The CSCT Living Lab for Computer Science and Computational Thinking

The “CSCT Living Lab” for Computer Science and Computational Thinking

Claudio Demartini, Fabrizio Lamberti, Marina Marchisio[1], Claudio Pardini[2], Amelio Patrucco[3]
Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino
claudio.demartini@polito.it, fabrizio.lamberti@polito.it
[1]Università di Torino, Via Carlo Alberto 10, 10124 Torino,
marina.marchisio@unito.it
[2]IIS Carlo Anti, Via Magenta 7, 37069 Villafranca di Verona,
dirigente@carloanti.it
[3]Fondazione Torino Wireless
Via Francesco Morosini 19, 10128 Torino
amelio.patrucco@torinowireless.it

Abstract *This work describes the “CSCT Living Lab”, where a new approach to learning, based on computational thinking and inquiry-based science, is practiced and shaped. Ten high schools, together with seven technical institutions and other three, chosen among vocational ones, have been involved. All these members, represented by teachers interested in joining the process, gave life to the lab, where enhanced scenarios for learning are planned, implemented and traced. An abstraction model is also proposed, to promote a collaborative learning context for teachers and students, derived from effective experiences carried out at University and Politecnico di Torino. A specific example, coming from process control domain, is described to show part of the scenario adopted to make inter-disciplinary learning a sustainable and effective experience.*

Keywords: Problem Posing&Solving, Computer Science, Computational Thinking, Process Control, IDEF0, Advanced Computing Environment

1. Introduction

The Project PP&S100 [Palumbo and Zich, 2012], Problem Posing & Solving, planned for 100 schools, grown to 150 in the meanwhile, is an initiative promoted by the General Directorate of the Italian Ministry of Education, Research and University, aimed at tracking innovations that have recently been introduced in the curricula adopted in school at the upper secondary level.

The purpose of this project, whose stakeholders include the Italian Association for Automatic Computing (AICA), the National Research Center (CNR), the National Industrial Unions, the University and the Politecnico di Torino, is to sustain the framework's tuning proposed at regulatory level,

pursuing the transition from "programs specified by the Ministry of Education" to national guidelines for high schools, technical institutions and vocational organizations complying with the EQF [Cedefop, 2012]. This change affects also teachers, who are now required to express a clearer role so that a more responsible and independent design of educational paths be possible for achieving learning outcomes that guidelines have established for the whole education system.

The project is focused on problem posing and solving and aims at sustaining computer science as an enabler of innovation in society and education. Computer science, by its own nature, has an immediate impact in any other scientific domain (physics, chemistry, natural sciences) and also brings a relevant potential within all other knowledge areas, even those placed in the socio-economic field, thanks to the basis provided by trans-disciplinary languages and information representations [Booch, 2013].

This already takes place in ordinary paths when regulations formally decree that mathematics and computer science must be joined together and combined when delivered in a class. Main drawbacks stem from the organization context, which lies on teachers, whose competence profile is often unavailable at school or, when present, unaware to face that kind of task, which requires clear specific skills and experience in the field.

2. Living Lab and Computational Thinking

To deal with that drawback, a "Living Lab" has been founded, which involves ten high school organizations, together with seven chosen from technical institutions and other three got among vocational ones. These institutional members, represented by teachers involved in the process, gave life to the lab, where a new approach to learning is planned and implemented.

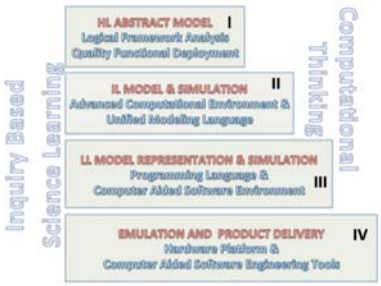

Fig. 1 – The Abstraction Hierarchy

Fig. 1 depicts the model drawn to introduce the rationale of the computer science discipline to be proposed in the secondary school, adopting as a background the "Inquiry based Science Learning model" and its specialization, represented by the "Computational Thinking" scenario [Wing, 2008][Yinnan and Chaosheng, 2012]. The first layer in the model hosts the highest abstraction process, where methodologies like the Logical Framework Analysis (LFA) [EU, 2004] investigate the essence of the problem, moving from instances promoted by the reference stakeholders, using brainstorm as a tool. They dig into the cause-effect relationships among the various facets that the complex structure of reality often shows, transforming it into the means-aims set in the solution domain. Given that, a further step leads to the release of the objective-purpose-results schema, which, at the end, brings to the activity scheduling and resource planning.

Still at the first layer (I), taking advantage of the "problem-posing" task, partly carried out in the previous phase, a further activity - rooted on a methodology which addresses issues related to the solution specification to be developed in the following layer (II) - can be found. To this purpose the Quality Function Deployment method, used to trace relationships between “customer voice” and “technical features” of the product/ service/ application, is suggested.

At the second layer, which is devoted to the solving perspective, target processes are first identified and analyzed to outline loosely dependent activities. The latter are given Specific functions in order to sustain process interactions, as planned by the IDEF0-based specification [IDEF0, 28/8/2013]. Once the functional model has been fully described, the next envisaged phase, in the same layer, concerns the Advanced Computation Environment (such as MAPLE or MATLAB [Chonacky and Winch, 2005]), where the IDEF model can be drawn to refine the mathematical perspective. Layer III and IV concerning language descriptions and prototype implementation will be dealt with in future works.

To validate the model, the Living LAB focused its attention first on a problem easily perceived by everybody and immediately understood in terms of effects, so that a reference framework could be shaped using available tools widely spread in the CS community. To this purpose a "process control system" was chosen and, in the following, a general abstraction model is shown taking advantage of the IDEF0 graphical notation.

3. Conceptual Modeling of a Process Control System

The choice of "*control theory*" [Doyle and Francis, 1990] is justified by the fact that any kind of process in real life is a subject of control. Even in sociological contexts this scenario happens normally, since control theory itself underlines how weak bonds between individuals and society make people free to go sometimes against the law, or other people having weak ties can engage themselves in crimes to gain benefit. This can be overcome when strong bonds make deviance more costly. Anyway to keep things simpler, a more technical domain has been taken as a stage to trace the example.

Fig. 2 depicts a controlled system taken in climate control domain. The “{I_{cs}| *Reference Temperature*}”, being I_{cs} the set of potential inputs for the process control system (CS meaning *Controlled System*), represents the target temperature the controlled system should hold. In other terms it should be the ideal temperature to be achieved within a {R_{cs}|*Room*}, being R_{cs} the set of potential supplies/resources for the whole process, even contrasting disturbances, {C_{cs}| *windows, doors*} - with C_{cs} representing the set of potential constraints to be applied to the process - which may occur, such as windows or doors left open or other accidents, able to induce a temperature change. The latter, in fact, can be somehow caught at the output, whenever the sampled temperature may not be coherent with the expected value given at the input. So to understand how a controlled system works, the hypothesis is that an appropriate solution should be able to perform repeated reading of the real

temperature achieved by the room under control. This means that the actual temperature should be taken by means of a suitable "{R_{cs} | *Sensor*}" placed somewhere in the "{R_{cs} | *Room*}" space. The picture gives more details about the sensor, able to catch the temperature, as "{O_{cs}| CV (*Controlled Variable*)}", and to bring it back towards the input. It is interesting to see the representation of the { R_{cs}|*Room*} assumed in the model description, which, in our case, is simply the room space together with its own heating system, { R_{cs} | *Heating System* }, without any control or regulation applied. Its input is labeled "{I_{ss}|MV (*Manipulated Variable*}" - being I_{ss} the set of inputs for the *Simple System* - and could be assigned, as a physical quantity, to the open angle of the valve regulating the fuel flow to the burner of the heating equipment. The abstraction concept represented here is that of "*composition association*" where a "*Controlled System*" is viewed as the composition of other subunits, the "Simple System" without control, which is the environment equipped with its own heating system, the "*Sensor*" to get samples of the temperature (output), and other units, all of them connected through arrows which represents specific information and/or material flows.

To give an example of the whole system, it is supposed to set the desired input, "{I_{cs}| *Reference Temperature*}", to k C° degrees. To get the output temperature, possibly aligned with the desired input specified above, the Controlled System should measure the current value of the output temperature "{ O_{cs} |CV}" shown by the simple system, constituted of a "{R_{cs}| Room}". A specific sensor, "{R_{cs}|*Sensor*}", transforms sampled temperatures into something else, easier to manage, such as a voltage signal. The higher the temperature, the higher the voltage brought back by the sensor to be used for other simple operations. To give an idea of the sensor sensitiveness it could be requested to equip the feedback line with a sensor able to convert a change of 1 C° of the sampled temperature into a variation of 0,01 V for the measured output.

The abstraction depicted graphically in Fig. 2 draws a picture of the model whose development is in progress. For the controlled system to do its job, it is needed to get the voltage fed back by the sensor compared against the desired temperature given as {I_{cs}| *Reference Temperature*}, in its turn translated into a voltage signal: this comparison can be performed using a simple "*Comparator*" able to detect the difference between the two physical quantities. The comparator provides the so-called "{E_{cs}|*Error Signal*}", able to give the "feeling" of the current difference between the desired input and the actual value experienced by the system to be controlled. The role of the error signal is fundamental, being it the real driver of the regulation/control action exercised on the simple system. The model drawing reported in the picture is completed with the last unit, which is the key to perform the regulation of the simple system (Room).

The picture makes it clear that the Controller takes the error signal as its input and transforms it into its output - actually the simple system input - labeled "{I_{ss}| *Manipulated Variable*}", which is the signal able to exercise the control on the simple system. In other words it takes the running difference between the

desired temperature and the actual temperature of the simple system, the error signal, to process it in order to shape an output signal able to act on the valve controlling the fuel to be provided to the burner. The higher the differences computed between the temperatures, the higher would be the signal used to change the position of the valve. Of course much more is required in terms of real equipment to be chosen to perform what the model suggests. As an example, if we suppose that a motor is used to exercise control on the open valve, or more simply tune the speed of a fan, attention should be paid to translate the output signal made available by the Controller, coping with the power needed to pilot the chosen actuator. In many cases the Controller is implemented in such a way that both the comparator and the Controller devices be joined in a single unit. According to this perspective, the controller is expected to carry out two main activities: computing the error signal and applying the computed control effort to the simple system.

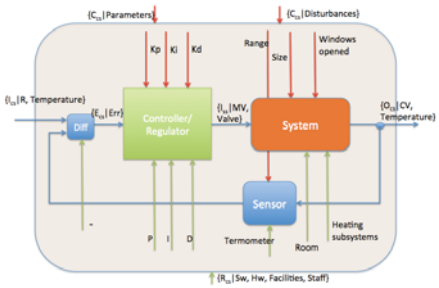

Fig. 2 - A Controlled System

Looking at the model it is quite immediate to realize that if the gain computed along the forward links - starting from the input error up to the output - is large enough, a small error experienced now might imply fairly large outputs. Let's imagine that, because of the controller, the valve be opened too much with respect to the computed error: as a consequence it is expected that a steep growth of the output temperature might occur. Hence, a strategy should be applied to the choice of the gain values, in fact it is needed to avoid unstable behavior of the Controlled System: in some cases the Controller itself, instead of exercising control, could become the cause of the simple system destruction. Anyway, there is a certain level of coherence with the choice of having high-level gains, since that makes it possible to get a fast recovery of the output with respect to very small errors experienced at the controller input.

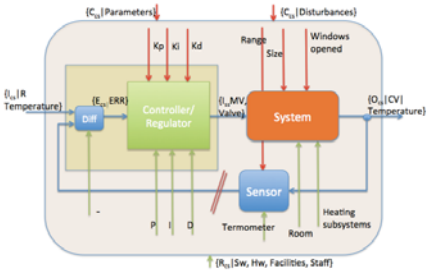

Fig. 3 - Closed and Open loop Control

3.1 Diving more into the Conceptual Model

The control action will be exercised either to vary the heat inflow or intervene on the leaks which determine the heat outflow from the simple system, this should be done to keep the temperature level at the set-point stated in the controlled system. Fig. 3 depicts the general schema of the model for achieving those objectives.

As it can be seen in the picture, the controlled system process can be

represented by a closed loop. A specific sensor monitors the simple system output, "*CV Temperature*", and the measured signal is fed-back to a comparator, represented by the block called *Diff*, sited close to the input of the controlled system. The second input provided to the comparator is the Reference (R) *Temperature*, being the comparator output the difference or error signal E computed on the two inputs. The regulator/controller, represented just as a black box, will provide the appropriate correction to maintain the process at its set point even in case of disturbances that may affect the heating system.

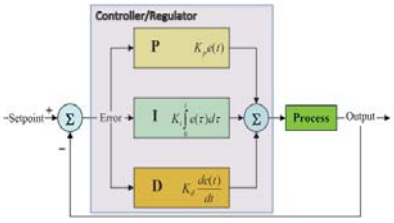

Fig. 4 - The PID Controller

The picture shows also that if the system were to be operated in manual control, placing an interruption symbol next to the sensor, the feedback path would be open as if it were not present. The operator would act in place of the feedback and apply necessary estimated corrections to the system while observing directly with his senses the effect on the controlled variable. This action is called *open loop operation*.

The best known form of control is given by the so-called "*PID controller*" [Åström and Hägglund, 1995], Proportional, Integral, Derivative control which, as depicted in Fig. 4, is based on the mathematical modeling of those three main actions, usually exercised intuitively in any ordinary individual experience when a behavior correction is required. As an example of this, it is immediate to recall what happens when a correction is to be performed by an exhausted driver as its car in a road is approaching the guardrail. At first the driver evaluates the relative position of the vehicle with respect to the guardrail and operates proportionally with the lateral distance using short turning of the steering wheel (proportional action). The proportional correction is usually applied repeatedly, in that performing what is called the integral action, which takes care of the previous history of the driver reaction. These corrections are also tuned by means of the speed applied to any single action, which should anticipate in short the next behavior of the car (derivative action).

Hence the effective calculation algorithm implemented in the Controller involves three separate parameters, as reported in Fig. 4: the proportional, the integral and the derivative values, denoted P, I, and D. Simply put, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. The weighted sum of these three actions is used to tune the process via a control element such as the position of a control valve, a damper, or the power supplied to a motor or a heating plant.

Given the absence of knowledge of the underlying system, a PID controller has long been considered as the best controller obtained by tuning the three parameters relevant in the PID controller algorithm, planned to devise control actions specifically designed for the identified process requirements. The features of the controller can be represented in terms of the responsiveness it

shows with respect to an error, given in terms of the degree with which the controller passes the indicated set-point, and the degree of oscillation which the system experiences at its output. On the base of this issue, it should be underlined that the PID algorithm does not guarantee optimal control of the system or even system stability.

3.2 Mathematical Perspective

The mathematical representation of the PID controller, in a form widely used in the industrial domain, is the so-called canonical form, where the Kp gain is applied to the integral and derivative terms, so that, being MV the manipulated variable, the following equivalence holds:

$$\mathrm{MV(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{d}{dt} e(t) \right)$$

where T_i is the integral time and T_d the derivative time. In this form, the parameters K_p, T_i and T_d have a precise physical meaning; in fact the inner summation computes a new error value, which is compensated taking care of past and future errors. In particular referring to the sum of the proportional and derivative components effectively predicts the error value looking at T_d seconds, or samples in the future. The integral component adjusts the error value to compensate for the sum of all past errors, by planning to cancel them in T_i seconds (or samples). Hence, the resulting compensated error value is scaled by the gain K_p.

There is also the so-called ideal parallel form in literature, given below:

$$\mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

where the relationship with the parameters shown in the previous one are the following: $K_i=K_p/T_i$ and $K_d= K_p*T_d$.

3.3 How can it be made understandable to young students?

The simplest way is given by the digital representation of phenomena where the analysis for designing a digital implementation of a PID controller in a microcontroller device requires the standard form be discretized. Approximations for first-order derivatives are made by backward finite differences. The integral term is discretized, with a sampling time Δt , as follows:

$$\int_0^{t_k} e(\tau)\, d\tau = \sum_{i=1}^{k} e(t_i) \Delta t$$

while the derivative term is approximated as:

$$\frac{de(t_k)}{dt} = \frac{e(t_k) - e(t_{k-1})}{\Delta t}$$

Thus, a velocity algorithm for implementing the discretized PID controller in a microcontroller is yielded by differentiating u(t) and also using the numerical definitions of the first and second derivative, then solving it for $u(t_k)$ getting the final result:

$$u(t_k) = u(t_{k-1}) + K_p \left[\left(1 + \frac{\Delta t}{T_i} + \frac{T_d}{\Delta t} \right) e(t_k) + \left(-1 - \frac{2T_d}{\Delta t} \right) e(t_{k-1}) + \frac{T_d}{\Delta t} e(t_{k-2}) \right]$$

The physical meaning of this last expression is easily understandable since it computes the new value for the manipulated variable, $u(t_k)$ as the old one, (u(tk-1)), compensated with the previous values of the error samples ($e(t_k)$, $e(t_{k-1})$, $e(t_{k-2})$, - got on a temporal window extended to two only previous samples, besides the current one -, suitably weighted, because of the derivative and integral contributions.

3.3 The System Model and its Simulation

Once a simple mathematical expression has been obtained to describe what seemed so complex at a first glance, the next step concerns the way a model can be perceived according to the typical sensing process human brain puts in place, whenever an abstract representation has to be validated. © MapleSim comes to the aid providing simulation scenarios on the basis of the many and various mathematical descriptions for several devices and phenomena made available by still growing and rich libraries.

MapleSim is a "multi-domain" system software simulator that allows analysts to create and simulate physical systems representations. Create a model means building the approximate description of a given physical system with a notation based on the combination of its main components, most of them often retrieved from already existing system libraries. That model is then able to generate its own mathematical representation in terms of a set of equations, solving that mathematical system, after proper simplification, leads to the definition of the physical quantities featuring the system in its temporal dynamics. MapleSim takes advantage of the © MAPLE [Chonacky and Winch, 2005] system software that provides a rich set of numerical and symbolic features becoming the real "engine" for any provided service. Maple and MapleSim together provide also support functionalities, such as graphical display of results, capability to extract specific analysis from mathematical descriptions such as, for example, other models given in a descriptive or programming languages such as C, further templates, and so forth. In particular MapleSim is able to describe and simulate systems having different physical nature: electrical, mechanical, thermal, hydraulic, to mention the most important. For this reason it is often referred to as a multi-domain environment.

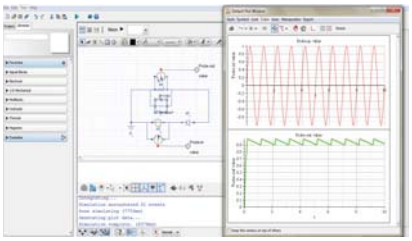

Fig. 5 - MapleSim user interface

Anyway, before deepening the description of the test-bed, a brief presentation of the user interface of the system is given, as depicted in Fig. 5, to underline the effectiveness of its organization, which makes interaction easy ensuring immediate understanding of operations. To this purpose it is organized into several sections, labeled panels, each devoted to a specific task and command. The central zone is the working area, where the analyst can build the model, taking the basic components from libraries organized by type. At the top are sets of controls grouped for handling model simulation providing a

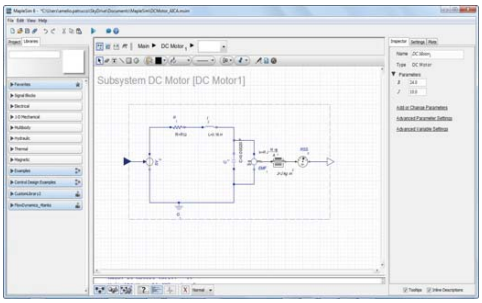

Fig. 6 – DC motor model in MapleSim

the results of a simulation.

Coming back to the construction phase of the control system model for a DC motor, the activity sequence starts with the description of the model for the engine to be controlled as shown in Fig. 6.

Fig. 7 reports a simulation result related

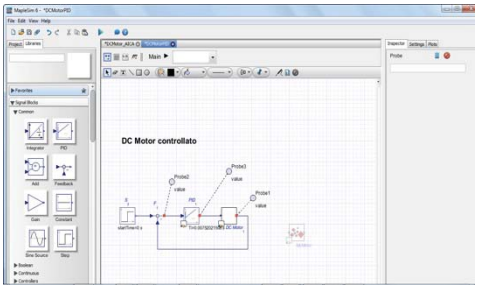

Fig. 8 - PID control for the DC motor

hierarchical navigation. Editing commands for the contents are also included: on the right side, in the figure covered with the window showing signals graphical images, the View menu together with a set of parameters associated with any component can be found. The window showing waveforms displays

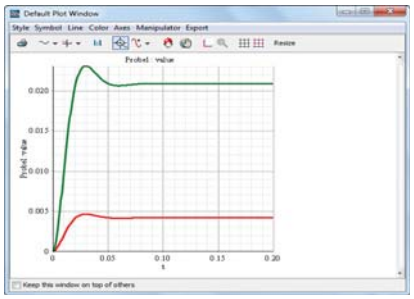

Fig. 7 - DC motor simulation results

to a small motor tested for two different values of power (1 and 5 volts respectively), in this specific case the angular velocity of the shaft is plotted in rad / sec.

The introduction of a PID control is carried through two steps as described in this example, the mathematical model of the motor is first linearized, then a control block, the PID controller, is built. Of course the linearization makes use once again of the link Maple-MapleSim through the creation of a specialized "template" for the generation of the model, Fig. 8 reports the result of this operation.

4. Conclusions

The “CSCT Living Lab” has been presented as a technological and social context where a new approach to learning is being experienced, working on the base of the conceptual stage offered by computational thinking and inquiry-based science. The Lab consists of teachers interested in joining the process gathered from ten high schools, seven technical institutes and three vocational ones, which have also been involved according to their role of institutions, by the Ministry itself. In the Lab all attendees, following a collaborative learning approach, promote enhanced scenarios for learning,

which should impact on 400 students for this year, laying the foundation to deploy the experience on traditional high school in the next years, increasing considerably the whole impact of the process.

An abstraction model was also proposed to help teachers and students share working contexts and experiences carried out at University and Politecnico di Torino.

The model was drawn to introduce the rationale of a new computer science discipline to be proposed in the secondary school, adopting as a background the “Inquiry based Science Learning model” and its specialization, the “Computational Thinking” scenario. A specific example, coming from process control domain, was also described to show part of the scenario adopted to make inter-disciplinary learning a sustainable and effective experience, so that the conceptual model could be validated.

The choice of "*control systems*" was justified by the fact that any kind of process in life is a subject of control. Anyway to keep things simpler, a more technical domain, related to climate control, has been selected as a stage to trace an example, which has been dealt with following the given abstract model and focusing mainly on its second layer, where more detailed specifications and simulations can be carried out using a set of suitable tools and methodologies.

References

[Palumbo and Zich, 2012] Palumbo, C., Zich, R., Matematica e informatica: costruire le basi di una nuova didattica, Bricks, Anno 2, n. 4, ISSN 2239-6187, 2012, pp. 10-19.

[Wing, 2008] Wing, J. M., Computational thinking and thinking about computing, Phil. Trans., R. Soc., 2008, 366, 3717-3725.

[Chonacky and Winch, 2005] Chonacky, N., Winch, D., 3MS for Instruction: MAPLE, MATHEMATICA, and MATLAB, Computing in Science & Engineering , Volume: 7 , Issue: 4 , July/August, 2005.

[Cedefop, 2012] Cedefop, Analysis and overview of NQF developments in European countries, Annual report 2012, © Cedefop, 2013.

[IDEF0, 28/8/2013] IDEF0, http://www.idef.com/IDEF0.htm, access: 38 August 2013.

[EU, 2004] EU Commission. 2004. *Project Cycle Management Guidelines.* pp 57-94.

[Doyle and Francis, 1990] Doyle, J., Francis, B., Allen Tannenbaum, Feedback Control Theory, Macmillan Publishing Co., 1990.

[Åström and Hägglund, 1995] Åström, K. J., Hägglund, T., PID Controllers: Theory, Design, and Tuning, ISA, Research Triangle Park, North Carolina, 1995.

[Booch, 2013] Booch G., From Minecraft to Minds, On Computing, IEEE Software, April, 2013.

[Yinnan and Chaosheng, 2012] Yinnan, Z., Chaosheng L., Training for Computational Thinking Capability on Programming Language Teaching, Computer Science & Education (ICCSE), 2012 7th Int. Conf., 14-17 July 2012, pp. 1804-09. ISBN: 978-1-4673-0241-8.