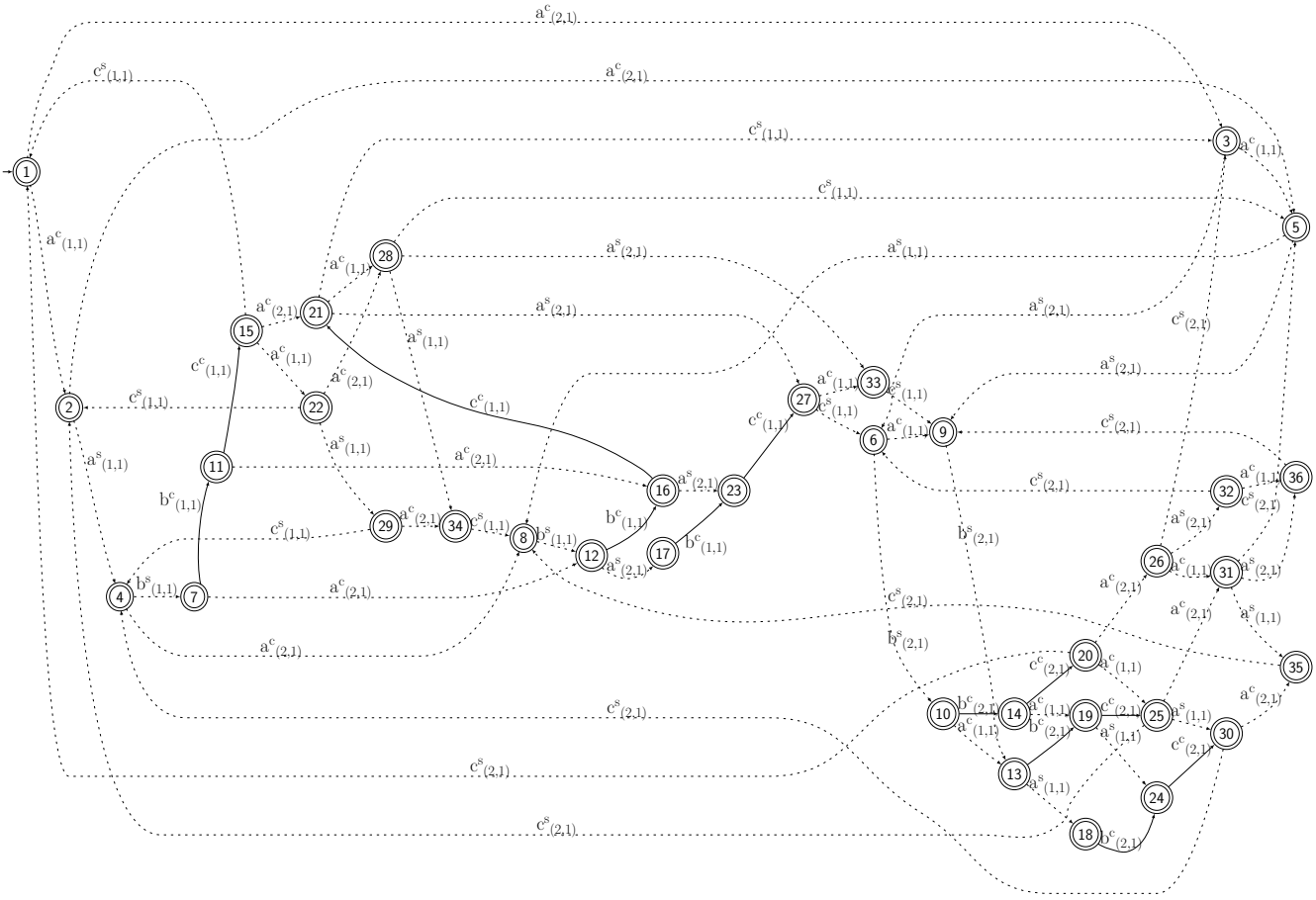


Figure 7. Automaton of $\mathcal{C}_{\{1,2\}\times\{1\}}$

it makes sense to consider safety properties defined by finite unions of sets as defined in (35).

Definition 16 (Scalable safety properties)**.** *Let $\mathcal{I}$ be a parameter structure, $\mathcal{B}_{\mathcal{I}} = (\mathcal{B}(J,J'))_{(J,J')\in\mathcal{I}\times\mathcal{I}}$ a corresponding isomorphism structure, T a finite set, and $\bar{F}_t \subset \Sigma^*_{\bar{J}_t}$ with $\bar{J}_t \in \mathcal{I}$ for each $t \in T$, then $(\mathcal{F}_J)_{J\in\mathcal{I}}$ with $\mathcal{F}_J := \bigcup_{t\in T} \mathcal{F}_J^{\bar{F}_t}$ is called a* scalable *safety property.*

Corollary 1. *For a well-behaved scalable system $(\mathcal{L}_J)_{J\in\mathcal{I}}$ the parameterised problem of verifying a scalable safety property is reduced to finite many finite state problems if the corresponding $\mathcal{L}_{\bar{J}_t}$ and $\bar{F}_t$ are regular languages.*

VIII. Conclusions and Further Work

Structural scalability of a system in terms of the ability to compose a system using a varying number of identical components of a few given types is a desired property that is analysed in this work. For safety critical systems as well as for business critical systems, assuring the correctness of systems composed in such a way is imperative. Thus, the focus of this paper is on property preserving structural scalability.

This motivates the formal definition of well-behaved scalable systems, which starts with a prototype system that fulfils a desired safety property and then "embeds" this prototype system in a scalable system. When this scalable system is constructed according to the methods given in this paper, then corresponding safety properties are fulfilled by any instance of the scalable system. In other words, it is shown that for well-behaved scalable systems a wide class of safety properties can be verified by finite state methods.

For this purpose, a formal framework is presented that can be utilised to construct well-behaved scalable systems in terms of prefix closed formal languages and alphabetic language homomorphisms. The basic parts of that framework are formalisations of parameter structures, influence structures and isomorphisms structures. Together with so-called prototype systems and behaviours of influence these structures formally define scalable systems, if certain conditions are fulfilled. With respect

to such scalable systems, the focus is on properties, which rely on specific component types and a specific number of individual components for these component types but not on the specific individuality a component. Well-behaved scalable systems are characterised by those systems, which fulfil such a kind of property if already one prototype system (depending on the property) fulfils that property. Self-similar scalable systems have this desired property. A sufficient condition for such self-similarity is given in terms of prototype systems and behaviours of influence. A deeper analysis of this condition is subject of a forthcoming paper of the authors.

Usually, behaviour properties of systems are divided into two classes: *safety* and *liveness* properties [25]. Intuitively, a safety property stipulates that "something bad does not happen" and a liveness property stipulates that "something good eventually happens". To extend this verification approach to reliability or general liveness properties, additional assumptions for well-behaved scalable systems have to be established. In [26], such assumptions have been developed for uniformly parametrised two-sided cooperations. To generalise these ideas to a wider class of well-behaved scalable systems is subject of further work.

Acknowledgement

Research reported in this publication was supported by the German Federal Ministry of Education and Research in the context of the project ACCEPT (ID 01BY1206D).

References

[1] P. Ochsenschläger and R. Rieke, "Construction principles for well-behaved scalable systems," in ICONS 2014, The Ninth International Conference on Systems, February 23 - 27, 2014 - Nice, France. IARIA, 2014, pp. 32–39.

[2] L. Duboc, D. S. Rosenblum, and T. Wicks, "A framework for modelling and analysis of software systems scalability," in Proceedings of the 28th International Conference on Software Engineering, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 949–952.

[3] A. B. Bondi, "Characteristics of scalability and their impact on performance," in Workshop on Software and Performance, 2000, pp. 195–203.

[4] S. Bullock and D. Cliff, "Complexity and emergent behaviour in ICT systems," Hewlett-Packard Labs, Tech. Rep. HP-2004-187, 2004.

[5] J. Weinman, "Axiomatic cloud theory," http://www.joeweinman.com/Resources/Joe_Weinman_Axiomatic_Cloud_Theory.pdf, July 2011, [retrieved: Nov, 2014].

[6] P. Zegzhda, D. Zegzhda, and A. Nikolskiy, "Using graph theory for cloud system security modeling," in Computer Network Security, ser. LNCS, I. Kotenko and V. Skormin, Eds. Springer, 2012, vol. 7531, pp. 309–318.

[7] P. Ochsenschläger and R. Rieke, "Security properties of self-similar uniformly parameterised systems of cooperations," in Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on, 2011, pp. 640–645.

[8] S. Schneider, "Security properties and CSP," in IEEE Symposium on Security and Privacy. IEEE Computer Society, 1996, pp. 174–187.

[9] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Trans. Dependable Sec. Comput., vol. 1, no. 1, pp. 11–33, 2004.

[10] C. N. Ip and D. L. Dill, "Verifying systems with replicated components in murφ," Formal Methods in System Design, vol. 14, no. 3, pp. 273–310, 1999.

[11] F. Derepas and P. Gastin, "Model checking systems of replicated processes with SPIN," in Proceedings of the 8th International SPIN Workshop on Model Checking Software (SPIN'01), ser. LNCS, M. B. Dwyer, Ed., vol. 2057. Toronto, Canada: Springer, 2001, pp. 235–251.

[12] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre, "Incremental verification by abstraction," in TACAS, ser. Lecture Notes in Computer Science, T. Margaria and W. Yi, Eds., vol. 2031. Springer, 2001, pp. 98–112.

[13] R. Milner, Communication and Concurrency, ser. International Series in Computer Science. NY: Prentice Hall, 1989.

[14] J. C. Bradfield, "Introduction to modal and temporal mu-calculi (abstract)," in CONCUR, ser. Lecture Notes in Computer Science, L. Brim, P. Jancar, M. Kretínský, and A. Kucera, Eds., vol. 2421. Springer, 2002, p. 98.

[15] S. Basu and C. R. Ramakrishnan, "Compositional analysis for verification of parameterized systems," Theor. Comput. Sci., vol. 354, no. 2, pp. 211–229, 2006.

[16] T. E. Uribe, "Combinations of model checking and theorem proving," in FroCoS '00: Proceedings of the Third International Workshop on Frontiers of Combining Systems. London, UK: Springer, 2000, pp. 151–170.

[17] E. M. Clarke, M. Talupur, and H. Veith, "Environment abstraction for parameterized verification," in VMCAI, ser. Lecture Notes in Computer Science, E. A. Emerson and K. S. Namjoshi, Eds., vol. 3855. Springer, 2006, pp. 126–141.

[18] M. Talupur, "Abstraction techniques for parameterized verification," Ph.D. dissertation, Computer Science Department, Carnegie Mellon University, 2006, CMU-CS-06-169.

[19] K. R. Apt and D. C. Kozen, "Limits for automatic verification of finite-state concurrent systems," Inf. Process. Lett., vol. 22, no. 6, pp. 307–309, May 1986.

[20] I. Suzuki, "Proving properties of a ring of finite-state machines," Inf. Process. Lett., vol. 28, no. 4, pp. 213–214, Jul. 1988.

[21] J. Sakarovitch, Elements of Automata Theory. Cambridge University Press, 2009.

[22] K. Falconer, Fractal Geometry: Mathematical Foundations and Applications. Wiley, 2003.

[23] N. Agoulmine, Autonomic Network Management Principles: From Concepts to Applications. Elsevier Science, 2010.

[24] P. Ochsenschläger and R. Rieke, "Uniform parameterisation of phase based cooperations," http://sit.sit.fraunhofer.de/smv/publications, Fraunhofer SIT, Tech. Rep. SIT-TR-2010/1, 2010, [retrieved: Nov, 2014].

[25] B. Alpern and F. B. Schneider, "Defining liveness," Information Processing Letters, vol. 21, no. 4, pp. 181–185, October 1985.

[26] P. Ochsenschläger and R. Rieke, "Reliability aspects of uniformly parameterised cooperations," in ICONS 2012, The Seventh International Conference on Systems, Reunion Island. IARIA, 2012, pp. 25–34.

Appendix

Theorem 2 (simplest well-behaved scalable systems)**.** *$(\dot{\mathcal{L}}(L)_I)_{I\in\mathcal{I}}$ is a well-behaved scalable system with respect to each isomorphism structure for $\mathcal{I}$ based on N and*

$$\dot{\mathcal{L}}(L)_I = \bigcap_{i\in N} (\tau_i^I)^{-1}(L) \text{ for each } I \in \mathcal{I}.$$

The proof of Theorem 2 will be given in context of influence structures because it consists of special cases of more general results on influence structures (see (59)).

Further requirements, which assure that $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ are well-behaved scalable systems, will be given with respect to $\mathcal{E}_\mathcal{I}$, $\mathcal{B}_\mathcal{I}$, L and V. This will be prepared by some lemmata.

Lemma 3. *Let $\mathcal{E}_\mathcal{I} := (E(t,I))_{(t,I)\in T\times\mathcal{I}}$ be an influence structure for $\mathcal{I}$ indexed by T, and let $V \subset \Sigma^*$. If*

$$E(t,I') = E(t,I) \cap I' \tag{36}$$

for each $t \in T$ and $I, I' \in \mathcal{I}$ $I' \subset I$, then

$$((\tau_{E(t,I)})^{-1}(V))_{I\in\mathcal{I}}$$

is a monotonic parameterised system for each $t \in T$, and by the intersection theorem

$$(\bigcap_{t\in T} (\tau_{E(t,I)})^{-1}(V))_{I\in\mathcal{I}}$$

is a monotonic parameterised system.

Proof: Let $I \in \mathcal{I}$ and $t \in T$. From the definitions of influence homomorphisms and influence structures it follows

$$\tau^I_{E(t,I)}(a_i) = \begin{cases} a \mid & a_i \in \Sigma_{E(t,I)} \\ \varepsilon \mid & a_i \in \Sigma_I \setminus \Sigma_{E(t,I)} \end{cases}.$$

For $I' \subset I$, $I' \in \mathcal{I}$ and $a_i \in \Sigma_{I'}$ then because of (36)

$$\begin{aligned} \tau^I_{E(t,I)}(a_i) &= \begin{cases} a \mid & a_i \in \Sigma_{E(t,I)} \cap \Sigma_{I'} \\ \varepsilon \mid & a_i \in \Sigma_{I'} \cap \Sigma_I \setminus \Sigma_{E(t,I)} \end{cases} \\ &= \begin{cases} a \mid & a_i \in \Sigma_{E(t,I')} \\ \varepsilon \mid & a_i \in \Sigma_{I'} \setminus (\Sigma_{E(t,I)} \cap \Sigma_{I'}) \end{cases} \\ &= \begin{cases} a \mid & a_i \in \Sigma_{E(t,I')} \\ \varepsilon \mid & a_i \in \Sigma_{I'} \setminus \Sigma_{E(t,I')} \end{cases} = \tau^{I'}_{E(t,I')}(a_i), \end{aligned}$$

and therefore

$$(\tau^{I'}_{E(t,I')})^{-1}(V) \subset (\tau^I_{E(t,I)})^{-1}(V) \text{ for } V \subset \Sigma^*.$$

So,

$$((\tau^I_{E(t,I)})^{-1}(V))_{I\in\mathcal{I}}$$

is a monotonic parameterised system for each $t \in T$. ■

Example 13. *Let $\mathcal{I}$ be a parameter structure based on N. For $I \in \mathcal{I}$ and $i \in N$ let:*

$$\dot{E}(i,I) := \begin{cases} \{i\} \mid & i \in I \\ \emptyset \mid & i \in N \setminus I \end{cases}.$$

By the definition of parameter structure $N \neq \emptyset$. So

$$\dot{\mathcal{E}}_\mathcal{I} := (\dot{E}(i,I))_{(i,I)\in N\times\mathcal{I}}$$

defines an influence structure for $\mathcal{I}$ indexed by N. $\dot{\mathcal{E}}_\mathcal{I}$ satisfies (36) and by $\tau^I_i = \tau^I_{\{i\}}$ $\tau^I_i = \tau^I_{\dot{E}(i,I)}$ for $i \in N$ and $I \in \mathcal{I}$.

Now by Lemma 3 for $V \subset \Sigma^*$

$$((\tau^I_i)^{-1}(V))_{I\in\mathcal{I}} \text{ is a monotonic parameterised system} \tag{37}$$

for each $i \in N$.

For this special influence structure $\dot{\mathcal{E}}_\mathcal{I}$ a stronger result can be obtained.

Lemma 4. *Let $\mathcal{I}$ be a parameter structure based on N and $\varepsilon \in L \subset \Sigma^*$. Then*

$$((\tau^I_i)^{-1}(L))_{I\in\mathcal{I}}$$

is a self-similar monotonic parameterised system for each $i \in N$, and by the intersection theorem

$$(\bigcap_{i\in N} (\tau^I_i)^{-1}(L))_{I\in\mathcal{I}}$$

is a self-similar monotonic parameterised system.

Proof: On account of (37)

$$\Pi^I_{I'}((\tau^I_i)^{-1}(L)) = (\tau^{I'}_i)^{-1}(L)$$

has to be shown for $I, I' \in \mathcal{I}$, $I' \subset I$, and $i \in N$.

(37) implies $(\tau^{I'}_i)^{-1}(L) \subset (\tau^I_i)^{-1}(L)$ and therefore,

$$(\tau^{I'}_i)^{-1}(L) = \Pi^I_{I'}((\tau^{I'}_i)^{-1}(L)) \subset \Pi^I_{I'}((\tau^I_i)^{-1}(L)). \tag{38}$$

It remains to show $\Pi^I_{I'}((\tau^I_i)^{-1}(L)) \subset (\tau^{I'}_i)^{-1}(L)$.

Case 1. $i \notin I'$

Because of $\varepsilon \in L$ and $\tau^{I'}_i(w) = \varepsilon$ for $i \notin I'$ and $w \in \Sigma^*_{I'}$ it holds $(\tau^{I'}_i)^{-1}(L) = \Sigma^*_{I'}$ and so

$$\Pi^I_{I'}((\tau^I_i)^{-1}(L)) \subset (\tau^{I'}_i)^{-1}(L) \text{ for } i \notin I'. \tag{39}$$

Case 2. $i \in I'$

From definitions of $\Pi^I_{I'}, \tau^I_i$ and $\tau^{I'}_i$ follows

$$\tau^I_i = \tau^{I'}_i \circ \Pi^I_{I'} \text{ for } i \in I'. \tag{40}$$

For $x \in \Pi^I_{I'}((\tau^I_i)^{-1}(L))$ exists $y \in \Sigma^*_I$ with $\tau^I_i(y) \in L$ and $x = \Pi^I_{I'}(y)$. Because of (40) holds

$$\tau^{I'}_i(x) = \tau^{I'}_i(\Pi^I_{I'}(y)) = \tau^I_i(y) \in L,$$

hence, $x \in (\tau^{I'}_i)^{-1}(L)$. Therefore,

$$\Pi^I_{I'}((\tau^I_i)^{-1}(L)) \subset (\tau^{I'}_i)^{-1}(L) \text{ for } i \in I'. \quad (41)$$

Because of (39), (41) and (38) holds

$$\Pi^I_{I'}((\tau^I_i)^{-1}(L)) = (\tau^{I'}_i)^{-1}(L)$$

for $I, I' \in \mathcal{I}$, $I' \subset I$ and $i \in N$. ■

Intersections of system behaviours play an important role concerning uniformity of parameterisation. Therefore, some general properties of intersections of families of sets will be presented.

Let T be a set. A family $f = (f_t)_{t\in T}$ with $f_t \in F$ for each $t \in T$ is formally equivalent to a function $f : T \to F$ with $f_t := f(t)$.

Let M be a set. A family $f = (f_t)_{t\in T}$ with $f_t \in F = \mathcal{P}(M)$ for each $t \in T$ is called a family of subsets of M.

Let now $T \neq \emptyset$ and f a family of subsets of M. The intersection $\bigcap_{t\in T} f_t$ is defined by

$$\bigcap_{t\in T} f_t = \{m \in M | m \in f_t \text{ for each } t \in T\}. \quad (42)$$

If $f = g \circ h$ with $h : T \to H$ and $g : H \to F$ then

$$\bigcap_{t\in T} f(t) = \bigcap_{x\in h(T)} g(x). \quad (43)$$

If especially $f = h$ and g is the identity on F, then from (43) follows

$$\bigcap_{t\in T} f(t) = \bigcap_{x\in f(T)} x.$$

For a second family of sets $f' : T' \to F$ with $f'(T') = f(T)$ follows then

$$\bigcap_{t\in T} f(t) = \bigcap_{t'\in T'} f(t').$$

In the following we will use family and function notations side by side.

Let $f = (f_t)_{t\in T}$ a family of sets with $f : T \to F = \mathcal{P}(M)$. If $T = \mathring{T} \cup \hat{T}$ with $\hat{T} \neq \emptyset$ and $f(\hat{T}) = \{M\}$, then from (42) follows

$$\bigcap_{t\in T} f(t) = \bigcap_{t\in \mathring{T}} f(t). \quad (44)$$

Let $\mathcal{E}_\mathcal{I} = (E(t,I))_{(t,I)\in T\times\mathcal{I}}$ be an influence structure for $\mathcal{I}$ indexed by T.

For each $I \in \mathcal{I}$ a family of sets

$$\mathcal{E}_\mathcal{I}(I) := (E(t,I))_{t\in T}$$

with $E(t,I) = \mathcal{E}_\mathcal{I}(I)(t) \in \mathcal{P}(I)$ is defined, and it holds

$$\mathcal{E}_\mathcal{I}(I) : T \to \mathcal{P}(I).$$

From (43) it follows (with $h = \mathcal{E}_\mathcal{I}(I)$)

$$\bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V) = \bigcap_{x\in\mathcal{E}_\mathcal{I}(I)(T)} (\tau^I_x)^{-1}(V) \quad (45)$$

for each $V \subset \Sigma^*$ and $I \in \mathcal{I}$.

For each $I \in \mathcal{I}$ holds $\tau^I_\emptyset(w) = \varepsilon$ for each $w \in \Sigma^*_I$. It follows,

$$(\tau^I_\emptyset)^{-1}(V) = \Sigma^*_I \text{ if } \varepsilon \in V \subset \Sigma^*. \quad (46)$$

Because of (43), (44), (45), and (46)

$$\begin{aligned}\bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V) &= \bigcap_{x\in\mathcal{E}_\mathcal{I}(I)(T_I)} (\tau^I_x)^{-1}(V) \\ &= \bigcap_{t\in T_I} (\tau^I_{E(t,I)})^{-1}(V) \quad (47)\end{aligned}$$

for each T_I with $\emptyset \neq T_I \subset T$ and $\mathcal{E}_\mathcal{I}(I)(T) \setminus \mathcal{E}_\mathcal{I}(I)(T_I) \in \{\emptyset, \{\emptyset\}\}$ and $\varepsilon \in V \subset \Sigma^*$.

Each bijection $\iota : I \to I'$ defines another bijection $\breve{\iota} : \mathcal{P}(I) \to \mathcal{P}(I')$ by

$$\breve{\iota}(x) := \{\iota(y) \in I' | y \in x\} \text{ for each } x \in \mathcal{P}(I).$$

Lemma 5. *Let $\mathcal{E}_\mathcal{I} = (E(t,I))_{(t,I)\in T\times\mathcal{I}}$ be an influence structure for $\mathcal{I}$ indexed by T, and let $\mathcal{B}_\mathcal{I} = (\mathcal{B}(I,I'))_{(I,I')\in\mathcal{I}\times\mathcal{I}}$ be an isomorphism structure for $\mathcal{I}$. Let*

$$\begin{aligned}&\varepsilon \in V \subset \Sigma^*, \text{ and let } (T_K)_{K\in\mathcal{I}} \text{ be a family} \\ &\text{with } \emptyset \neq T_K \subset T \text{ and} \\ &\mathcal{E}_\mathcal{I}(K)(T) \setminus \mathcal{E}_\mathcal{I}(K)(T_K) \in \{\emptyset, \{\emptyset\}\} \text{ for each } K \in \mathcal{I}, \\ &\text{such that } \breve{\iota}(\mathcal{E}_\mathcal{I}(I)(T_I)) = \mathcal{E}_\mathcal{I}(I')(T_{I'}) \\ &\text{for each } (I,I') \in \mathcal{I}\times\mathcal{I} \text{ and } \iota \in \mathcal{B}(I,I'), \quad (48)\end{aligned}$$

then

$$\bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V) = \bigcap_{t\in T_I} (\tau^I_{E(t,I)})^{-1}(V) \quad (49)$$

for each $I \in \mathcal{I}$, and

$$\iota^I_{I'}[\bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V)] = \bigcap_{t\in T} (\tau^{I'}_{E(t,I')})^{-1}(V) \quad (50)$$

for each $(I,I') \in \mathcal{I}\times\mathcal{I}$ and $\iota \in \mathcal{B}(I,I')$.

Proof of (49)*:* Because of (47) from assumption (48) directly follows (49). ■

For the proof of (50) the following property of the homomorphisms τ^I_K is needed:

Let $\iota : I \to I'$ a bijection and $K \subset I$, then $\tau^{I'}_{\iota(K)} \circ \iota^I_{I'} = \tau^I_K$ and so

$$\tau^{I'}_{\iota(K)} = \tau^I_K \circ (\iota^I_{I'})^{-1}. \quad (51)$$

Proof of (51)*:*

The elements of Σ_I are of the form a_i with $i \in I$ and $a \in \Sigma$. For these elements holds

$$\begin{aligned}\tau^I_K(a_i) &= \begin{cases} a \mid & i \in K \\ \varepsilon \mid & i \in I \setminus K \end{cases} \\ &= \begin{cases} a \mid & \iota(i) \in \iota(K) \\ \varepsilon \mid & \iota(i) \in I' \setminus \iota(K) \end{cases} \\ &= \tau^{I'}_{\iota(K)}(a_{\iota(i)}) = \tau^{I'}_{\iota(K)}(\iota^I_{I'}(a_i)),\end{aligned}$$

which proves (51). ∎

Proof of (50)*:* Because of (47) and (51)

$$\begin{aligned}&\iota^I_{I'}[\bigcap_{t\in T}(\tau^I_{E(t,I)})^{-1}(V)] \\ &= \iota^I_{I'}[\bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^I_x)^{-1}(V)] \\ &= ((\iota^I_{I'})^{-1})^{-1}[\bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^I_x)^{-1}(V)] \\ &= \bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}((\iota^I_{I'})^{-1})^{-1}[(\tau^I_x)^{-1}(V)] \\ &= \bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^I_x \circ (\iota^I_{I'})^{-1})^{-1}(V) \\ &= \bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^{I'}_{\iota(x)})^{-1}(V) \\ &= \bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^{I'}_{\breve{\iota}(x)})^{-1}(V). \qquad (52)\end{aligned}$$

From (43) (with $h = \breve{\iota}$) and the assumption (48) follows

$$\begin{aligned}\bigcap_{x\in\mathcal{E}_{\mathcal{I}}(I)(T_I)}(\tau^{I'}_{\breve{\iota}(x)})^{-1}(V) &= \bigcap_{x'\in\breve{\iota}(\mathcal{E}_{\mathcal{I}}(I)(T_I))}(\tau^{I'}_{x'})^{-1}(V) \\ &= \bigcap_{x'\in\mathcal{E}_{\mathcal{I}}(I')(T'_I)}(\tau^{I'}_{x'})^{-1}(V).\end{aligned}$$

Furthermore, from (47) follows

$$\bigcap_{x'\in\mathcal{E}_{\mathcal{I}}(I')(T'_I)}(\tau^{I'}_{x'})^{-1}(V) = \bigcap_{t\in T}(\tau^{I'}_{E(t,I')})^{-1}(V). \qquad (53)$$

(52) - (53) prove (50). ∎

The case $T = N$, where $\mathcal{I}$ is based on N, allows a simpler sufficient condition for (49) and (50).

Lemma 6. *Let $\mathcal{I}$ be a parameter structure based on N, $\mathcal{E}_{\mathcal{I}} = (E(n,I))_{(n,I)\in N\times\mathcal{I}}$ be an influence structure for $\mathcal{I}$, and let $\mathcal{B}_{\mathcal{I}} = (\mathcal{B}(I,I'))_{(I,I')\in\mathcal{I}\times\mathcal{I}}$ be an isomorphism structure for $\mathcal{I}$.*

Let $\varepsilon \in V \subset \Sigma^*$, (54a)

for each $I \in \mathcal{I}$ and $n \in N$ let $E(n,I) = \emptyset$, or it exists an $i_n \in I$ with $E(n,I) = E(i_n,I)$, and (54b)

for each $(I,I') \in \mathcal{I}\times\mathcal{I}, \iota \in \mathcal{B}(I,I')$ and $i \in I$ holds $\iota(E(i,I)) = E(\iota(i),I')$. (54c)

Then

$$\bigcap_{n\in N}(\tau^I_{E(n,I)})^{-1}(V) = \bigcap_{n\in I}(\tau^I_{E(n,I)})^{-1}(V)$$

for each $I \in \mathcal{I}$, and

$$\iota^I_{I'}[\bigcap_{n\in N}(\tau^I_{E(n,I)})^{-1}(V)] = \bigcap_{n\in N}(\tau^{I'}_{E(n,I')})^{-1}(V)$$

for each $(I,I') \in \mathcal{I}\times\mathcal{I}$ and $\iota \in \mathcal{B}(I,I')$.

Proof: From (54b) follows $\mathcal{E}_{\mathcal{I}}(I)(N) = \mathcal{E}_{\mathcal{I}}(I)(I)$ or $\mathcal{E}_{\mathcal{I}}(I)(N) = \mathcal{E}_{\mathcal{I}}(I)(I)\cup\{\emptyset\}$, so

$$\mathcal{E}_{\mathcal{I}}(I)(N) \setminus \mathcal{E}_{\mathcal{I}}(I)(I) \in \{\emptyset, \{\emptyset\}\} \text{ for each } I \in \mathcal{I}. \qquad (55)$$

From (54c) follows

$$\breve{\iota}(\mathcal{E}_{\mathcal{I}}(I)(I)) \subset \mathcal{E}_{\mathcal{I}}(I')(I'). \qquad (56)$$

Because $\iota : I \to I'$ is a bijection, for each $i' \in I'$ exists an $i \in I$ with $\iota(i) = i'$. Because of (54c) holds $\breve{\iota}(E(i,I)) = E(i',I')$, where $E(i,I) \in \mathcal{E}_{\mathcal{I}}(I)(I)$. From this follows

$$\mathcal{E}_{\mathcal{I}}(I')(I') \subset \breve{\iota}(\mathcal{E}_{\mathcal{I}}(I)(I)). \qquad (57)$$

Because of (55) - (57), with $T = N$ and $(T_I)_{I\in\mathcal{I}} = (I)_{I\in\mathcal{I}}$,

(54a) – (54c) implies (48).

∎

Example 14 (Example 13 (continued))**.** *Let $\mathcal{I}$ be a parameter structure based on N and $\mathcal{B}_{\mathcal{I}} = (\mathcal{B}(I,I'))_{(I,I')\in\mathcal{I}\times\mathcal{I}}$ be an isomorphism structure for $\mathcal{I}$. Then $\dot{\mathcal{E}}_{\mathcal{I}}$ satisfies* (54b) *and* (54c)*.*

So for $\varepsilon \in L \subset \Sigma^*$ Lemma 6 implies

$$\begin{aligned}&\bigcap_{n\in N}(\tau^I_n)^{-1}(L) = \bigcap_{n\in I}(\tau^I_n)^{-1}(L) \text{ for each } I \in \mathcal{I} \text{ and} \\ &\iota^I_{I'}[\bigcap_{n\in N}(\tau^I_n)^{-1}(L)] = \bigcap_{n\in N}(\tau^{I'}_n)^{-1}(L) \qquad (58)\end{aligned}$$

for each $(I,I') \in \mathcal{I}\times\mathcal{I}$ and $\iota \in \mathcal{B}(I,I')$.

Now Lemma 4 together with (58) proves Theorem 2. (59)

Because of $\tau^I_n = \tau^I_{\dot{E}(n,I)}$ for $I \in \mathcal{I}$ and $n \in N$, (58) and the definitions of $(\dot{\mathcal{L}}(L)_I)_{I\in\mathcal{I}}$ and $(\mathcal{L}(L,\mathcal{E}_{\mathcal{I}},V)_I)_{I\in\mathcal{I}}$ imply

$$\begin{aligned}\dot{\mathcal{L}}(L)_I &= \bigcap_{n\in I}(\tau^I_n)^{-1}(L) = \bigcap_{n\in I}(\tau^I_n)^{-1}(L) \cap \bigcap_{n\in I}(\tau^I_n)^{-1}(V) \\ &= \dot{\mathcal{L}}(L)_I \cap \bigcap_{n\in N}(\tau^I_n)^{-1}(V) \\ &= \dot{\mathcal{L}}(L)_I \cap \bigcap_{n\in N}(\tau^I_{\dot{E}(n,I)})^{-1}(V) \\ &= \mathcal{L}(L,\dot{\mathcal{E}}_I,V)_I \qquad (60)\end{aligned}$$

for $I \in \mathcal{I}$ and $V \supset L$.

(60) gives a representation of $(\dot{\mathcal{L}}(L)_I)_{I\in\mathcal{I}}$ in terms of $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$.

For the following theorems please remember that by the general definition of $\mathcal{L}(L,\mathcal{E}_I,V)_I$ it is assumed that $\emptyset \neq L \subset V$ and L,V are prefix closed. This implies $\varepsilon \in L \subset V$.

Lemma 7. *Let $\mathcal{I}$ be a parameter structure, $\mathcal{E}_\mathcal{I}$ an influence structure for $\mathcal{I}$ indexed by T and $\mathcal{B}_\mathcal{I}$ an isomorphism structure for $\mathcal{I}$.*

Assuming (36) *and* (48)*, then*

$$(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$$

is a scalable systems with respect to $\mathcal{B}_\mathcal{I}$. It holds

$$\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I = \dot{\mathcal{L}}(L)_I \cap \bigcap_{n\in T_I} (\tau^I_{E(n,I)})^{-1}(V)$$

for each $I \in \mathcal{I}$.

Proof: By Theorem 2, $(\dot{\mathcal{L}}(L)_I)_{I\in\mathcal{I}}$ is a scalable system with respect to $\mathcal{B}_\mathcal{I}$. By Lemma 3 and 5 (50)

$$(\bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V))_{I\in\mathcal{I}}$$

is a scalable system with respect to $\mathcal{B}_\mathcal{I}$ too. Now part (ii) of the intersection theorem proves $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ to be a scalable system with respect to $\mathcal{B}_\mathcal{I}$. Lemma 5 (49) completes the proof of Lemma 7. ■

Using Lemma 6 instead of Lemma 5 proves the following.

Theorem 3 (construction condition for scalable systems)**.** *By the assumptions of Lemma 6 and* (36) *with $T = N$, $(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$ is a scalable system with respect to $\mathcal{B}_\mathcal{I}$. It holds*

$$\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I = \dot{\mathcal{L}}(L)_I \cap \bigcap_{n\in I} (\tau^I_{E(n,I)})^{-1}(V)).$$

Remark 3. *It can be shown that in* $\mathrm{SP}(L,V)$ $\mathbb{N}$ *can be replaced by each countable infinite set.*

More precisely, let N' be another set and $\iota : \mathbb{N} \to N'$ a bijection. $\iota^\mathbb{N}_{N'} : \Sigma^*_\mathbb{N} \to \Sigma^*_{N'}$ is the isomorphism defined as in the definition of isomorphism structure. It now holds

$$\Theta^\mathbb{N} = \Theta^{N'} \circ \iota^\mathbb{N}_{N'} \text{ and } \tau^\mathbb{N}_n = \tau^{N'}_{\iota(n)} \circ \iota^\mathbb{N}_{N'} \tag{61}$$

for each $n \in \mathbb{N}$. Furthermore,

$$\iota^\mathbb{N}_{N'} \circ \Pi^\mathbb{N}_K = \Pi^{N'}_{\iota(K)} \circ \iota^\mathbb{N}_{N'} \tag{62}$$

for each $K \subset \mathbb{N}$. From (61) and commutativity of intersection now

$$\begin{aligned}(\bigcap_{n\in\mathbb{N}} (\tau^\mathbb{N}_n)^{-1}(L)) \cap (\Theta^\mathbb{N})^{-1}(V) = \\ = (\iota^\mathbb{N}_{N'})^{-1}[(\bigcap_{n\in\mathbb{N}} (\tau^{N'}_{\iota(n)})^{-1}(L)) \cap (\Theta^{N'})^{-1}(V)] \\ = (\iota^\mathbb{N}_{N'})^{-1}[(\bigcap_{n'\in N'} (\tau^{N'}_{n'})^{-1}(L)) \cap (\Theta^{N'})^{-1}(V)].\end{aligned} \tag{63}$$

By (62),

$$\Pi^\mathbb{N}_K \circ (\iota^\mathbb{N}_{N'})^{-1} = (\iota^\mathbb{N}_{N'})^{-1} \circ \Pi^{N'}_{\iota(K)}. \tag{64}$$

Because of (63) and (64)

$$\begin{aligned}\Pi^\mathbb{N}_K[(\bigcap_{n\in\mathbb{N}} (\tau^\mathbb{N}_n)^{-1}(L)) \cap (\Theta^\mathbb{N})^{-1}(V)] = \\ = (\iota^\mathbb{N}_{N'})^{-1}(\Pi^{N'}_{\iota(K)}[(\bigcap_{n'\in N'} (\tau^{N'}_{n'})^{-1}(L)) \cap (\Theta^{N'})^{-1}(V)]).\end{aligned}$$

From

$$\Pi^\mathbb{N}_K[(\bigcap_{n\in\mathbb{N}} (\tau^\mathbb{N}_n)^{-1}(L)) \cap (\Theta^\mathbb{N})^{-1}(V)] \subset (\Theta^\mathbb{N})^{-1}(V)$$

now follows

$$\begin{aligned}\Pi^{N'}_{\iota(K)}[(\bigcap_{n'\in N'} (\tau^{N'}_{n'})^{-1}(L)) \cap (\Theta^{N'})^{-1}(V)] \\ \subset \iota^\mathbb{N}_{N'}((\Theta^\mathbb{N})^{-1}(V)).\end{aligned} \tag{65}$$

Because of (61) $\Theta^\mathbb{N} \circ (\iota^\mathbb{N}_{N'})^{-1} = \Theta^{N'}$ and so

$$(\Theta^{N'})^{-1}(V) = \iota^\mathbb{N}_{N'}((\Theta^\mathbb{N})^{-1}(V)).$$

Therefore, from (65) follows

$$\Pi^{N'}_{\iota(K)}[(\bigcap_{n'\in N'} (\tau^{N'}_{n'})^{-1}(L)) \cap (\Theta^{N'})^{-1}(V)] \subset (\Theta^{N'})^{-1}(V).$$

Because for each $\emptyset \neq K' \subset N'$ it exists an $\emptyset \neq K \subset \mathbb{N}$ with $K' = \iota(K)$, by $\mathrm{SP}(L,V)$, we get for each $\emptyset \neq K \subset \mathbb{N}$ a corresponding inclusion with N' replacing $\mathbb{N}$ and K' for K.

Lemma 8. *The assumptions of Lemma 3 and Lemma 4 together with* $\mathrm{SP}(L,V)$ *imply that $(X(L,V,t)_I)_{I\in\mathcal{I}}$ with*

$$X(L,V,t)_I := \bigcap_{n\in N} (\tau^I_n)^{-1}(L) \cap (\tau^I_{E(t,I)})^{-1}(V)$$

is a self-similar monotonic parameterised system for each $t \in T$.

Proof: By Lemma 3 and Lemma 4, $((\tau^I_{E(t,I)})^{-1}(V))_{I\in\mathcal{I}}$ and $(\bigcap_{n\in N} (\tau^I_n)^{-1}(L))_{I\in\mathcal{I}}$ are monotonic parameterised systems. So by the intersection

theorem $(X(L,V,t)_I)_{I\in\mathcal{I}}$ is a monotonic parameterised system for each $t \in T$. Therefore,

$$X(L,V,t)_{I'} = \Pi^I_{I'}(X(L,V,t)_{I'}) \subset \Pi^I_{I'}(X(L,V,t)_I)$$

for each $I, I' \in \mathcal{I}$ with $I' \subset I$. So the proof of self-similarity can be reduced to the proof of

$$\Pi^I_{I'}(X(L,V,t)_I) \subset X(L,V,t)_{I'} \tag{66}$$

for each $t \in T$ and $I, I' \in \mathcal{I}$ with $I' \subset I$.

Because by Lemma 4

$$(\bigcap_{n\in N} (\tau^I_n)^{-1}(L))_{I\in\mathcal{I}}$$

is self-similar, it holds

$$\Pi^I_{I'}(X(L,V,t)_I) \subset \Pi^I_{I'}(\bigcap_{n\in N} (\tau^I_n)^{-1}(L)) = \bigcap_{n\in N} (\tau^I_n)^{-1}(L).$$

So the proof of (66) can be reduced to the proof of

$$\Pi^I_{I'}[\bigcap_{n\in N} (\tau^I_n)^{-1}(L) \cap (\tau^I_{E(t,I)})^{-1}(V)] \subset (\tau^{I'}_{E(t,I')})^{-1}(V) \tag{67}$$

for each $t \in T$ and $I, I' \in \mathcal{I}$ with $I' \subset I$.

For each

$$w \in (\bigcap_{n\in N} (\tau^I_n)^{-1}(L)) \cap (\tau^I_{E(t,I)})^{-1}(V)$$

exists a $r \in \mathbb{N}$ and $u_i \in \Sigma^*_{E(t,I)}$ for $1 \le i \le r$ and $v_i \in \Sigma^*_{I\setminus E(t,I)}$ for $1 \le i \le r$ with $w = u_1v_1u_2v_2\dots u_rv_r$. Note that $\Sigma_\emptyset := \emptyset$ and $\emptyset^* = \{\varepsilon\}$. Because $u_1u_2\dots u_r \in \Sigma^*_{E(t,I)}$ and $v_1v_2\dots v_r \in \Sigma^*_{I\setminus E(t,I)}$ holds

$$\begin{aligned}\Theta^N(u_1u_2\dots u_r) &= \tau^I_{E(t,I)}(u_1u_2\dots u_r)\\ &= \tau^I_{E(t,I)}(w) \in V.\end{aligned} \tag{68}$$

With the same argumentation holds

$$\tau^N_n(u_1u_2\dots u_r) = \tau^I_n(u_1u_2\dots u_r) = \tau^I_n(w) \in L \tag{69}$$

for $n \in E(t,I)$ and

$$\tau^N_n(u_1u_2\dots u_r) = \varepsilon \in L \tag{70}$$

for $n \in N \setminus E(t,I)$. With (68) - (70) now

$$u_1u_2\dots u_r \in (\bigcap_{n\in N} (\tau^N_n)^{-1}(L)) \cap (\Theta^N)^{-1}(V),$$

and on behalf of precondition $\mathrm{SP}(L,V)$ holds

$$\begin{aligned}\Pi^N_{I'}(u_1u_2\dots u_r) =& \Pi^{E(t,I)}_{I'\cap E(t,I)}(u_1u_2\dots u_r)\\ &\in \Sigma^*_{I'\cap E(t,I)} \cap (\Theta^N)^{-1}(V).\end{aligned} \tag{71}$$

Furthermore,

$$\begin{aligned}\Pi^I_{I'}(w) =& \Pi^I_{I'}(u_1v_1u_2v_2\dots u_rv_r)\\ =& \Pi^{E(t,I)}_{I'\cap E(t,I)}(u_1)\Pi^{I\setminus E(t,I)}_{I'\setminus E(t,I)}(v_1)\dots\\ & \Pi^{E(t,I)}_{I'\cap E(t,I)}(u_r)\Pi^{I\setminus E(t,I)}_{I'\setminus E(t,I)}(v_r).\end{aligned} \tag{72}$$

Because of (36), $E(t,I') \subset E(t,I)$ and so $I' \setminus E(t,I) \subset I' \setminus E(t,I')$ and thus

$$\tau^{I'}_{E(t,I')}(\Pi^{I\setminus E(t,I)}_{I'\setminus E(t,I)})(v_i) = \varepsilon$$

for $1 \le i \le r$. With (36) and (72) it follows

$$\tau^{I'}_{E(t,I')}(\Pi^I_{I'}(w)) = \tau^{I'}_{E(t,I')}(\Pi^{E(t,I)}_{E(t,I')}(u_1\dots u_r)). \tag{73}$$

Because $\tau^{I'}_{E(t,I')}(x) = \Theta^N(x)$ for each $x \in \Sigma^*_{E(t,I')}$ now on behalf of (73), (36), and (71)

$$\tau^{I'}_{E(t,I')}(\Pi^I_{I'}(w)) = \Theta^N(\Pi^{E(t,I)}_{E(t,I')}(u_1\dots u_r)) \in V,$$

and thus

$$\Pi^I_{I'}(w) \in (\tau^{I'}_{E(t,I')})^{-1}(V).$$

This proves (67) and completes the proof of Lemma 8. ■

Because of the idempotence of intersection

$$\begin{aligned}&\bigcap_{n\in N} (\tau^I_n)^{-1}(L) \cap \bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V)\\ &\quad = \bigcap_{t\in T}[\bigcap_{n\in N} (\tau^I_n)^{-1}(L) \cap (\tau^I_{E(t,I)})^{-1}(V)].\end{aligned}$$

Now the intersection theorem and Lemma 8 imply

Lemma 9. *If* $\mathrm{SP}(L,V)$*, then by the assumptions of Lemma 3 and 4*

$$[\bigcap_{n\in N} (\tau^I_n)^{-1}(L) \cap \bigcap_{t\in T} (\tau^I_{E(t,I)})^{-1}(V)]_{I\in\mathcal{I}}$$

is a self-similar monotonic parameterised system.

Combining Lemma 9 with Lemma 7 or Theorem 3 imply

Theorem 4 (construction condition for well-behaved scalable systems)**.** *By the assumptions of Lemma 7 or Theorem 3 together with* $\mathrm{SP}(L,V)$

$$(\mathcal{L}(L,\mathcal{E}_\mathcal{I},V)_I)_{I\in\mathcal{I}}$$

is a well-behaved scalable system.

Dynamic Pattern Development for UAV Navigation Support

Florian Segor, Igor Tchouchenkov, Sebastian Friedrich, Anna Nehaichik, and Chen-Ko Sung
IAS – Department Interoperability and Assistance Systems
Fraunhofer IOSB
Karlsruhe, Germany
{florian.segor, igor.tchouchenkov, sebastian.friedrich, anna.nehaichik}@iosb.fraunhofer.de
chen-ko.sung@sung.de

***Abstract*—Electrically operated Vertical Takeoff and Landing (VTOL) Unmanned Aerial Vehicle (UAV) systems are used for aerial situation awareness and reconnaissance for civil security because they can be controlled easily on account of the simple handling and the good maneuverability even during applications in urban areas. The applications of such systems for rescue purposes strongly increase and, therefore, the need for professional support systems arises steadily. Takeoffs of a VTOL UAV system and in particular the landing have no meaning for the quality of a reconnaissance operation, but require the undivided attention of the operator. To automate takeoff and landing, the concept of a dynamic ground pattern for position correction and communication is suggested. The developed procedure is drafted and the advancement of the basic pattern projecting technology to two different working prototypes is described. The suitability of the prototypes is examined and reviewed. Main focus, in this occasion, is on the comparison of the different pattern projecting technologies to provide a statement about their strengths and weaknesses.**

Keywords-automatic UAV guidance; pattern projector; pattern detection; visual communication; civil rescue forces.

I. INTRODUCTION

As already illustrated in [1], there are various systems and sensors to support rescue forces in their work to manage natural or manmade disasters. One focus of the research done at Fraunhofer IOSB is the application of modern sensors and sensor carriers to support police and rescue forces in such situations. The project AMFIS [2] is concerned with developing an adaptable modular system for managing heterogenic mobile, as well as stationary sensors. The main task of its ground control station is to serve as an ergonomic user interface and a data integration hub between multiple sensors mounted on light UAVs, Unmanned Ground Vehicles (UGVs), stationary platforms (network cameras), ad hoc networked sensors, etc. and a super-ordinated control center.

Within the amount of different sensor carriers already integrated in the laboratory test bed, micro UAVs, especially small VTOL systems, play a special role. An application of multi-rotor systems within rescue or security scenarios had become more realistic in recent years because of the rising usability and higher levels of automation. The further extension of the application ability and the computer-guided-control of these sensor carriers is also within the focus of research done in the AMFIS project. The aim is a ground control station permitting a single operator to control a complex heterogeneous reconnaissance system, not only sequentially by dealing with one sensor carrier at a time, but in parallel with reduced workload and supported by a high level of automation.

Our experiments in the past have shown that the achieved level of automation is sufficient in most cases for the automated application of multiple sensor carriers with a minimum of operator interaction [3][4][5].

Though, the automatic take off process of a GPS supported VTOL UAV is possible without supervision, however, this flight sequence is far away from an absolutely secure procedure and can be further improved therefore.

The landing process needs the unlimited attention of the user or a manual steering pilot because the navigation based on GPS and pressure sensors is in most cases not precise enough for a secure, unattended, automatic landing when space is the limiting factor.

To remove these restrictions and to protect the aircraft as well as the personnel and the material near the lift off and landing site, procedures were developed to provide an on board visual detection of ground pattern to use this information for an exact automatic landing [6].

However, using a static pattern, some problems and limitations have to be considered. Flying on different altitudes, the size of a static pattern varies and a partial coverage of the pattern is inevitable on low altitudes making it hard to provide robust pattern detection. To cope with these problems we extended the concept of using a visual fix point to provide a safe landing by introducing a dynamic pattern that can changes its representation in size and content. Therefore, it can be adapted to the altitude of the UAV and reduces the detection of false positives by an addition logical level within the detection process. In addition, dynamic patterns can be used as a communication channel to control the UAV.

For these reasons, the developed basic detection algorithms were designed to be capable of detecting different patterns and to extract additional information from the ground pattern as for example deviation from the approach path or the direction and speed of a potential movement of the landing platform (if, e.g., mounted on a vehicle).

The Introduction will be followed by review of related research in the field of automatic UAV landing facilitating pattern detection systems. Section III is describing the application scenario and the addressed problems in detail, followed by the subsumed results in Section IV on the original pattern recognition. Section V is introducing the main topic of this paper dealing with the development of different dynamic pattern techniques to create an advanced test bed that allows an intense validation of the overall concept. This is succeeded by an assessment of the created pattern systems in Section VI. Finally, the results are recapitulated in Section VII followed by conclusion and future work.

II. Related Work

With the advance of the technological progress, UAVs can be successfully used for more and more applications. Hence, during the last 10 years, varied research results concerning UAV-swarming, independent navigation behavior, sense-and-avoid procedures and also work within the topic of automatic landing and lift off were published.

Within the field of research about the automatic landing of a VTOL UAV, the principle of using a ground pattern and visual pattern recognition for navigation and position extraction has been treated extensively. This application of visual extraction poses a special problem within the field of image exploitation. Procedures for the processing and recognition of structures in a video stream are used in different areas professionally. For example number plate recognition or the automatic detection of deposit bottles in sorting machines should be mentioned. However, in most applications position, distance and orientation of the pattern to detect can be forecasted very exactly reducing the complexity of the application. This does not apply when using pattern recognition as a navigation support on board a moving UAV. The pattern can become visible in different distances, dimensions and rotations and, hence, poses a more complicated problem in the field of image exploitation. Nevertheless, the usability and applicability of this approach is undoubted according to the achieved success.

S. Sharp et al. [7] presented a test bed for onboard detection of a defined ground pattern using Commercial Of The Shelf (COTS) camera and hardware components.

Saripalli examines a very interesting application in [8] using a pattern detection algorithm on board of a small unmanned rotary aircraft. A theoretical approach to track and to land the UAV on a co-operative moving object is presented.

Zhou et al. [9] as well as Yang et al. [10] examined the possibilities of an autonomous landing on a static "H"-shaped pattern. Especially, Yang pays special attention to the high noise immunity and the rotation independence of the detection algorithm.

Xiang et al. [11] describe a very interesting set up with low-cost COTS components (IR Cam of the Wii remote). The components are used to build an active IR pattern for the positioning system of a multi-rotor UAV.

Lange et al. [12] also address the landing of an UAV on a ground pattern. They concentrate on handling the problem of the discrete scaling of the pattern independent of the different flight altitudes of the UAV by introducing a special designed circular ground pattern. Through different circles, which are becoming smaller to the centre of the pattern, the algorithm is capable of detecting the landing site also during the final flight stage of an approach without the need to adapt the absolute magnitude of the pattern.

A similar approach is followed by Richardson et al. in [13], describing the landing of an autonomous UAV on a moving ground platform by using a pattern detection algorithm in co-operative surroundings. As in [12], a multistage pattern, which enables the complete visibility of the pattern for on board recognition also at a low flight level, is used.

All these researchers have shown good success in addressing very similar purposes. However, the suggested solutions suffer from some limitations as for example the restrictions due to the missing discrete pattern scaling during landing and takeoff. Additionally, each static pattern approach can react on a pattern-like natural or man-made structure with miss-interpretation or detection errors.

The dynamic pattern introduced in this research allows the construction of an additional communication link to the UAV and, besides, solves problems, which are not handled yet.

III. Application Scenario and Motivation

One of the central application scenarios of the AMFIS system is to deal with the support of rescue forces in disasters or accidents. The varied application of different sensors on board of a UAV can be used to acquire important reconnaissance information to make the work of the people in the field more safe and efficient. Derived from the experiments done with the AMFIS system, the missing capability of the UAVs used within these scenarios to precisely take off and land automatically on a designated position was identified as one of the main challenges for the professional application – especially when multiple UAVs are deployed at the same time.

The endurance of electrically operating UAVs is limited and in most cases several take offs and landings become necessary in order to fulfill the mission. In these flight phases the UAV must be supervised and neither the operator nor the UAV can contribute to the mission's target. To automate these flight phases the navigation exactness needs be improved. A visually extracted geographical fix point at the landing position is, on this occasion, a promising start. The here presented draught is based on already achieved success with visually extracted patterns and extended to use dynamic pattern recognition with the aim to receive a more stable and reliable navigation support.

A dynamic pattern is not necessarily compelling for the solution of the primary problem and quite good results were achieved using non-dynamic, static patterns. Indeed, a dynamic pattern offers additional advantages which extend the application possibilities of such a system. Just by using

the access to an, in principle, almost unlimited pool of different signs and symbols, the abilities of a pattern concept can be clearly enlarged. By that, the detection capability of the algorithm is not limited to the pure localization of the pattern any more. It can be extended by the functionality to extract information content hidden within a detected pattern. Besides, a dynamic pattern still offers some other advantages. As already Lange et al. [12] stressed out, an essential problem within using ground patterns originates from the detection of a static pattern at different flight altitudes. Even when using a fish-eye lens during an approach of the sensor to the pattern, the probability rises that parts of the pattern are not grasped by the sensor because of the limited aperture angle and the increasing appearance of the image or pattern. The use of a dynamically adaptable pattern allows resizing the shown pattern. Thus, the size of the pattern can be adjusted matching the current flight altitudes raising the chance that the sensor is capable of viewing the shape completely. Though, the algorithm is designed to be rotation and scale independent, nevertheless, the result quality of the detection algorithm could possibly be further improved by aligning the orientation of the pattern with the direction of the UAV as well as considering its point of view and distorting its perspective. An optimized projection of the pattern considering not only distance but also the orientation and view angles is assumed to potentially reduce the load on the low-power on-board processor.

However, the introduction of an additional visual communication channel provides even more advantages. Unfortunately, the widely used radio data connections between UAVs and their dedicated ground stations can be very easily disturbed - intentionally or unintentionally. The detection of a used radio frequency can be done using COTS systems and even if it is not so easy to break into the communication to take over the UAV, in most cases it can be overlaid leading to a complete communication breakdown between the ground control and the aerial system. Using a visual communication system, interfering with the communication becomes more difficult because a potential disrupter stays hardly unnoticed if applying a permanent influence on the pattern providing ground platform.

IV. Onboard Detection Chain

The basic functions for adaptive pattern recognition on board the UAV have been reported in [5]. The implemented on board detection chain basically consists of two major tasks.

The first task is the separation and extraction of possible pattern sub images from image sequences as pattern candidates for the recognition and interpretation of manmade landmarks. The implemented process chain with an adaptive threshold operation for this task works well and has not been modified for the present investigation.

The second task is the recognition of patterns or manmade landmark images from the identified candidates. The challenge of this task is that the onboard process for image evaluation must be robust, non-compute-intensive, expandable and fast. For that reason, we developed a so-called "zigzag" method, which analyzes how many binary values of relevant parts of an object image are correlated with the expected values within the selected region identified as a possible pattern.

The previous investigation has shown that the methods and the complete on board detection chain is stable, easy to extend and provides good results on detecting the patterns on the ground in different conditions.

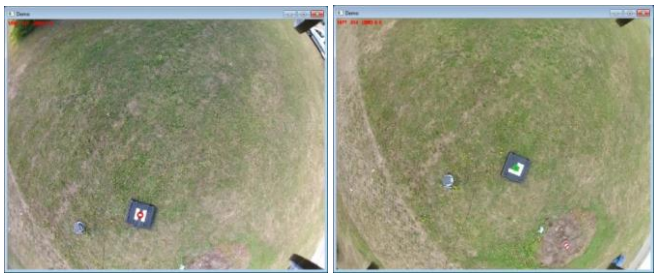


Figure 1. In-flight detection of shape "H" and "L" marked by colored circles at the center of the pattern ("H" is marked red; "L" is marked green) camera: GoPro Hero 2, altitude: 30 metres.

An important part in the first task of the process chain is the recalculation of identified possible patterns. These sub image regions are translated into a standard region. The algorithm inherits therefore some serious advantages, as for example the rotation and scaling independence necessary for an UAV application (see detection in Figure 1).

At the same time the designed is not limited to only detect a pattern on the ground to calculate correct and GPS independent navigation information, but also to extract information from the different pattern sequences. The used "zig-zag" method has great advantages because of the fast and simple logic, used to recognize a single pattern. The procedure is quick and efficient and, hence, suited to deliver usable results with limited hardware capacity onboard, which has been proven in the past attempts. Using the detection of different signs in different sequences for creating a pattern language allows the transmission of reduced information form the ground to the aerial system.

Figure 2. Examples of used patterns.

To achieve a sufficient information density, the number of different patterns has to be enlarged to reach the capability to transmit more complex information by combining symbols (see Figure 2).

This can be seen as one other the key features of the dynamic pattern detection beside the improvement of the navigational information for the automatic landing. As already mentioned above, different patterns are shown at the same projection plane sequentially and can be recognized on board the UAV. On the one hand, by flipping the patterns,

errors occurring due to the detection of similarly looking natural structures should be avoided in future, because the system expects a regular change in the detected area. On the other hand, dedicated information will be linked to the single symbols. Orders or important information, as for example the current wind direction or a possible movement of the ground platform, can be encoded and transferred using the pattern sequences.

Therefore, the palette of used symbols was complemented with additional signs to extend the capability of encoding more complex information into a pattern sequence by switching between the introduced signs. Nevertheless, the used pattern pool is held small at the present time, because for every new introduced pattern the algorithm needs to be adapted in order to "learn" the new shape and to recognize it during the detection sequence. Additionally, an enlargement of the pattern pool also requires more logical operations during the scan process of possible pattern blobs found in the images, which leads directly to an enlargement of process time and workload during the classification of the pattern in flight. It remains to optimize the balance between size of the pattern pool (for information encoding) and duration of the pattern classification process.

V. ADAPTIVE PATTERN DEVELOPMENT

The currently used setup for development, evaluation and demonstration of the conceptual design was based on different simple pattern projectors to evaluate the concept and its functionality. The identified technologies that can be used to set up a working pattern projector needed to be consolidated in order to create a more flexible, adaptable test bed. The central object for further development is therefore the technological realization of the dynamic ground platform to create a complete working prototype, which will be integrated into the AMFIS communication backbone for information exchange and to receive control commands from the system in the future (see Figure 3).

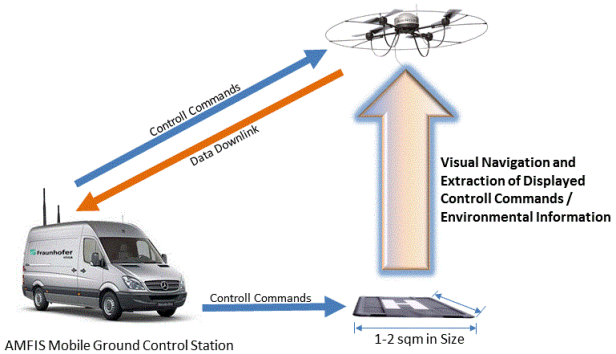


Figure 3. Sktech of the final target system.

For the initial non-dynamic testing of the algorithm, a static ground pattern with the shape of a white "H" on a black background was used. This test setup was designed to experimentally deploy the developed algorithm in a realistic test scenario under real conditions and environmental factors (e.g., sunshine). However, on account of the long-term aim of developing and applying a dynamic pattern, the adaptability and expandability of the detection and the interpretation algorithms was emphasized. Hence, the developed dynamic pattern should show the same static pattern (a white sign on black background) as exactly as possible to achieve the highest possible contrast in the first experiments.

Because the detection should be functional under bad lighting conditions and the missing possibility to introduce new or adapted patterns in the future, a mechanical solution with flipping parts was excluded. It has been assumed that the final working system could need an extension on the pattern alphabet or a change within the available patterns when new demands arise. A simple solution to display different symbols or patterns in different representations and scaling needed to be found. To cope with this, different Light-Emitting Diode (LED) matrices were examined and tested for their suitability.

The experimental used technologies for dynamic ground patterns are all slightly different in technology and size. The originally used prototype based on single low cost LED panels and reached a size of 65 x 65centimeter. Tested under realistic conditions, it shaped up that the low cost image display matrix, which provides control over every single LED, is not suitable on account of the used Pulse Duration Modulation (PDM) and the low fixed refresh rate. The PDM controlled LED cause a flickering not visible for the human eye, but for the camera. Experiments showed that this flickering troubles the algorithm in detecting possible blobs for the pattern in the video.

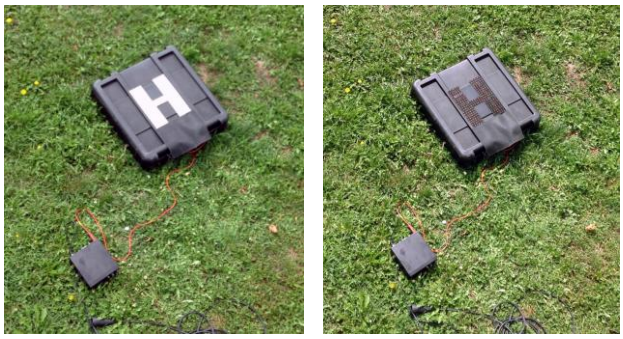


Figure 4. Illuminated and non-illuminated ground pattern.

To reach a non-flickering representation, small 3x3 illumination LED matrices were used and assembled to an 18x21 experimental matrix even smaller than the original test system (see Figure 4). This pattern matrix turned out to be absolutely flickering free and can, therefore, be detected by the algorithm as one structure without any problems. The second advantage is that the assembled platform was luminous strong and provided the capability to see and detect the ground pattern even in bright sun light.

The functionality of these different projection technologies were tested under different circumstances. In [1] it was shown that the developed algorithms in combination with the two described technological diverse pattern projectors are applicable for pattern supported navigation. Nevertheless, the validation of the overall concept for a final pattern projection technology is a central precondition for further advancements. Because different draughts for pattern projectors were pursued it is important to consolidate this technology and to transfer the knowledge from the validation process into a final technical draught. Based on the results of the present technological experiments two technology demonstrators were developed and tested. Both systems are based on matrix LEDs that can project different patterns. Indeed, they differ in the way the representation of the single patterns are generated as well as in their technical construction.

A. *Large Pixel Pattern Projector (L3P)*

The L3P (see Figure 5) is based on technical specifications of the 3x3 LED illumination matrixes also facilitated in the projector in Figure 4. The main difference to other tested matrixes is that the control of single LEDs to visualize certain forms or pictures is not their scope of application, but a constant full-area backlight illumination.

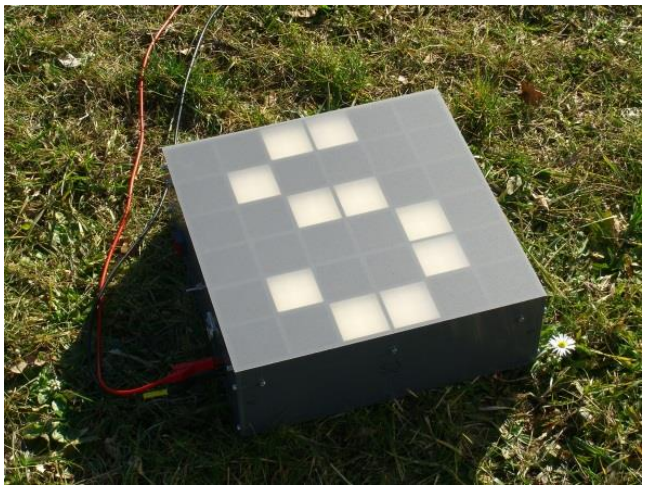


Figure 5. Large Pixel Pattern Projector (L3P).

The single modules are equipped with 9 LEDs, which can be either fully activated or deactivated. Based only on this technology a true-dynamic pattern projector cannot be realized. Hence, for the active pattern a projector module was developed, which includes several of the lighting modules, which can be switched on or off computer-controlled. The so designed pattern module consists of a total of 36 lighting modules and permits all possible permutations of illuminated and deactivated light fields controlled by the integrated hardware. Every single light field is separated with footbridges from the neighboring fields to allow a clean, sharp-edged projection. The projection screen is concluded with a diffusor, which compensates the relatively big distance between the single LEDs and prevents the covering of partial LED segments when the approach angles of an UAV are getting sharper.

In contrast to a fully adaptable pattern projector the ability of scaling the image is decreased by the size of the single pixels and the interconnected low overall resolution. On the other hand, originating from the diffusor and the size of the single pixels, it was assumed that less detection problems will arise during final or low flight phases.

B. *Flexible Advanced Pattern Projector (FlAPP)*

Beside design and construction of the L3P a second solution for a fully adaptable projection technology was developed. In opposite to the reduced scaling capabilities of the L3P the FlAPP should provide a high flexible pattern projection. An exact control of single LEDs is essential for a visualization of patterns in different scaling. For this purpose different high end LED panels were examined. As a main problem, on this occasion, it turned out that most LED screens suffer from a too low refresh rate. As a result of the flickering representations of the patterns the algorithms could not recognize the content and, hence, failed.

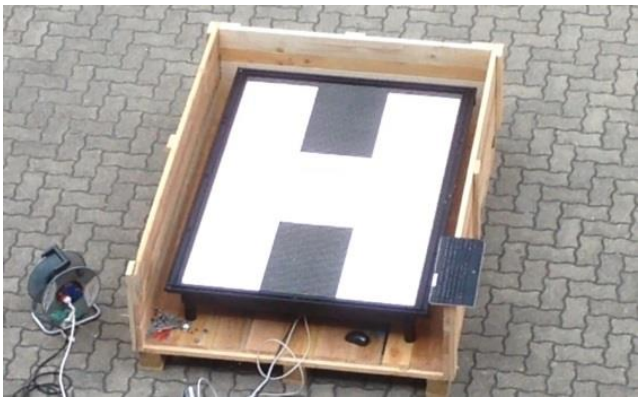


Figure 6. Flexible Advanced Pattern Projector (FlAPP).

The FlAPP was conceived as a LED panel build from SMD LEDs, which refresh rates were heavily raised with additional LED control technology to eliminate these problems.

VI. ADAPTIVE PATTERN ASSESSMENT

To further improve the development of a test system for the pattern-recognition-supported precision landing, the L3P and FlAPP had to be comparatively tested. By these tests under equal conditions both technologies become comparable to each other and can support a final technology decision or lead to a new development cycle to improve the test bed. Both draughts have their advantages and disadvantages, which were known partially in advance or were discovered in the draught-related test studies.

The L3P distinguishes itself by high contrast and angle independence by the accordingly scattering diffusor. However, it is limited in its scaling possibilities because the

single pixels cannot fall short of a minimum of 45 x 45 millimeter design dependent. Therefore, the resolution is low with ca. 386 pixels per square meter. The big advantages are the directly supplied LED modules, which are not controlled by a cyclic refresh process and provide a non-flickering representation independent of the used camera system. However, in comparison the FlAPP provides by its more than 40-times higher pixel density of 15,683 pixels per square meter a better adaptability in the representation of single patterns and also in their scaling. In addition, the available test system is equipped with RGB pixels, so that test series in different frequency bands of the visible light spectrum become possible.

For the comparison tests of the developed pattern systems the demands for a functional projector were gathered and realistic scenarios within the scope of the common takeoff and landing routines were extracted.

Regardless of the type of control (manually or computer-controlled), the approach on the landing position in present flights occur accordingly to the same workflow. The UAV stays on a safe flight altitude, which can be assumed to be free from any obstacles within the operation area.

If emergency procedures after a communication loss or a low energy alarm are disregarded, the UAV returns for landing to its starting point or another geographical position specified by the user. If the UAV has reached its landing position on a safe flight altitude, the pilot or the computer reduces the thrust and the UAV is approaching the ground. None or only GPS based course corrections are occurring in the computer-controlled mode, while a manual flying pilot can adapt the descent in speed as well as in horizontal direction to provide a safe landing. Hence, the direct vertical approach to the ground pattern arises as a primary test scenario.

Beside the recognition of the pattern, the scalability of the patterns is one essential factor to be tested. Dependent on the selected EO sensors the minimum size of the projected pattern in different distances has to be determined in order to selcet a suitable scaling.

Beside the maximum height or distance between sensor and projector, the minimal possible distance is of big relevance. Due to the used algorithm the projected shape of the pattern must have an interconnected structure. If the shape falls into pieces because of a too big pixel distance, the algorithm cannot recognize the pattern anymore and the pattern detection fails. This happens because of the adaptive threshold operation when the algorithm is searching for possible pattern blobs in the image. If parts of the pattern are disconnected to the rest, they will be detected as stand-alone-blobs. The detection tasks will try to recognize them and will fail. Particularly for the application of the FlAPP this problem is of central importance as a diffusor is absent and perhaps would have to be subsequently mounted to close possible appearing gaps at short distances between camera and projector. But also the L3P design has caused narrow dividing footbridges between the pixels that could limit the detection robustness.

Beside the primary task of validating the pattern technology concerning a functional direct vertical landing, the enlarged abilities of the draught are also to be examined. The above described scenario implies a low angle divergence during approach. However, if the possibilities of the used UAVs to adapt the optics horizontally as well as vertically are taken into account, sharper angles of approach need also to be considered. This scenario slightly adjusts the demands for the pattern technology concerning the homogeneous radiation of the LEDs or the diffusor. It was assumed that the L3P will provide a clearly steadier image projection on account of the diffusor whereas the FlAPP could suffer from color and intensity changes in different views. Hence, the experiments were extended to achieve a simple comparison between the projectors on account of different view angles. The perspective distortion of the patterns was neglected and is of minor importance as the algorithm is scale and rotation invariant.

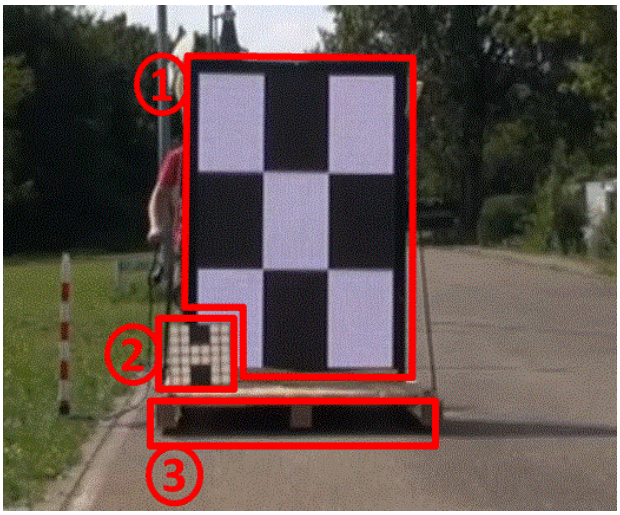


Figure 7. Test set up: FlAPP (1), L3P (2), mobile plattform (3).

In preceding test cases, enlarged flight experiments had already proved basic functionality of the concept facilitating illuminated but non-dynamic patterns. The knowledge and results from these experiments influenced the development of the L3P and the design of the FlAPP.

Particularly the development of a suitable diffusor that provides enough dispersion on the one hand and a low damping rate on the other hand, so that recognition is still possible under direct solar irradiation, is decisive for the functional L3P.

All initial test series were conducted under the premise of realistic application surroundings. Therefore, the pattern projectors were installed horizontally on the ground. All test recordings were done on board of a UAV with direct solar irradiation on the pattern. This modus operandi allowed checking and validating the design and functionality of the approach (see Figure 1).

The subsequent test series were focused on the applicability of the selected cameras as well as on the evaluation of the different pattern projector technologies.

In order to be able to compare the well-chosen electro-optical sensors, the image recordings must be done at the same time from the same position in identical distance and

lighting conditions. As the UAV chosen as target platform is not capable of carrying all cameras and their recording equipment, the experimental set-up was transferred for simplicity reasons from a vertical test bed into a horizontal one.

For this purpose the FlAPP was installed upright on a mobile platform. This was not necessary for the L3P, because it can be moved easily from hand due to its small size and weight (see Figure 7: FlAPP (1), L3P (2), mobile platform (3)). A test track of a maximum of 50 meter in length was set up where the FlAPP as well as the L3P were recorded by the different electro-optical sensors in different distances. Beneath the direct view at the sensor, additional approach angles to the pattern projector were simulated by panning the mobile platform.

VII. RESULTS

Essential topics for the further advancement of the technology could be identified by the evaluation of the test series and the recorded data. The functional limits defined by design referred to an operational distance of 0 to 100 meters between projector and image sensor. The tested set up covered a maximum distance of 50 meters, the results for distances beyond 50 meters where calculated. For the distance tests the FlAPP was used as reference system because of its size and scalability.

Based on the acquired data the main restriction identified for the chosen approach is that the complete application range cannot be covered by a single camera system fix fixed optics under the addressed conditions. In average, using different image sensors and distances the pattern was recognized down to a lower border of 6% of the side lengths of the original image resolution. In dependence from sensor, optics and the size of the pattern, the possible maximum distances for a successful detection can be calculated therefore.

The scaling possibilities of the FlAPP allow adapting the pattern to the flight altitude of the approaching UAV. Particularly during deep flight phase this is vital, because it covers the most critical part of a final approach. Hence, within 0 – 5 meters above ground, special demands for the image sensor and the optics arise. Though, the pattern is reduced in size, however, for a successful detection it should not exceed 60% of the image until shortly before landing. A wide-angular optics is suitable particularly for the final flight phase. At heights of 20 meters and more above ground, these camera systems fail in delivering a suitable image for detecting the pattern. Hence, the application of a telephoto lens is unavoidable when the functionality should be also guaranteed in higher operation levels.

Based on the minimum side length of 6% and 60% as a maximum value, 10% and 50 % were used for the calculation of the final optics. The considered buffer should permit a safe detection even at the outer bounds of the specification.

A vario zoom optic is not always possible because of its weight and the low payload capacity of the UAV.

Based on the test results of different camera systems a camera of the company IDS-IMAGING, the UI-2230SE was selected for further testing. Based on the performance data the necessary focal length can be calculated. The sensor size of the UI-2230SE is 1/3" (*B*), 3.6 millimeter to 4.8 millimeter and 6.0 millimeter diagonal. Image distance (*b*), object distance (*g*), focal length (*f*) and object (*G*):

$$f' = \frac{g}{\frac{G}{B} + 1}$$

With the restriction of the minimum and maximum picture ratio a theoretical focal length of 24.4 millimeter arises for the distance up to 15 meter and 81.2 millimeter focal length for distances of 20 – 50 meter. A continuous coverage for 0 – 100 meter is not possible with these restrictions. Pushing it to the edge using 6% image cover as determined during the test, the full distance up to a flight altitude of 100 meter is covered.

TABLE I. FOCAL LENGTH FOR DISTANCE

Focal length	Distance						
	0 - 15	*15-30*	*30-45*	*45-60*	*60-75*	*75-100*	*100 +*
100.00							
80.00							
65.00							
50.00							
15.00							

Attempts in the infrared spectrum of light have proven that detection of the patterns is possible but not effective. As it has been expected, the radiation of the used LEDs in the infrared spectrum is near zero. Merely the up-warming electronic modules were recognized with a big delay. A change of the pattern projection needs therefore several minutes to become visible to the IR sensor. After switching off of the pattern the last indicated symbol is still detectable for some time. Using IR for the pattern projection is interesting but would need a complete redesign of the pattern projection technology. Available LED panels are equipped with LEDs for the visual spectrum of the light due to their application purposes. For a fully working IR panel the LEDs need to be changed into special LEDs emitting light in the infrared spectrum. Further experiments with IR are therefore expulsed.

The comparative test of the developed projectors L3P and FlAPP could be used to evaluate the basic design as well as the special stages of development. Besides, both pattern technologies could show their strength. However, the identification of possible weak spots and problems was important. As illustrated in Figure 8, unexpected side effects were detected on the FlAPP during the measuring campaign. Partly heavy Moiré effects could be observed in some recordings in dependence of the used camera, certain distances and view angles. Though the effects of the image

interferences turned to be acceptable to the algorithm, or could be removed by known procedures like, for example, a combination of image dilatation and image erosion, nevertheless, such effects need to be avoided if possible to provide a more robust detection and to keep the workload of the on-board hardware as low as possible.

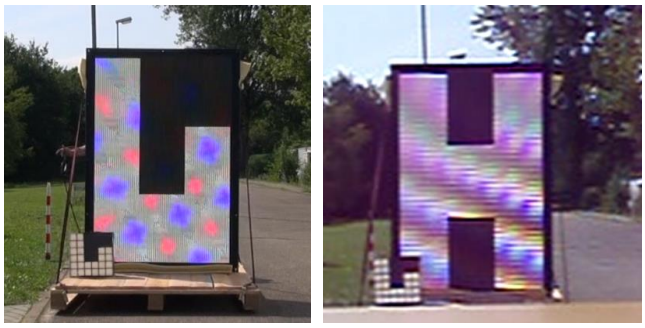


Figure 8. Moiré effects on the FlAPP.

We assume that the Moiré effects are originating from the overlapping of the matrix structure of the FlAPP by the matrix of the digital sensor. Therefore, the appearances of these effects are depending on distance and angle between camera and matrix LED. This phenomenon is strongly dependent to the combination of used image sensor, distance and angle. Hence, the appearance of such image interferences is difficult to avoid just by changing the sensor. The L3P does not show these effects on account of the fixed projection of the single large pixels and the distant mounted diffusor. Because of the pixel size and the steady light emission of the L3P the matrix is not filigree enough to generate Moiré effects by an overlapping with the raster of the image sensor. Tests have shown that the application of the same diffusor used on the L3P reduces the Moiré effects on the FlAPP to a minimum.

As expected, the scalability of the patterns proved to be the central functionality that can guarantee successful detection during the final landing approach. The L3P showed here its weaknesses, because the display of the pattern is of limited scalability. To deal with these problems, this technology requires the implementation of a special solution for the final approach sequence like introducing a new pattern consisting of a single white square (a single Pixel of the L3P when scaled to the minimum).

In addition to the internal factors, problems with the brightness of the projectors could be identified in the test. Originally it was assumed that detection problems will arise mainly in bright sunlight. The tests have not confirmed these concerns. But changing the conditions towards a poorer external lighting, some image sensors tend to catch a blurry representation of the pattern, especially at larger distances between projector and sensor. The pattern becomes indistinct to a single spot and thus cannot be detected anymore. The smaller the pattern (the greater the distance), the more intense is this effect, since fewer image pixels are accordingly covered by the pattern. At close range, this effect also occurs, but because the pattern is sufficiently large, the effect on the detection is low. The FLAPP is already equipped with an ambient sensor that can adjust the brightness to the external influence, but the sensor was not considered in the current test series. The L3P does not have such a sensor and therefore, needs to be upgraded.

Both pattern projectors have shown their strength and weaknesses during the test series. Based on the results the further development will focus on the application on the FlAPP as a final technology. But, because of its simplicity, the good handling and the low price, the L3P could also be updated and considered in future test set-ups.

VIII. Conclusion and Future Work

In this paper the activities of Fraunhofer IOSB in the area of civil security and their relevance for a supporting application in emergency situations were explained. For this work the applicability of small VTOL UAV systems to support rescue forces with local reconnaissance were brought into focus; the importance of a further improved automation was described. The essential restrictions of this technology for a realistic application concerning the critical flight phases of take-off and landing were discussed. As a solution for these problems the application of pattern recognition on board of an UAV in combination with a dynamic pattern projector on the ground was suggested. Besides, this works is built on diverging scientific research in the area of pattern based VTOL UAV landing, the essential difference is the introduction of a dynamic, adaptive ground pattern, which can visualize different patterns in different scaling. Therefore, central problems of pattern-supported navigation can be solved with the proposed approach. The likelihood of a false positive on the basis of natural structures similar to the pattern can be drastically lowered when a pattern is confirmed only within a detected structured sequence of different patterns. Missing the pattern in low flight altitudes due to dimension problems are avoided until the touch-down because the patterns can be adapted in their size according to the flight altitude of the UAV. In addition, the pattern sequences can be used for a low rate data exchange. Thus, relevant information can be transferred to the approaching UAV, for example, a divergence of the landing path or special alignments or course corrections.

To develop a dynamic pattern, different LED technologies were examined and checked on their applicability. The functionality of the draught was checked by successful system demonstrations. The identified functional LED technologies were further examined and two operational prototypes were developed for extended operational tests. These prototypes were operated in parallel and recorded with different IO sensors. On the set up test-range, sensors and projectors were evaluated in defined distances. Based on this data and the detection results, statements about the future technologies concerning cameras and ground pattern were made and necessary changes in the approach were identified. In particular, the quality increases by a distant mounted diffusor, as well as the better luminous

performance of the FlAPP will affect future works. Regardless of the pattern technology the detection algorithm is to be extended by the still missing pattern recognition for the new introduced patterns. Additionally, the development of a suitable pattern language as well as the safe ground pattern identification on base of pattern sequences has to be concluded.

ACKNOWLEDGMENT

The authors would like to thank their colleagues and students, who have contributed to the work presented in this paper.

REFERENCES

[1] F. Segor, C. K. Sung, R. Schoenbein, I. Tchouchenkov, and M. Kollmann, "Dynamic Pattern Utilization for Automatic UAV Control Support," The Ninth International Conference on Systems ICONS, pp. 140-144, 2014.

[2] S. Leuchter, T. Partmann, L. Berger, E. J. Blum, and R. Schönbein, "Karlsruhe generic agile ground station," Beyerer J. (ed.), Future Security, 2nd Security Research Conference, Fraunhofer Defense and Security Alliance, pp. 159-162, 2007.

[3] F. Segor, A. Bürkle, M. Kollmann, and R. Schönbein, "Instantaneous Autonomous Aerial Reconnaissance for Civil Applications - A UAV based approach to support security and rescue forces," The 6th International Conference on Systems ICONS, pp. 72-76, 2011.

[4] A. Bürkle, F. Segor, and M. Kollmann, "Towards Autonomous Micro UAV Swarms," Journal of Intelligent & Robotic Systems 61, pp. 339-353, 2011.

[5] E. Santamaria, F. Segor, I. Tchouchenkov, and R. Schönbein, "Path Planning for Rapid Aerial Mapping with Unmanned Aircraft Systems," The Eighth International Conference on Systems, pp. 82-87, 2013.

[6] C.-K. Sung and F. Segor, "Onboard pattern recognition for autonomous UAV landing," Proc. SPIE 8499, Applications of Digital Image Processing XXXV, 84991K, October 2012.

[7] S. Sharp, O. Shakernia, and S. Sastry, "A Vision System for Landing an Unmanned Aerial Vehicle," Proc. of IEEE International Conference on Robotics and Automation, pp. 1720-1728, 2001.

[8] S. Saripalli, "Vision-based Autonomous Landing of an Helicopter on a Moving Target," AIAA Guidance Navigation and Control Conference, August 2009.

[9] Y. Zhou, T. Wang, J. Liang, C. Wang, and Y. Zhang, "Structural target recognition algorithm for visual guidance of small unmanned helicopters," IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 908-913, December 2012.

[10] S. Yang, S. A. Scherer, and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," Journal of Intelligent and Robotic Systems 69(1-4), pp. 499-515, 2013.

[11] W. Xiang, Y. Cao, and Z. Wang, "Automatic take-off and landing of a quad-rotor flying robot," IEEE 24th Chinese Control and Decision Conference (CCDC), pp. 1251-1255, May 2012.

[12] S. Lange, N. Sünderhauf, and P. Protzel, "Autonomous Landing for a Multirotor UAV Using Vision," Workshop Proc. of SIMPAR 2008 International Conferrence on Simulation, Modeling and Programming for Autonomous Robots, pp. 482-491, 2008.

[13] T. S. Richardson, C. G. Jones, A. Likhoded, E. Sparks, A. Jordan, I. Cowling, and S. Willcox, "Automated Vision ‐ based Recovery of a Rotary Wing Unmanned Aerial Vehicle onto a Moving Platform," Journal of Field Robotics 2013, pp. 667-684, 2013.

[14] C.-K. Sung and F. Segor, "Adaptive Pattern for Autonomous UAV Guidance," Proc. SPIE 8856, Applications of Digital Image Processing XXXVI, 88560P, September 2013.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
issn: 1942-2679

International Journal On Advances in Internet Technology
issn: 1942-2652

International Journal On Advances in Life Sciences
issn: 1942-2660

International Journal On Advances in Networks and Services
issn: 1942-2644

International Journal On Advances in Security
issn: 1942-2636

International Journal On Advances in Software
issn: 1942-2628

International Journal On Advances in Systems and Measurements
issn: 1942-261x

International Journal On Advances in Telecommunications
issn: 1942-2601