BIOMICS

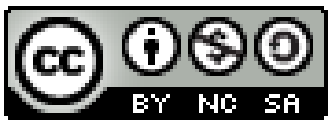
Proceedings of the 1st BIOMICS Summer Workshop

University of Debrecen, Hungary, 17-19 July 2013

Paolo Dini
Gábor Horváth
(Eds.)

SEVENTH FRAMEWORK PROGRAMME

This work was supported by the European Commission within the framework of the BIOMICS project, Contract Number EU-CNECT-FET-FP7-318202, `www.biomicsproject.eu`.



This work is licensed under a
Creative Commons
Attribution-NonCommercial-ShareAlike 3.0 Unported License

Table of Contents

Introduction 1

Systems Biology: Computing, Mathematics and Biology 3
Nicoline den Breems

Symmetry Analysis of Ordinary Differential Equations and Applications 4
Kyriakos Christodoulides

Lie Group Analysis of Two Biochemical Pathway Modules 5
Paolo Dini

Linearised Symmetry Condition for Systems of ODEs 23
Gábor Horváth and Zoltán Muzsnay

Classification of Semisimple Lie Algebras 32
Károly Podoski

Octonions 41
Ágota Figula

Groups Associated to Simple Lie Algebras 51
Zoltán Halasi

What Functors and Categories are Suitable for Relating Mathematical and Biological Models? . 60
Chrystopher L. Nehaniv, Maria Schilstra, Nicoline den Breems, Alastair Munro, and Daniel Schreckling

Algebra and Symmetry Structure in Discrete Models of Biochemical and Biological Systems: Natural Subsystems, Permutation Groups, and the Weak Control Hierarchy 61
Chrystopher L. Nehaniv, John Rhodes, Attila Egri-Nagy, Paolo Dini and Maria J. Schilstra

TECtonIc: Tick-Enforceable Control-Flow Integrity for Web-Applications 62
Eric Rothstein Morris

Holonomy Decomposition of the 3-Queens Puzzle 74
Eric Rothstein Morris, Paolo Dini, Chrystopher L. Nehaniv, Daniel Schreckling and Attila Egri-Nagy

Introduction

The BIOMICS project[1] is a 3-year STREP funded under the 7th Framework Programme of research of the European Commission, in the Unconventional Computation FET Proactive Initiative.[2] BIOMICS was inspired by the observation that cell metabolic/regulatory systems are able to self-organize and/or construct order dynamically, through random interactions between their components and based on a wide range of possible inputs. We are undertaking research into whether this behaviour can be reproduced in a controllable way through interacting finite-state automata. If so, this will lead to a radically new model of "bottom-up computation" with equal applicability to computer science and systems biology that we call Interaction Computing (IC).

Self-organizing behaviour in biological systems is a consequence of regularities of physical laws. The concept of regularity can be made more precise by relying on the mathematical concept of symmetry. Since the mathematical formalization of invariance is expressed through the language of algebra, IC research has greatly benefitted from algebraic automata theory. One of the aims of BIOMICS is to compare well-understood features of dynamical systems, such as their Lie group structure, to features of the algebraic structure of the automata derived from the same dynamical systems in the hope that the former will shed light on the latter.

The workshop that was held in July 2013 at the University of Debrecen enabled the discussion and comparison of various theoretical and applied research activities conducted during the first 9 months of the project, whose motivation and rationale can be deduced from the research framework shown diagrammatically in Figure 1.

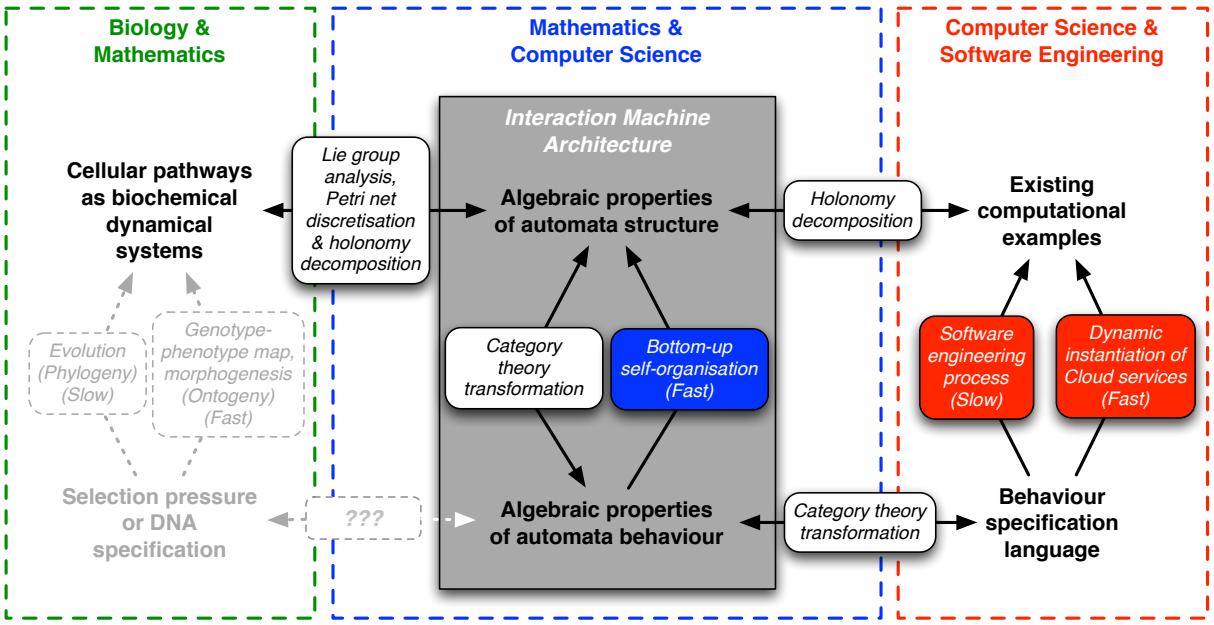


Fig. 1: **BIOMICS Research framework**

From this diagram it is clear that BIOMICS is a radically interdisciplinary project spanning very different research methologies, formalisms, and disciplinary "languages". A large part of the challenge, therefore, lies in developing the ability to communicate across disciplinary boundaries in order to

[1] Biological and Mathematical Basis of Interaction Computing. http://www.biomicsproject.eu
[2] http://cordis.europa.eu/fp7/ict/fet-proactive/ucomp_en.html

maximize the potential for each discipline to bring to bear its own analytical and creative skills on the conceptualization and formalization of Interaction Computing.

Accordingly, in this first project workshop each scientist was asked to present the activities they had been working on in the first 9 months of the project in such a way as to make the concepts, theories, and results as accessible as possible to scientists from other disciplines. For this reason, for example, no new mathematical results are included in these papers. The novelty, rather, is in combining a tutorial style generally reserved for elementary material to the explanation of fairly advanced concepts and theories in each research area – because the problems we need to solve in order to reach the BIOMICS objectives are far from elementary.

The prioritization of interdisciplinary communications over the reporting of new results made the peer review of the submissions less important, so in fact none of the submissions was referreed. Partly for this reason only seven of the eleven presentations were submitted as full papers. The remaining four were submitted as abstracts in order not to jeopardize their eligibility as new contributions to be submitted to relevant journals.

Paolo Dini

Royal Society Wolfson Biocomputation Research Lab
Centre for Computer Science and Informatics Research
University of Hertfordshire, Hatfield, United Kingdom
p.dini@herts.ac.uk

Gábor Horváth

Institute of Mathematics
University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
ghorvath@science.unideb.hu

Systems Biology: Computing, Mathematics and Biology

Nicoline den Breems

Medical Research Institute
Division of Population Health Sciences
University of Dundee, Dundee, United Kingdom
n.denbreems@dundee.ac.uk

Abstract. Biological functionality arises from the complex interactions of simple components. Traditionally a reductionist approach has led to identification of cellular and molecular components. Detailed cartoons like wiring diagrams present the interactions in signalling pathways and gene network regulation. More recently the rise of high throughput technologies and increasingly sophisticated computational tools have revealed an increasing regulatory complexity, many more interacting signalling pathways and interdependencies between components. With the increasing complexity verbal reasoning and simple cartoons can no longer explain the dynamics of the interactions. A systems biology approach combining computing, mathematics and biology is therefore needed to analyse the complex interactions and dynamics. Here I focus on in silico modelling using Ordinary Differential Equations for the analysis of signalling and gene network regulation.

Symmetry Analysis of Ordinary Differential Equations and Applications

Kyriakos Christodoulides

Institute of Mathematics
University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
christodoulides@science.unideb.hu

Abstract. Symmetry Analysis is a method which allows a systematic construction of exact solutions of differential equations. In this talk we will outline the underlying concepts and techniques utilised for solving first order ordinary differential equations (ODEs) and systems of first order ODEs. We will then proceed to exhibiting how these techniques can be applied in order to elucidate biological systems.

Lie Group Analysis of Two Biochemical Pathway Modules

Paolo Dini

Royal Society Wolfson Biocomputation Research Lab
Centre for Computer Science and Informatics Research
University of Hertfordshire, Hatfield, United Kingdom
p.dini@herts.ac.uk

Abstract. The paper discusses the analytical solution by Lie groups, and otherwise, of two cell regulatory/metabolic component systems. Only the first one is solved completely. For the second, a symmetry is found and the steps of the derivation are shown in detail. The level of detail and the kind of exposition followed in this paper are specifically meant to make the Lie group method accessible to non-mathematicians. Thus the emphasis is on showing the algorithmic details of the method and on helping the reader interpret their significance through visualizations. The paper shows the connection between Lie groups and the elementary method for solving ordinary differential equations (ODEs) known as 'separation of variables', and begins an in-depth discussion of the solution of more complex systems of two coupled 1st-order ODEs.

1 Introduction

In spite of the fact that the theory behind continuous symmetry methods based on Lie groups for solving ordinary differential equations (ODEs) is well understood (see for instance [8] in this volume, or [2,9,15,13,16,3,11,12,10,19]), no general method exists to *find* the symmetries that will actually solve a given ODE or a given system of ODEs. This situation obliges us to approach the analysis of systems inspired by cellular pathways – one of the tasks of the BIOMICS project – in a somewhat haphazard way and following a taxonomic approach.

A good starting point for our taxonomy is provided by the pathway modules proposed by Tyson et al. [17], which we have started to analyse in this manner already in [5]. As explained in [17], more realistic cellular pathways can be obtained by combining the basic 'building blocks' presented therein, a modelling approach that is becoming increasingly popular in systems biology in general [1]. As the complexity of the systems analysed increases, analytical methods alternative to the Lie group method become less effective or irrelevant. Although symmetry methods remain valid irrespective of the number and degree of non-linearity of the equations, their application in practical terms – i.e. the search for symmetries – becomes increasingly difficult. For this reason we have been trying to understand as much as possible what kinds of symmetries work for simpler systems, starting with those presented in [17], hoping that we may be able to generalize any insights to the more complex systems. When the closed-form solution can be obtained in some other way, this is discussed relative to the Lie group method, and when neither is possible numerical solutions are also discussed. The analytical and modelling choices made in this paper follow some of the work carried out at the University of Debrecen, in particular [8] and related still-unpublished work.

The level of detail and the kind of exposition followed in this paper are meant to make the Lie group method more accessible to non-mathematicians. So the emphasis is on showing the algorithmic details of the method and on helping the reader interpret their significance. Having said that, the algorithmic details by themselves are a bit dry and far from illuminating. Therefore, the non-mathematical readership is encouraged to consult the informal set of notes [4], which places a great emphasis on

motivation and visualization, as a tutorial to complement the other more standard references already cited.

In this paper we start with Case 1c from [17] and move on to Case 1d, which is the first 2-equation system considered in Tyson et al.'s paper. Because of its higher dimensionality, this latter problem requires somewhat more involved concepts. The process and ideas needed to *find* its symmetries are covered very well in [8], whereas the extension of the algorithm for *using* any such symmetries in higher-dimensional problems is not discussed in the simpler books on Lie groups (e.g. [2,10]) and will need to be developed in our future work specifically for the kinds of problems of interest to BIOMICS.

Case 1c models two coupled phosphorylation and dephosphorylation reactions by means of two enzymatic reactions based on Michaelis-Menten kinetics. The pathway is shown in Figure 1 and the resulting ordinary differential equation (ODE), fully derived in [5], is

$$\frac{dR_p}{dt} = \frac{k_1 S(R_T - R_p)}{k_{m_1} + R_T - R_p} - \frac{k_2 R_p}{k_{m_2} + R_p}, \tag{1}$$

where R is the protein being phosphorylated, R_p is the phosphorylated form of the protein, R_T is the total amount of protein, k_1 and k_2 are the two reaction rates in the forward and reverse directions, respectively, k_{m_1} and k_{m_2} are the Michaelis-Menten constants in the two directions and, as shown in [5], S is the ratio of the total amounts of the two enzymes catalyzing the two reactions. As this case is fairly involved from the algebraic and Lie groups points of view, it is better to analyse first the single-enzyme case, which we can solve completely. In this manner we can examine the full mathematical details of this kind of reaction, but with a minimum of clutter. We note that although the original

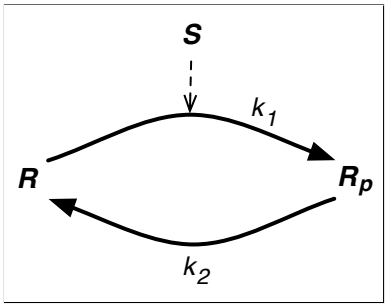


Fig. 1: **Phosphorylation and dephosphorylation pathway**

Michaelis-Menten saturated-enzyme assumption has been extensively critiqued and improvements to the model have been proposed (see for example [18]), for the purpose of understanding how to analyse and interpret dynamical behaviour the original model will do fine for now as a component in more complex pathways.

2 Analysis of Single-Enzyme Michaelis-Menten Kinetics

As shown in [5], the Michaelis-Menten ODE for the rate of depletion of the substrate under the quasi-steady-state approximation is

$$[\dot{R}] = -V_{max} \frac{[R]}{k_m + [R]}. \tag{2}$$

Schnell and Mendoza [14] show that this equation has the closed-form solution

$$R(t) = k_m \, W\left(\frac{R_T}{k_m} e^{\frac{R_T - V_{max} t}{k_m}} \right). \tag{3}$$

In this equation $W(\ )$ is the Lambert W function, which can be thought of as a generalization of the logarithm.[1] To see what this means, we first note that although the familiarity of the natural logarithm $\log(x)$ may induce us to think of it as an independent function, it is more properly defined as the inverse of the exponential function e^x. Formally, therefore, we can take the equation

$$e^{\log(x)} = x \tag{4}$$

as the implicit *definition* of the natural logarithm. In an entirely analogous manner $W(x)$ is defined as

$$W(x)e^{W(x)} = x. \tag{5}$$

This means that $W(x)$ works in a way similar to an exponent, implying that when the argument x is a large positive number we expect $W(x)$ to be a relatively small positive number (even smaller than $\log(x)$ in fact). The argument of W in Eq. 3 is a complicated-looking function involving a decaying exponential, but no matter: using *Mathematica*, in which $W(x)$ is called `ProductLog[x]`, this function is easily plotted as shown in Figure 2. We now solve this ODE using Lie groups and then show how the same analytical solution can also be obtained in a few simple steps that do not involve Lie groups.

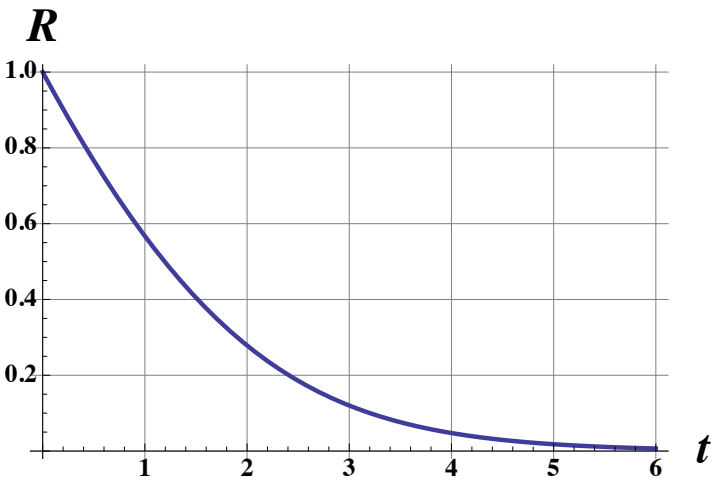


Fig. 2: **Lambert W function, Eq. (3), as the analytical solution to Eq. (2) for $V_{max} = R_T = k_m = 1$**

2.1 Lie Group Analysis

In order to gain insight into how Eq. (1) might be solvable, and in particular how it might be solvable by Lie groups, we first apply the Lie group method to its simpler version (2) expressed in the following form:

$$\frac{dy}{dx} = -V_{max}\frac{y}{k_m + y}, \tag{6}$$

where we have set $R = y$ and $t = x$. Details of the Lie group method can be found in [5,4,10,11] and references therein, so here we assume that the reader is already familiar with the method for 1st-order ODEs, at least in broad outline. Our concern in this paper is more to investigate what the method tells us about the problems being analysed in addition to helping us find their analytical solutions.

For 1st-order systems, the Taylor series form of a Lie point symmetry to order ϵ is

$$\begin{aligned} \hat{x}(x, y; \epsilon) &= x + \epsilon\xi(x, y) \\ \hat{y}(x, y; \epsilon) &= y + \epsilon\eta(x, y). \end{aligned} \tag{7}$$

[1] See [7] for an entertaining discussion of this interesting function.

Out of the many functional forms we could assume for the vector field components we choose a particularly simple form here:

$$\begin{aligned}\xi(x,y) &= a_0 + a_1 x + a_2 y \\ \eta(x,y) &= b_0 + b_1 x + b_2 y.\end{aligned} \tag{8}$$

Substituting these expressions into the linearized symmetry condition (LSC)

$$\eta_x + \omega(\eta_y - \xi_x) - \omega^2 \xi_y = \omega_x \xi + \omega_y \eta, \tag{9}$$

where $\omega(x,y)$ is given by the right-hand side of Eq. (6), gives

$$\begin{aligned} b_1 - \frac{V_{max} y}{k_m + y}(b_2 - a_1) - \frac{V_{max}^2 y^2}{(k_m+y)^2} a_2 &= -\frac{V_{max}(k_m + y) - y}{(k_m+y)^2}(b_0 + b_1 x + b_2 y) \\ b_1(k_m+y)^2 - V_{max} y(k_m+y)(b_2 - a_1) - a_2 V_{max}^2 y^2 &= -(b_0 + b_1 x + b_2 y) k_m V_{max}.\end{aligned}$$

Simplifying further,

$$\begin{aligned} b_1(k_m^2 + 2k_m y + y^2) - (k_m V_{max} y + V_{max} y^2)(b_2 - a_1) - \\ V_{max}^2 a_2 y^2 + k_m V_{max} b_0 + k_m V_{max} b_1 x + k_m V_{max} b_2 y = 0 \\ k_m^2 b_1 + 2k_m b_1 y + b_1 y^2 - (k_m V_{max} b_2 y - k_m V_{max} a_1 y + V_{max} b_2 y^2 - V_{max} a_1 y^2) - \\ V_{max}^2 a_2 y^2 + k_m V_{max} b_0 + k_m V_{max} b_1 x + k_m V_{max} b_2 y = 0 \\ (k_m^2 b_1 + k_m V_{max} b_0) + \\ (k_m V_{max} b_1) x + \\ (2k_m b_1 - k_m V_{max} b_2 + k_m V_{max} a_1 + k_m V_{max} b_2) y + \\ (b_1 - V_{max} b_2 + V_{max} a_1 - V_{max}^2 a_2) y^2 = 0. \end{aligned} \tag{10}$$

Setting the coefficients of the various powers of the dependent and independent variables in the last equation equal to zero[2] enables us to solve for the undetermined constants we assumed in Eq. (8):

$$\begin{aligned} x^0, y^0 : \quad & k_m^2 b_1 + k_m b_0 = 0 \quad \rightarrow \quad b_0 = -k_m b_1 \\ x^1 : \quad & k_m b_1 = 0 \quad \rightarrow \quad b_1 = 0 \quad \rightarrow \quad b_0 = 0 \\ y^1 : \quad & k_m V_{max} a_1 = 0 \quad \rightarrow \quad a_1 = 0 \\ y^2 : \quad & -V_{max} b_2 - V_{max}^2 a_2 = 0 \quad \rightarrow \quad b_2 = -V_{max} a_2 \\ \Rightarrow \quad & a_0, a_2 = \text{arbitrary}. \end{aligned}$$

If we let $a_0 = k_m$ and $a_2 = 1$, then $\xi(x,y) = k_m + y$ and $\eta(x,y) = -V_{max} y$, such that the auxiliary ODE becomes

$$\frac{dy}{dx} = \frac{\eta}{\xi} = -V_{max} \frac{y}{k_m + y},$$

which is identical to the ODE we are trying to solve. So clearly this choice of arbitrary constants just gives us a trivial symmetry. Let's use instead $a_0 = 0$ and $a_2 = 1$, so that

$$\begin{aligned}\xi(x,y) &= y \\ \eta(x,y) &= -V_{max} y.\end{aligned} \tag{11}$$

Setting

$$\frac{d\hat{x}}{d\epsilon} = \xi(x,y) = \hat{y} \qquad \text{and} \qquad \frac{d\hat{y}}{d\epsilon} = \eta(x,y) = -V_{max}\hat{y}, \tag{12}$$

[2] Rather than an equation for the unknowns x and y, which could be satisfied by a particular (x,y) pair of values (another equation would be needed if that's what we were after), this is a condition that must hold for *any and all* values that x and y can assume. Therefore, the coefficients of each power of x and y must vanish identically and independently.

the equations for the symmetry orbits are easily found as

$$\hat{x}(x,y;\epsilon) = x + \frac{y}{V_{max}}(1 - e^{-V_{max}\epsilon}) \qquad \text{and} \qquad \hat{y}(x,y;\epsilon) = ye^{-V_{max}\epsilon}. \tag{13}$$

We need to express the symmetry orbits as an explicit function $\hat{y}(\hat{x})$ for the 3-D visualizations of the jet space and first integral, below. Thus, eliminating ϵ,

$$\hat{y}(\hat{x}) = -V_{max}\hat{x} + (V_{max}x + y), \tag{14}$$

where (x, y) is any fixed starting point of a given orbit. The auxiliary ODE is

$$\frac{dy}{dx} = -V_{max} \qquad \rightarrow \qquad y(x) = -V_{max}x + r(x,y), \tag{15}$$

where $r(x, y)$ generalizes the constant of integration, in the sense that it assumes different constant values for each solution curve of the ODE. Each solution curve can therefore be seen as the level set or contour curve of such a surface $r(x, y)$. Thus,

$$r(x,y) = y + V_{max}x. \tag{16}$$

We next find the other canonical coordinate s:

$$s = \int \frac{dx}{\xi(x, y(x,r))} = \int \frac{dx}{r - V_{max}x} = -\frac{1}{V_{max}} \log(r - V_{max}x); \tag{17}$$

hence,

$$s(x,y) = -\frac{1}{V_{max}} \log(y). \tag{18}$$

Inverting,

$$\begin{aligned} y(r,s) &= e^{-V_{max}s} \\ x(r,s) &= \frac{1}{V_{max}} \left(r - e^{-V_{max}s}\right). \end{aligned} \tag{19}$$

The partial derivatives are

$$y_r = 0 \qquad\qquad y_s = -V_{max}e^{-V_{max}s} \tag{20}$$

$$x_r = \frac{1}{V_{max}} \qquad\qquad x_s = e^{-V_{max}s}. \tag{21}$$

Using the chain rule to transform the derivative $\frac{dy}{dx}$, the original ODE (6) can be expressed in terms of the canonical variables (r, s) as

$$\frac{y_r + \frac{ds}{dr}y_s}{x_r + \frac{ds}{dr}x_s} = \omega(x(r,s), y(r,s)). \tag{22}$$

Substituting the expressions (20) and (21) into (22), we seek an ODE in r and s whose RHS is just a function of r:

$$\begin{aligned} \frac{0 - V_{max}e^{-V_{max}s}\frac{ds}{dr}}{\frac{1}{V_{max}} + e^{-V_{max}s}\frac{ds}{dr}} &= -V_{max}\frac{e^{-V_{max}s}}{k_m + e^{-V_{max}s}} \\ -V_{max}e^{-V_{max}s}\frac{ds}{dr}(k_m + e^{-V_{max}s}) &= -V_{max}e^{-V_{max}s}\left(\frac{1}{V_{max}} + e^{-V_{max}s}\frac{ds}{dr}\right) \\ k_m\frac{ds}{dr} + e^{-V_{max}s}\frac{ds}{dr} &= \frac{1}{V_{max}} + e^{-V_{max}s}\frac{ds}{dr} \\ \frac{ds}{dr} &= \frac{1}{k_m V_{max}}. \end{aligned} \tag{23}$$

The RHS is a constant, so that certainly qualifies as a $f(r)$. Integrating with respect to r and substituting from (16) and (18),

$$\begin{aligned} s(r) &= \frac{r}{k_m V_{max}} + C_1 \\ -\frac{1}{V_{max}} \log(y) &= \frac{y + V_{max} x}{k_m V_{max}} + C_1 \\ \log(y) &= -\frac{1}{k_m} y - \frac{V_{max}}{k_m} x - C_1 V_{max} \\ y(x) &= e^{-\frac{y(x)}{k_m}} e^{-\frac{V_{max}}{k_m} x} C_2 \\ y(x) e^{\frac{y(x)}{k_m}} &= C_2 e^{-\frac{V_{max}}{k_m} x}, \end{aligned}$$

where, since C_1 is an arbitrary constant of integration yet to be set by initial conditions, we can freely replace it with $C_2 = e^{C_1 V_{max}}$. Doing this again by setting $C_2 = C_3 k_m$,

$$\frac{y(x)}{k_m} e^{\frac{y(x)}{k_m}} = C_3 e^{-\frac{V_{max}}{k_m} x}, \tag{24}$$

such that, comparing with Eq. (5), it is clear that

$$\frac{y(x)}{k_m} = W\left(C_3 e^{-\frac{V_{max}}{k_m} x} \right). \tag{25}$$

Now we switch (x, y) back to (t, R) and impose the initial condition:

$$R(0) = k_m W(C_3) = R_T \qquad \rightarrow \qquad W(C_3) = \frac{R_T}{k_m} \qquad \Rightarrow \qquad C_3 = \frac{R_T}{k_m} e^{\frac{R_T}{k_m}} \tag{26}$$

Substituting back,

$$R(t) = k_m W\left(\frac{R_T}{k_m} e^{\frac{R_T}{k_m}} e^{-\frac{V_{max}}{k_m} t} \right) = \boxed{k_m W\left(\frac{R_T}{k_m} e^{\frac{R_T - V_{max} t}{k_m}} \right)}, \tag{27}$$

which is the stated solution (3). In a manner that is exactly analogous to how we derived Eq. (16), if we solve for C_3 in Eq. (24), we can regard each value of this constant of integration as defining a different level set of a 'first integral' surface $\phi(R, t)$:

$$\phi(R, t) = \frac{R(t)}{k_m} e^{\frac{R(t) + V_{max} t}{k_m}}. \tag{28}$$

Figure 3 shows the solution curves for two initial values of the substrate drawn on the two surfaces defined by the hull in the jet space, i.e. the RHS of (2), and the first integral just derived. The shapes of the surfaces are such that they are more easily viewed from different vantage points, but of course the projections of the solution curves and of the symmetry orbits (shown in red) onto the R-t plane are identical. The red arrows show the action of the symmetry curves given by Eq. (14). Of course, Eq. (14) is defined in the plane, so drawing the orbits on these two surfaces is achieved by a nice *Mathematica* trick,[3] so that for the first integral surface the arrows are merely a convenient visualization. For the hull, on the other hand, the orbits fall precisely on this surface. In fact, as discussed in [8], this is the geometrical interpretation of the LSC (9), i.e. it is by enforcing tangency to the hull that the symmetries are found.

What has the Lie group method told us that we did not know before? It has highlighted the fact that solution curves live on the surface shown on the left of Figure 3, and that symmetries of this ODE are tangent to this surface at every point. This is a nice geometrical insight, but in truth does not seem to warrant the amount of work we had to do to obtain the solution – especially if we can obtain it in a simpler way, as we now show.

[3] See the CurveGraphics6 Package by Gianluca Gorni at http://users.dimi.uniud.it/~gianluca.gorni/Analisi1/index.html

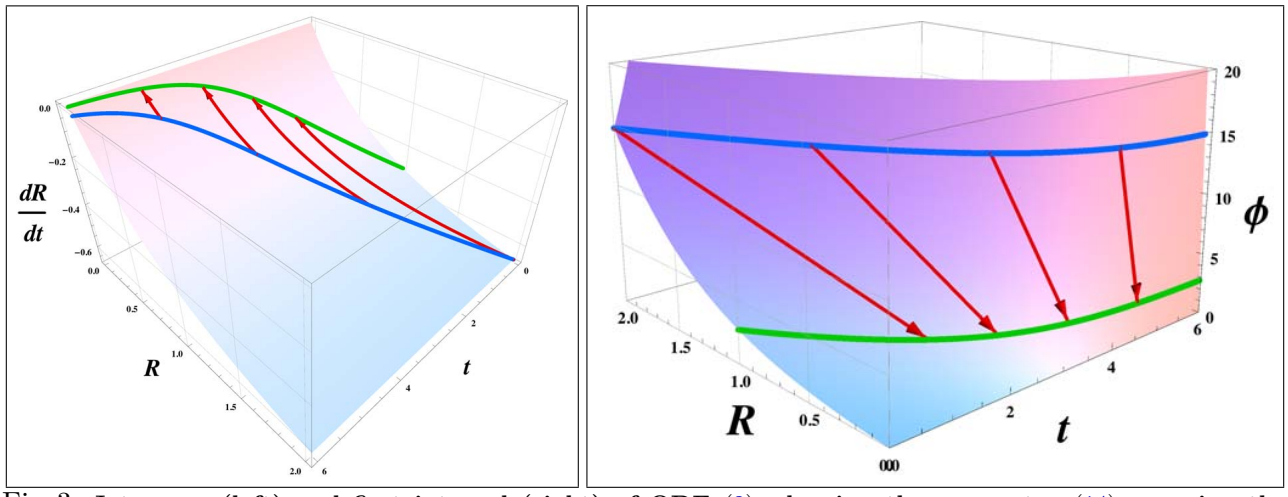


Fig. 3: **Jet space (left) and first integral (right) of ODE (2), showing the symmetry (14) mapping the solution for $R_T = 2$ to that for $R_T = 1$ ($V_{max} = k_m = 1$)**

2.2 A Simpler Way to Find the Analytical Solution, without Lie Groups

Although to some readers Eq. (2) will look immediately integrable by separation of variables, let's follow a slightly different presentation that will be justified in the next section. We take the reciprocal of (both sides of) Eq. (2):

$$\frac{dt}{dR} = -\frac{1}{V_{max}} \frac{k_m + R}{R}. \tag{29}$$

Integrating,

$$\begin{aligned} t(R) &= -\frac{1}{V_{max}} \int \left(\frac{k_m}{R} + 1 \right) dR \\ &= -\frac{1}{V_{max}} \left[k_m \log(R) + R \right] + C \end{aligned} \tag{30}$$

Inverting to solve for $R(t)$ we get

$$\log(R) + \frac{R}{k_m} = -\frac{V_{max} t}{k_m} + C$$

$$R(t) = C e^{\left(-\frac{R}{k_m} - \frac{V_{max} t}{k_m} \right)} \tag{31}$$

$$\frac{R(t)}{k_m} e^{\frac{R(t)}{k_m}} = C e^{-\frac{V_{max} t}{k_m}}, \tag{32}$$

where we did not show how the constant of integration C changed at each step since in any case it still needs to be set by the initial condition. As we said in the previous derivation, the above equation and Eq. (5), together, imply that

$$\frac{R(t)}{k_m} = W\left(C e^{-\frac{V_{max} t}{k_m}} \right). \tag{33}$$

Enforcing the initial condition exactly as we did in (26) leads again to the same solution (27). We now turn to the analysis of Eq. (1) to continue exploring the relationship between the method of separation of variables and Lie groups.

3 Case 1c: Phosphorylation and Dephosphorylation with Michaelis-Menten Kinetics

Applying a Lie group analysis to Eq. (1) led to the realization that any ODE of the form

$$\frac{dy}{dx} = \omega(y) \tag{34}$$

can be integrated immediately by taking the reciprocal, integrating, and then inverting the result of the integral:

$$\frac{dx}{dy} = \frac{1}{\omega(y)} \quad \rightarrow \quad x(y) = \int \frac{1}{\omega(y)} dy + C \quad \rightarrow \quad y(x) \tag{35}$$

Of course, in elementary textbooks on differential equations this is normally explained as 'separation of variables':

$$\frac{dy}{\omega(y)} = dx \quad \rightarrow \quad \int \frac{dy}{\omega(y)} = x + C \quad \rightarrow \quad y(x), \tag{36}$$

which is algebraically equivalent. The slightly different form shown in (35) arises from the use of canonical coordinates. Of course, in both cases the inversion of $x(y)$ to obtain $y(x)$ is not necessarily easy or even possible. We now demonstrate these points through the analysis of (1).

3.1 Lie Group Analysis

The size and complexity of Eq. (1) justify using the symbolic manipulation capability of *Mathematica*. We simplify the notation somewhat in order to make it easier to read and interpret the *Mathematica* output:

$$\text{Let} \qquad a = k_1 S, \quad b = k_{m_1} + R_T, \quad c = R_T, \quad d = k_2, \quad e = k_{m_2}, \quad x = t, \quad y = R_p. \tag{37}$$

Then, Eq. (1) becomes

$$\frac{dy}{dx} = \frac{a(c-y)}{b-y} - \frac{yd}{e+y}. \tag{38}$$

Assuming the same functional form for the infinitesimal vector field as for the previous problem, i.e. Eq. (8), the LSC (9) becomes

$$(a_1 - b_2)\left[\frac{a(c-y)}{b-y} - \frac{yd}{e+y}\right] + a_2\left[\frac{a(c-y)}{b-y} - \frac{yd}{e+y}\right]^2 + (b_0 + b_1 x + b_2 y)\left[\frac{a(c-y)}{(b-y)^2} - \frac{a}{b-y} - \frac{d}{e+y} + \frac{yd}{(e+y)^2}\right] = b_1 \tag{39}$$

Since x appears only once, we can set its coefficient $b_1 = 0$:

$$(a_1 - b_2)\left[\frac{a(c-y)}{b-y} - \frac{yd}{e+y}\right] + a_2\left[\frac{a(c-y)}{b-y} - \frac{yd}{e+y}\right]^2 + (b_2 y + b_0)\left[\frac{a(c-y)}{(b-y)^2} - \frac{a}{b-y} - \frac{d}{e+y} + \frac{yd}{(e+y)^2}\right] = 0. \tag{40}$$

We now expand, multiply by $(b-y)^2(e+y)^2$, and further simplify to collect the coefficients of the various powers of y:

$$\begin{aligned}
&y(2a^2a_2c^2e - 2a^2a_2ce^2 - a_1b^2de - 2aa_2bcde + 2ab_2ce^2 + 2aa_1bce - 2ab_2bce + \\
&\qquad 2ab_0ce - aa_1be^2 - 2ab_0be - aa_1ce^2 + 2b_0bde) + \\
&y^2(a_2a^2c^2 - 4a_2a^2ce + a_2a^2e^2 + a_2b^2d^2 - a_1b^2d - 2a_2abcd + 4ab_2ce + a_1abc + \\
&\qquad ab_0c - abb_2c + 2a_2abde + 2a_1bde - ab_2e^2 - 2a_1abe - abb_0 + \\
&\qquad 2a_2acde - 2a_1ace + a_1ae^2 + b^2b_2d - b_0de) + \\
&y^3(-2a_2a^2c + 2a_2a^2e + 2ab_2c - 2a_2bd^2 + 2a_2abd + 2a_1bd - 2ab_2e - a_1ab + \\
&\qquad 2a_2acd - a_1ac - 2a_2ade - a_1de + 2a_1ae - 2bb_2d) + \\
&\qquad y^4(a_2a^2 - ab_2 + a_2d^2 - 2a_2ad - a_1d + a_1a + b_2d) +
\end{aligned}$$

$$a^2a_2c^2e^2 + aa_1bce^2 + ab_0ce^2 - abb_2ce^2 - abb_0e^2 - b^2b_0de = 0. \tag{41}$$

We notice that this rather long equation does not involve a_0 at all. This is a direct consequence of the fact that in this problem $\omega(x, y)$ is actually only a function of y: thus, in the LSC (9) the only term multiplying ξ is zero. This will have an important consequence. Be that as it may, we proceed by setting the coefficients of each power of y equal to 0. This involves 5 equations (for y^0, y^1, y^2, y^3, and y^4) in 4 undetermined constants (a_1, a_2, b_0, and b_2, since a_0 is not fixed by these equations and is therefore arbitrary), so this is an over-determined system. The following are the 5 coefficients factored by *Mathematica*:

$$(a-d)[a_2(a-d) + a_1 - b_2] = 0 \tag{42}$$

$$a_1\{-[a(b+c-2e) + d(e-2b)]\} - 2[a_2(d-a) + b_2][a(e-c) + bd] = 0 \tag{43}$$

$$\begin{aligned} a_2[a^2(c^2 - 4ce + e^2) + 2ad(-bc + be + ce) + b^2d^2] + a_1[a(bc - 2be - 2ce + e^2) - \\ bd(b - 2e)] + 4ab_2ce - ab_2bc + ab_0c - ab_2e^2 - \\ ab_0b + b_2b^2d - b_0de = 0 \end{aligned} \tag{44}$$

$$\begin{aligned} -e\Big(a_1\{a[b(e-2c) + ce] + b^2d\} - 2\{aa_2c[a(c-e) - bd] + \\ b_0(-ab + ac + bd) + ab_2c(e-b)\}\Big) = 0 \end{aligned} \tag{45}$$

$$e(a^2a_2c^2e + aa_1bce - ab_2bce + ab_0ce - ab_0be - b_0b^2d) = 0 \tag{46}$$

We can now solve for the undetermined constants one-by-one. From Eq. (42), we have

$$a_1 = b_2 - a_2(a-d). \tag{47}$$

Substituting into (43) and simplifying, we get

$$(aa_2 - a_2d - b_2)[a(b-c) + de] = 0, \tag{48}$$

which implies that

$$a_2(a-d) - b_2 = -a_1 = 0. \tag{49}$$

Therefore, so far we have that

$$a_1 = 0 \tag{50}$$

$$b_2 = a_2(a-d). \tag{51}$$

Substituting these two values into Eqs. (44), (45), and (46),

$$a^2a_2c(c-b) + a\left\{a_2b^2d - b[a_2d(c-2e) + b_0] + a_2de(e-2c) + b_0c\right\} - b_0de = 0 \tag{52}$$

$$-2e\left[a^2a_2c(b-c) + a(a_2cde + bb_0 - b_0c) - bb_0d\right] = 0 \tag{53}$$

$$-e\left\{abe[a_2c(a-d) + b_0] - ace(aa_2c + b_0) + b^2b_0d\right\} = 0. \tag{54}$$

Solving for b_0 from (54),

$$b_0 = -a_2\frac{ace\,(ab - ac - bd)}{ae(b-c) + b^2d}. \tag{55}$$

At this point, if this expression is substituted for b_0 into either (52) or (53), in both cases we get that $a_2 = 0$. As a consequence, $b_0 = 0$ too. Although it is possible that this ODE may admit some other symmetry with a different functional form from Eq. (8), for now we have established that an arbitrary value of a_0 will not change the form of the differential equation or, equivalently, that it will map the hull in the jet space to itself. In other words, any time translation is a symmetry of this ODE.

Taking a step back to look at the bigger picture, this is not so surprising, in hindsight. Since $\omega(x,y)$ is actually only $\omega(y)$, clearly the hull is a surface that depends only on y and that is everywhere parallel to the x-axis, which is the time axis. Therefore, any translation in the x or t direction will map such a surface to itself. Let's complete the analysis so we can visualize all this. Since a_0 is arbitrary we set it equal to 1. Therefore,

$$\begin{aligned} \xi(x,y) &= 1 \\ \eta(x,y) &= 0. \end{aligned} \tag{56}$$

The auxiliary ODE, the level curves of whose first integral define the canonical coordinate $r(x,y)$, therefore, is

$$\frac{dy}{dx} = 0, \qquad \text{such that} \qquad y(x) = C = r(x,y). \tag{57}$$

In other words, $r(x,y) = y$, which means that the symmetry orbits are straight lines parallel to the x-axis, as we expected. We find s as before from

$$s = \int \frac{dx}{\xi(x, y(x,r))} = \int dx = x + C. \tag{58}$$

The canonical coordinates, therefore, are simply

$$\begin{aligned} r(x,y) &= y \\ s(x,y) &= x. \end{aligned} \tag{59}$$

As explained in [4], the original ODE (38) expressed in canonical coordinates is given by

$$\begin{aligned} \frac{ds}{dr} &= \frac{s_x + \omega(x(r,s), y(r,s)) s_y}{r_x + \omega(x(r,s), y(r,s)) r_y} \\ &= \frac{1 + \omega(x(r,s), y(r,s)) \cdot 0}{0 + \omega(x(r,s), y(r,s)) \cdot 1} = \frac{1}{\omega(x(r,s), y(r,s))} = \frac{1}{\frac{a(c-r)}{b-r} - \frac{rd}{e+r}} \end{aligned} \tag{60}$$

As expected, the RHS is a function only of r, meaning that we can integrate this directly to obtain $s(r)$. But, using (59), we can see that this is nothing more than the reciprocal of the original ODE:

$$\frac{dx}{dy} = \frac{1}{\frac{a(c-y)}{b-y} - \frac{yd}{e+y}}, \tag{61}$$

which is analogous to the method of separation of variables for 1st-order ODEs. We have therefore shown that this method can be "explained" or in any case justified in terms of a uniform time translation symmetry, which applies whenever $\omega(x,y) = \omega(y)$ only.

In practical terms, however, the solution is not yet within grasp because Eq. (60), although integrable, leads to a solution in the form $x(y)$, which corresponds to $t(R_p)$:

$$\begin{aligned} t(R_p) = &-\frac{R_p}{2(k_2 - k_1 S)} + \\ &\frac{k_1^2 k_{m_1} S^2 (k_{m_2} + R_T) - k_1 k_2 S \left[k_{m_1}^2 + k_{m_1}(4k_{m_2} + R_T) + k_{m_2}(k_{m_2} + R_T)\right] + k_2^2 k_{m_2}(k_{m_1} + R_T)}{(k_2 - k_1 S)^2 \sqrt{-k_1^2 S^2 (k_{m_2} + R_T)^2 + 2k_1 k_2 S[k_{m_1}(R_T - k_{m_2}) + R_T(k_{m_2} + R_T)] - k_2^2 (k_{m_1} + R_T)^2}} \\ &\tan^{-1}\left[\frac{k_1 S(-k_{m_2} - 2R_p + R_T) - k_2(k_{m_1} - 2R_p + R_T)}{\sqrt{-k_1^2 S^2 (k_{m_2} + R_T)^2 + 2k_1 k_2 S[k_{m_1}(R_T - k_{m_2}) + R_T(k_{m_2} + R_T)] - k_2^2 (k_{m_1} + R_T)^2}}\right] - \\ &\frac{(k_1 k_{m_1} S + k_2 k_{m_2})}{2(k_2 - k_1 S)^2} \log\left[k_2 R_p(-k_{m_1} + R_p - R_T) - k_1 S(k_{m_2} + R_p)(R_p - R_T)\right]. \end{aligned} \tag{62}$$

To obtain the solution $R_p(t)$ this function needs to be inverted, which does not appear to be possible in this case. Therefore, having gained a better understanding of what kind of Lie symmetry can solve this kind of ODE, we might as well revert to a numerical solution since the above function could only be inverted numerically in any case.

3.2 Numerical Solution

Figure 4 shows the time evolution of the concentration of the phosphorylated protein for different values of S and for different values of the Michaelis-Menten constants k_{m_1} and k_{m_2}. As discussed by Goldbeter and Koshland [6], small values for these constants correspond to both enzymes being saturated, which leads to very abrupt switch-like behaviour. When the enzymes are not saturated, on the other hand, the steady-state concentrations of the phosphorylated and unphosphorylated protein change much more slowly as the relative total concentration of the two enzymes (the value of S) is varied. As shown in detail in [5], setting Eq. (1) equal to zero we can derive a steady-state expression

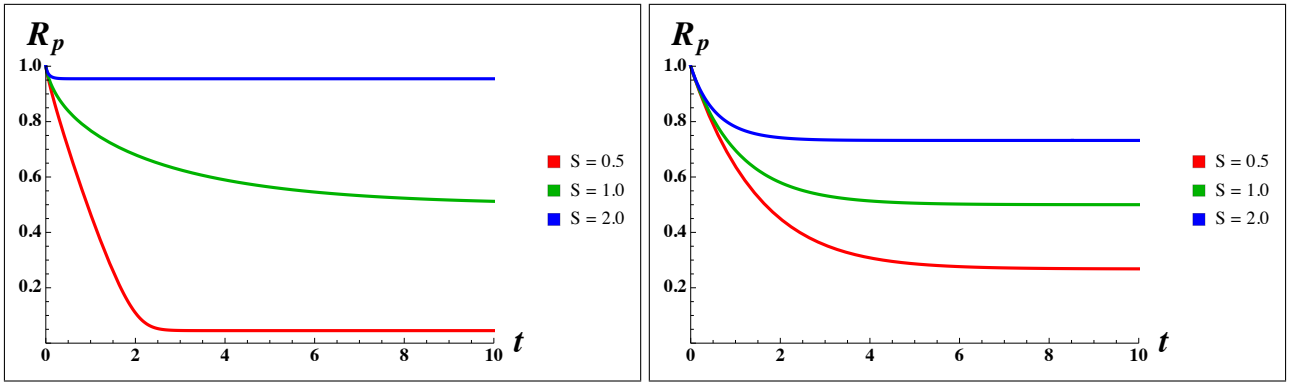


Fig. 4: **Numerical solution for the Case 1c system (Eq. (1)) and three values of S. Left: the values of the physical constants are taken from [17]: $R_T = 1$, $k_{m_1} = k_{m_2} = 0.05$, $k_1 = k_2 = 1$. Right: here we follow [6] to show the much softer switch behaviour when $k_{m_1} = k_{m_2} = 1.0$**

for R_p as a function of S. This is shown in Figure 5 for the same two cases shown in Figure 4. We will keep this behaviour in mind when we start combining different modules to construct more complex pathways. For example, a pathway that synthesizes one of the two enzymes could be coupled to this one as an additional equation for the first or second enzyme. As its concentration changed, S would change too, causing the phosphorylation of R to switch on or off, and the abruptness of the switch behaviour could be "tuned" using the Michaelis-Menten constants k_{m_1} and k_{m_2}. Before we can start approaching such a problem we need to understand how to solve systems with two or more ODEs. In the next section we take the first steps in this direction.

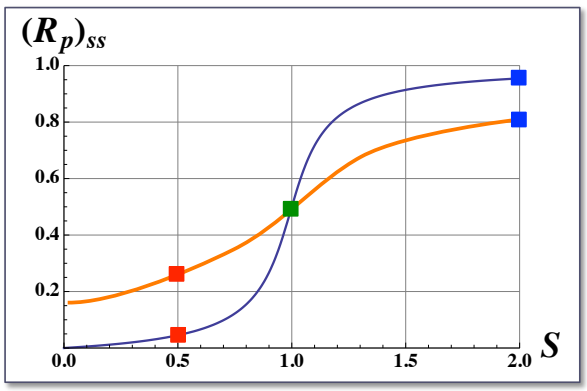


Fig. 5: **Steady-state solution for the Case 1c system (Eq. (1)) as a function of S. The two curves were obtained with the values of the Michaelis-Menten constants given in Figure 4. Also for the legend see Figure 4.**

4 Case 1d

The equations for Case 1d in [17] are

$$\begin{aligned} \frac{dR}{dt} &= k_1 S - k_2 X R \\ \frac{dX}{dt} &= k_3 S - k_4 X, \end{aligned} \tag{63}$$

and the corresponding pathway is shown in Figure 6. This system is called a 'sniffer' because it exhibits perfect adaptation. As Tyson et al. explain, 'perfect adaptation means that although the signaling pathway exhibits a transient response to changes in signal strength, its steady-state response R_{ss} is independent of S'. This corresponds to a similar adaptation of our sense of smell, which senses a new smell or a change in its intensity, but stops sensing it after a while, once we get used to it, regardless of the intensity level. Since the intensity of the signal is modelled by S, this is a system where the changes in S are explicit. However, they are effected as step-changes between periods when S is constant. Thus, the system is analysed during the periods in which S is constant, as before.

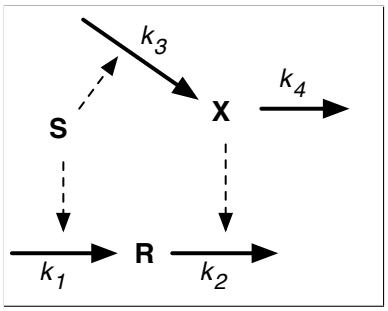


Fig. 6: **Pathway for Case 1d in [17]**

4.1 Analytical Solution with *Mathematica*

Since the second equation is separable, it can be integrated directly to yield

$$X(t) = \frac{k_3}{k_4} S - \left(\frac{k_3}{k_4} S - X_0 \right) e^{-k_4 t}. \tag{64}$$

Substituting this solution into the first equation gives a single ODE for R:

$$\frac{dR}{dt} = k_1 S - k_2 R \left[\frac{k_3}{k_4} S - \left(\frac{k_3}{k_4} S - X_0 \right) e^{-k_4 t} \right]. \tag{65}$$

Tyson el al. set the constants to the following values

$$k_1 = k_2 = 2 \qquad \text{and} \qquad k_3 = k_4 = 1, \tag{66}$$

such that

$$\frac{dR}{dt} = 2S - 2R \left[S - (S - X_0)\, e^{-t} \right]. \tag{67}$$

Setting $X_0 = 1$, this ODE has a general solution given by *Mathematica* as

$$R(t) = e^{-2[St + (S-1)e^{-t}]} \left[C + \int_1^t 2 e^{2[Sw + (S-1)e^{-w}]} S dw \right]. \tag{68}$$

It is informative to focus on the three values of $S = 0.5$, 1.0, and 2.0, for which the ODEs are:

$$\text{Case 1d.1:} \qquad \frac{dR}{dt} = 1 - R \left(1 + e^{-t} \right) \tag{69}$$

Case 1d.2: $$\frac{dR}{dt} = 2(1-R) \tag{70}$$

Case 1d.3: $$\frac{dR}{dt} = 4 - 2R\left(2 - e^{-t}\right). \tag{71}$$

It turns out that these are all integrable analytically, their solutions being special cases of the function (68). The solution for Case 1d.2 is elementary by separation of variables, whereas the other two cases were obtained with *Mathematica*, as follows:

$$R(t) = e^{(e^{-t}-t)}\left[C + e^{(e^{-t}-t)} + \mathrm{Ei}(-e^{-t})\right] = 1 + e^{(e^{-t}-t)}\left[C + \mathrm{Ei}(\sinh(t) - \cosh(t))\right] \tag{72}$$

$$R(t) = 1 - Ce^{-2t} \tag{73}$$

$$R(t) = e^{(-2e^{-t}-4t)}\left\{C + \frac{1}{3}\left[e^{(2e^{-t}+t)}(4 + 2e^{t} + 2e^{2t} + 3e^{3t}) - 8\mathrm{Ei}(2e^{-t})\right]\right\}, \tag{74}$$

where Ei( ) is the Exponential Integral function. Enforcing the initial condition $R(0) = R_0$, we can solve for the constant of integration in the three cases:

1d.1: $$C = \frac{R_0 - 1}{e} - \mathrm{Ei}(-1) \tag{75}$$

1d.2: $$C = 1 - R_0 \tag{76}$$

1d.3: $$C = e^2(R_0 - \frac{11}{3}) + 8\mathrm{Ei}(2), \tag{77}$$

and substitute back to obtain the particular solutions

1d.1: $$R(t) = e^{-e^{-x}(e^x x - 2X_0 + 1)}\Big((2X_0 - 1)\mathrm{Ei}\left(e^{-x}(1 - 2X_0)\right) + (1 - 2X_0)\mathrm{Ei}(1 - 2X_0) + R_0 e^{1-2X_0} + e^{-2e^{-x}X_0 + x + e} \tag{78}$$

1d.1: $$R(t) = 1 + e^{(e^{-t}-t)}\left[\frac{R_0 - 1}{e} - \mathrm{Ei}(-1) + \mathrm{Ei}(\sinh(t) - \cosh(t))\right] \tag{79}$$

1d.2: $$R(t) = 1 - (1 - R_0)e^{-2t} \tag{80}$$

1d.3: $$R(t) = e^{(-2e^{-t}-4t)}\Big\{e^2(R_0 - \frac{11}{3}) + 8\mathrm{Ei}(2) + \frac{1}{3}\left[e^{(2e^{-t}+t)}(4 + 2e^{t} + 2e^{2t} + 3e^{3t}) - 8\mathrm{Ei}(2e^{-t})\right]\Big\}. \tag{81}$$

These solution curves are plotted in Figure 7 for $R_0 = 1.5$.

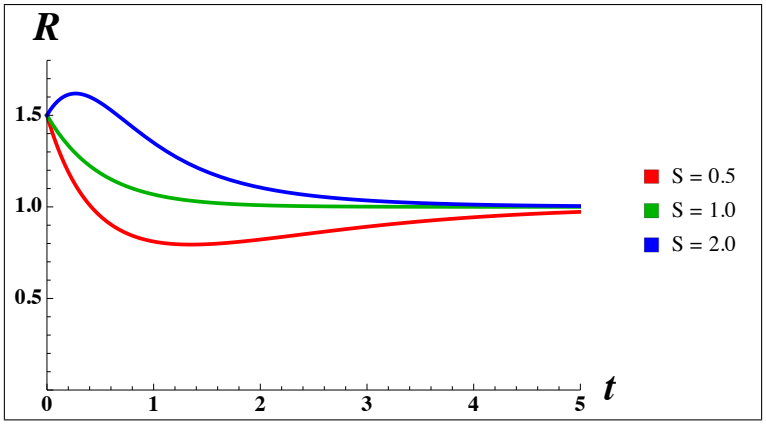


Fig. 7: **Exact solution of Case 1d from [17], for $R_0 = 1.5$ and different values of S**

4.2 Analytical Solution with Lie Groups

The hulls and solution vector fields for Eqs. (69), (70), and (71) are shown in Figure 8. The vector fields are clearly tangent to the solution curves shown in Figure 7 for the corresponding values of S.

This visualization may help us find us a set of symmetries that solve this problem, or may help us interpret them and why they worked once we have found them. However, for now we attempt a Lie group analysis by going back to the original system (63), using the constant values given by (66):

$$\frac{dR}{dt} = 2S - 2XR, \qquad \frac{dX}{dt} = S - X. \tag{82}$$

To facilitate formal manipulation in applying the results presented in [8], we rewrite this system in (x, y) notation:

$$\frac{dy^1}{dx} = \omega^1(x, y^1, y^2), \qquad \omega^1(x, y^1, y^2) = 2(S - y^1 y^2) \tag{83}$$

$$\frac{dy^2}{dx} = \omega^2(x, y^1, y^2), \qquad \omega^2(x, y^1, y^2) = S - y^2. \tag{84}$$

Consistently with standard differential geometry usage the superscripts are not powers but denote the dimension intended.

Finding a Symmetry. We seek symmetries as families of curves parametrized by a real parameter ϵ and expressible in Taylor series form as

$$\hat{x}(x, y^1, y^2; \epsilon) = x + \epsilon\xi(x, y^1, y^2) \tag{85}$$

$$\hat{y}^1(x, y^1, y^2; \epsilon) = y^1 + \epsilon\eta^1(x, y^1, y^2) \tag{86}$$

$$\hat{y}^2(x, y^1, y^2; \epsilon) = y^2 + \epsilon\eta^2(x, y^1, y^2) \tag{87}$$

As derived in [8], the LSC for this system involves the following two partial differential equations (PDEs), which must be satisfied simultaneously by the three functions ξ, η^1, and η^2:

$$\eta^1_x + \omega^1\eta^1_{y^1} + \omega^2\eta^1_{y^2} - \omega^1(\xi_x + \omega^1\xi_{y^1} + \omega^2\xi_{y^2}) - \xi\omega^1_x - \eta^1\omega^1_{y^1} - \eta^2\omega^1_{y^2} = 0 \tag{88}$$

$$\eta^2_x + \omega^1\eta^2_{y^1} + \omega^2\eta^2_{y^2} - \omega^2(\xi_x + \omega^1\xi_{y^1} + \omega^2\xi_{y^2}) - \xi\omega^2_x - \eta^1\omega^2_{y^1} - \eta^2\omega^2_{y^2} = 0, \tag{89}$$

where the subscripts indicate partial derivatives. In order to minimize the work needed to find a symmetry, we make use of the Lie algebra decomposition presented in [8] and focus only on time-preserving symmetries. Therefore, $\xi(x, y^1, y^2) \equiv 0$. The above equations therefore become:

$$\eta^1_x + \omega^1\eta^1_{y^1} + \omega^2\eta^1_{y^2} - \eta^1\omega^1_{y^1} - \eta^2\omega^1_{y^2} = 0 \tag{90}$$

$$\eta^2_x + \omega^1\eta^2_{y^1} + \omega^2\eta^2_{y^2} - \eta^1\omega^2_{y^1} - \eta^2\omega^2_{y^2} = 0. \tag{91}$$

Substituting for the functions ω^1 and ω^2 defining the hull and given by Eqs. (83) and (84),

$$\eta^1_x + 2(S - y^1y^2)\eta^1_{y^1} + (S - y^2)\eta^1_{y^2} + 2y^2\eta^1 + 2y^1\eta^2 = 0 \tag{92}$$

$$\eta^2_x + 2(S - y^1y^2)\eta^2_{y^1} + (S - y^2)\eta^2_{y^2} + \eta^2 = 0. \tag{93}$$

Since the second equation depends only on η^2 let's work with it first. Assume

$$\eta^2(x, y^1, y^2) = c_0 + c_1x + c_2y^1 + c_3y^2. \tag{94}$$

Substituting in (93),

$$c_0 + c_1 + c_1x + c_2y^1 + c_3(S - y^2) + c_3y^2 + 2c_2(S - y^1y^2) = 0. \tag{95}$$

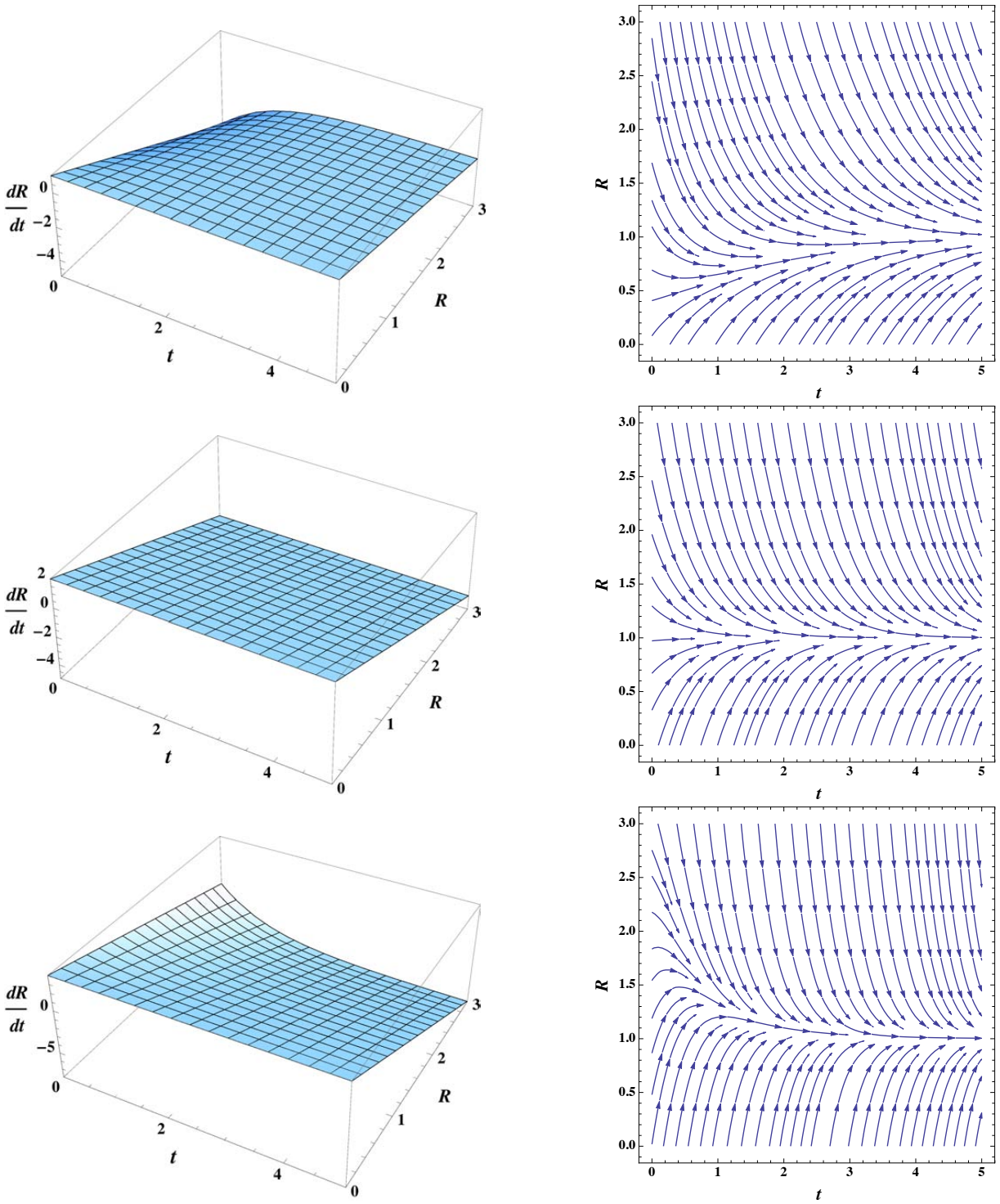


Fig. 8: **Hulls of Eqs. (69), (70), and (71) (left column) in the jet space for Case 1d and vector fields tangent to the solution curves of the same three ODEs (right column), organized by row for $S = 0.5$, 1.0 and 2.0, respectively. The solution curves are level sets of the corresponding first integral surfaces, *not* of the hulls shown on left.**

Rather than treating this equation as part of a system of 3 algebraic equations by means of which *a single set* of values of x, y^1, and y^2 that simultanesouly satisfy them could be found, this condition must be true for *all* possible values of these variables. Therefore, the coefficients of each power of the depedendent and independent variables must equal 0 separately. Since c_1 is the only coefficient that multiplies x and c_2 the only coefficient that multiplies y^1y^2 (or, equivalently, y^1), they must both equal 0. Therefore, (95) becomes

$$c_0 + c_3(S - y^2) + c_3y^2 = 0. \tag{96}$$

Simplifying further, we see that the terms involving y^2 cancel, leaving

$$c_0 + c_3 S = 0. \tag{97}$$

This is one equation involving two undetermined constants, so one of them is arbitrary. We set $c_3 = 1$, such that $c_0 = -S$ and

$$\boxed{\eta^2(x, y^1, y^2) = y^2 - S.} \tag{98}$$

We can now substitute this function in the LSC for the first ODE and solve for η^1:

$$\eta^1_x + 2(S - y^1 y^2)\eta^1_{y^1} + (S - y^2)\eta^1_{y^2} + 2y^2\eta^1 - 2(S - y^2)y^1 = 0. \tag{99}$$

As we well know there is no systematic way to do this, so we report here the steps that worked. The first heuristic was to attempt to simplify Eq. (92) as much as possible. As shown in Eq. (100), it seems possible to guess at a function $\eta^1(x, y^1, y^2)$ such that the terms coloured in red font cancel, and similarly for the terms in blue font:

$$\eta^1_x + 2S\eta^1_{y^1} - 2y^1 y^2 \eta^1_{y^1} + (S - y^2)\eta^1_{y^2} + 2y^2\eta^1 - 2(S - y^2)y^1 = 0. \tag{100}$$

After some playing around with pencil and paper, the following function seemed like a promising start:

$$\eta^1(x, y^1, y^2) = b_0 y^1 y^2 + f(x, y^1, y^2). \tag{101}$$

Plugging into (100),

$$\begin{aligned} f_x + 2S(b_0 y^2 + f_{y^1}) - 2y^1 y^2(b_0 y^2 + f_{y^1}) + (S - y^2)(b_0 y^1 + f_{y^2}) \\ + 2y^2(b_0 y^1 y^2 + f) - 2(S - y^2)y^1 = 0 \\ f_x + 2b_0 S y^2 + 2S f_{y^1} - 2b_0 y^1 (y^2)^2 - 2y^1 y^2 f_{y^1} + b_0(S - y^2)y^1 + (S - y^2) f_{y^2} \\ + 2b_0 y^1 (y^2)^2 + 2y^2 f - 2(S - y^2)y^1 = 0. \end{aligned} \tag{102}$$

If we now let $b_0 = 2$ the red and blue terms cancel, leaving us with

$$f_x + 4Sy^2 + 2Sf_{y^1} - 2y^1 y^2 f_{y^1} + (S - y^2)f_{y^2} + 2y^2 f = 0. \tag{103}$$

This looks almost as bad, but let's press on, assuming the following functional form for f:

$$f(x, y^1, y^2) = c_0 + c_1 e^{c_2 x} + c_3 y^1 y^2. \tag{104}$$

Substituting into (103),

$$c_1 c_2 e^{c_2 x} + c_3 y^1 (S - y^2) + 4Sy^2 + 2c_3 S y^2 - 2c_3 y^1 (y^2)^2 + 2y^2(c_0 + c_1 e^{c_2 x} + c_3 y^1 y^2) = 0. \tag{105}$$

Collecting terms with respect to y^2, we get

$$c_1 c_2 e^{c_2 x} + c_3 y^1 S + \left(c_3 y^1 + 4S + 2c_3 S - \cancel{2c_3 y^1 y^2} + 2c_0 + 2c_1 e^{c_2 x} + \cancel{2c_3 y^1 y^2}\right) y^2 = 0. \tag{106}$$

Clearly, $c_3 = 0$ and $c_1 = 0$, so that

$$(4S + 2c_0)\, y^2 = 0 \qquad \longrightarrow \qquad c_0 = -2S. \tag{107}$$

Hence,

$$\boxed{\eta^1(x, y^1, y^2) = 2(y^1 y^2 - S)} \tag{108}$$

The infinitesimal generator of this symmetry is the operator obtained from the dot product of the infinitesimal symmetry vector tangent to the hull and the gradient operator:

$$X = \boldsymbol{X} \cdot \boldsymbol{\nabla} = (\xi \hat{i} + \eta^1 \hat{j} + \eta^2 \hat{k}) \cdot \left(\frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y^1}\hat{j} + \frac{\partial}{\partial y^2}\hat{k} \right) \tag{109}$$

$$= \left[0\hat{i} + 2(y^1y^2 - S)\hat{j} + (y^2 - S)\hat{k}\right] \cdot \left(\frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y^1}\hat{j} + \frac{\partial}{\partial y^2}\hat{k}\right) \tag{110}$$

$$= 2(y^1y^2 - S)\frac{\partial}{\partial y^1} + (y^2 - S)\frac{\partial}{\partial y^2}, \tag{111}$$

where $\hat{i}$, $\hat{j}$, $\hat{k}$ are the unit vectors corresponding to the three jet-space variables x, y^1, and y^2. A good way to understand this operator is through its effect on any scalar function ϕ defined over the (x, y^1, y^2) space: it gives the directional derivative (rate of change) of $\phi(x, y^1, y^2)$ in the direction of $\boldsymbol{X}$. This is most easily visualized when ϕ is a function of only 2 variables, i.e. a regular surface, but the concept generalizes to any number of dimensions. Finally, although we wrote out all the details in order to show the connection between the differential geometry notation used in mathematics and the 3-D vector notation more common in physics, in general we tend to be more sloppy in referring interchangeably to the infinitesimal generator of the symmetry as a vector and/or an operator.

We notice that both components of the infinitesimal symmetry vector turned out to equal factors explicitly present in the linearized symmetry condition, Eqs. (92) and (93). It is not clear whether this is a coincidence or whether it has a deeper significance, but we will keep our eyes open for possible new insights related to this observation in the Lie group method in general and/or in this problem in particular. We now establish a few geometrical facts.

Visualization of the Results So Far. As discussed in [8], the symmetry generator is tangent to the hull, which is embedded in the jet space. In the case of a system of two ODEs, the hull is defined by two simultaneous constraint equations which are nothing more than the original governing equations (83) and (84) written in terms of functions defined in the jet space:

$$\begin{aligned} \Sigma^1(x, y_1, y_2, \dot{y}_1, \dot{y}_2) &= \dot{y}_1 - \omega^1(x, y^1, y^2) = 0 \\ \Sigma^2(x, y_1, y_2, \dot{y}_1, \dot{y}_2) &= \dot{y}_2 - \omega^2(x, y^1, y^2) = 0, \end{aligned} \tag{112}$$

where $\dot{y}_j = \frac{dy^j}{dx}$. We can say the following:

- Each function $\Sigma^j(x, y_1, y_2, \dot{y}_1, \dot{y}_2)$ is a 5-D manifold that can in principle be plotted in a 6-D jet space. So the 6th dimension accounts for the value of this function at each of the points defined by the 5-D space of its independent variables.
- If we set each $\Sigma^j = 0$, for each equation we get a constraint on the 5-D space, so the set of points that satisfy each equation can be "visualized" as a 4-D manifold embedded in a 6-D space.
- Since each $\Sigma^j = 0$ is a constraint on the same 5-D space, each defines a 4-D manifold. Since the two equations are supposed to apply simultaneously (or equivalently, since they are both equal to zero they can be set equal to each other), we are interested in the set of points that satisfy both equations simultaneously. This is a 3-D manifold or a solid.
- To see the above, we can think of the intersection of 2 normal 2-D surfaces in Eucleadian 3-D space: their intersection *can* be a finite or infinite 2-D manifold, if the two surfaces overlap exactly on a finite or infinite subset of their points, but the simpler picture is that they intersect at a non-zero angle, such that their intersection is "usually" a 1-D space curve. For the same reason, the subset Σ (i.e. the hull) *can* be a 4-D object, but "normally" it will be a 3-D object.

We can therefore see why the infinitesimal generator for two 1st-order ODEs in general is a 3-D vector (see e.g. Eq. (109)). In the case of this specific problem, however, we have sought and found a time-preserving symmetry, for which $\xi(x, y^1, y^2) \equiv 0$. Therefore, we are dealing with a vector field in two dimensions, which is essentially a cross-section of Σ and can be easily visualized as shown in Figure 9. We should emphasize that, unlike the vector plots shown in Figure 8, these vector plots show the vector field tangent to the symmetry curves rather than to the solution curves.

5 Conclusion

In order to solve Case 1d we need to seek canonical coordinates for this higher-dimensional system, which is not a trivial extension of the method used for a single 1st-order ODE. In this paper we have

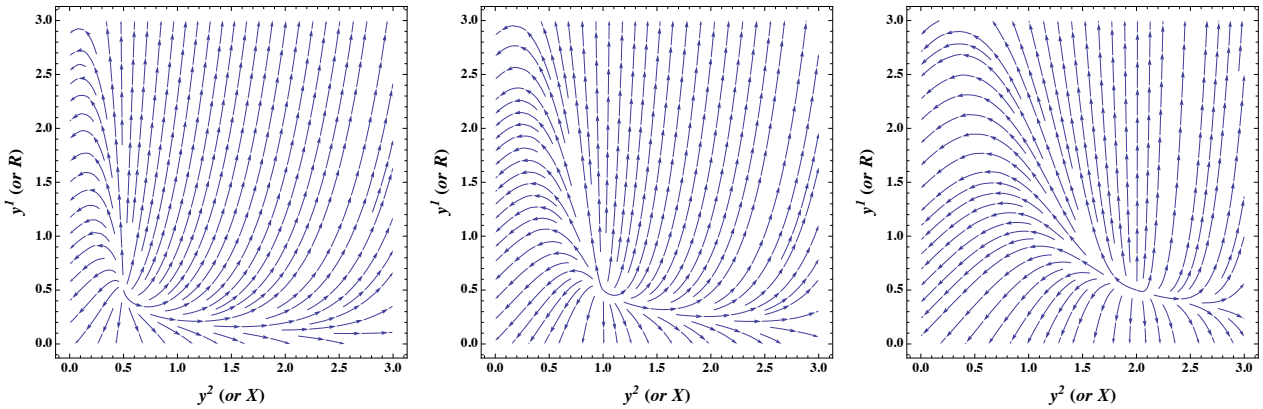


Fig. 9: **Vector fields corresponding to the infinitesimal generator of the Lie group, Eq. (111), for $S = 0.5$, 1.0, and 2.0. The (y^1, y^2) plane shown is embedded in the jet space and is a cross-section of the 3-D hull of Case 1d orthogonal to the time axis.**

been contented to demonstrate the Lie group method for two different metabolic pathway modules from Tyson et al.'s seminal paper [17], Cases 1c and 1d. Having found one symmetry group for each system, the main conclusion we can draw is that although the geometrical visualizations are very pleasing they are only partly successful in increasing our understanding of the method. In truth, however, their effectiveness has not been demonstrated to the fullest extent in this paper. For example, we have not shown visualizations of the canonical coordinates for a single 1st-order ODE. Further, although the situation in general is even worse for higher-dimensional systems, we expect the visualizations to be very helpful for two 1st-order ODEs such as Case 1d analysed here. Therefore, we will pick up the thread in the next paper or project report and will complete the Lie group analysis of Case 1d then.

References

1. U Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall, Boca Raton, Florida, 2007.
2. G W Bluman and J D Cole. *Similarity Methods for Differential Equations*. Springer, 1974.
3. G W Bluman and S Kumei. *Symmetries and Differential Equations*. Springer, 1989.
4. P Dini. Basic Facts about Symmetries of ODEs, 2005. Available from: http://www.biomicsproject.eu/file-repository/category/27-public-publications-techrep.
5. P Dini. Lie Group Analysis of Three Single-ODE Biochemical Pathway Modules, September 2013. University of Hertfordshire Computer Science Technical Report.
6. A Goldbeter and D E Koshland. An amplified sensitivity arising from covalent modification in biological systems. *Proceedings of the National Academy of Science*, 78(11):6840–6844, 1981.
7. B Hayes. Why W? *American Scientist*, 93(2):104–108, 2005.
8. G Horváth and Z Muzsnay. Linearised Symmetry Condition for Systems of ODEs. In P Dini and G Horváth, editors, *1st BIOMICS Summer Workshop*, University of Debrecen, Hungary, 17-19 July, 2013.
9. R Howe. Very Basic Lie Theory. *Americal Mathematical Monthly*, 90(9):600–623, 1983.
10. P E Hydon. *Symmetry Methods for Differential Equations: A Beginner's Guide*. Cambridge University Press, 2000.
11. N K Ibragimov. Group analysis of ordinary differential equations and the invariance principle in mathematical physics (for the 150th anniversary of Sophus Lie). *Russian Mathematical Surveys*, 47(4):89–156, 1992.
12. P Olver. *Applications of Lie Groups to Differential Equations*. Springer, 2nd edition, 1993.
13. C Rogers and W F Ames. *Nonlinear Boundary Value Problems in Science and Engineering*. Academic Press, Boston, 1989.
14. S Schnell and C Mendoza. Closed Form Solution for Time-dependent Enzyme Kinetics. *Journal of Theoretical Biology*, 187:207–212, 1997.
15. R Seshadri and T Y Na. *Group Invariance in Engineering Boundary Value Problems*. Springer, New York, 1985.
16. H Stephani. *Differential equations: Their solution using symmetries*. Cambridge University Press, Cambridge, 1989.
17. J J Tyson, K C Chen, and B Novak. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology*, 15:221–231, 2003.
18. A R Tzafriri. Michaelis-Menten Kinetics at High Enzyme Concentrations. *Bulletin of Mathematical Biology*, 65:1111–1129, 2003.
19. C E Wulfman. *Dynamical Symmetry*. World Scientific, Singapore, 2011.

Linearised Symmetry Condition for Systems of ODEs

Gábor Horváth and Zoltán Muzsnay

Institute of Mathematics
University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
[ghorvath,muzsnay]@science.unideb.hu

Abstract. In this note we derive the linearised symmetry condition for a system of ODEs (see formula 24). We also give a decomposition of the Lie algebra of infinitesimal symmetries into the direct sum of two components (Theorem 12). The first component is the trivial one, determined by the direction of the flow of the vector field associated to the system of ODEs. The elements of the second component will be called time-preserving infinitesimal symmetries, because their action on the time coordinate is trivial. These elements can be characterised by the Lie bracket condition (see Lemma 11).

In this paper the Einstein summation convention for repeated indexes will be applied.

1 Systems of ODEs, Associated Vector Fields and Solutions

Let us consider the following time-dependent system of ordinary differential equations in $\mathbb{R}^n$:

$$\frac{dy^i}{dx} = \omega^i(x, y), \qquad i = 1...n. \tag{1}$$

We can complete $y = (y^1, ..., y^n) \in \mathbb{R}^n$ with the $y^0 := x$ coordinate to obtain $(y^0, y) := (y^0, y^1, ..., y^n) \in \mathbb{R}^{n+1}$ and have a time-independent system in $\mathbb{R}^{n+1}$:

$$\frac{dy^0}{dx} = 1, \quad \frac{dy^1}{dx} = \omega^1(y^0, y), \quad ... \quad \frac{dy^n}{dx} = \omega^n(y^0, y). \tag{2}$$

Introducing the vector field $Y = Y^\alpha \frac{\partial}{\partial} y^\alpha$ in $\mathbb{R}^{n+1}$ where

$$Y^0 = 1, \quad Y^1 = \omega^1, \quad ... \quad , \; Y^n = \omega^n, \tag{3}$$

the system (1) can be represented by the vector field Y as

$$\frac{dy}{dt} = Y(y) \tag{4}$$

with $y = (y^0, ..., y^n)$, which is a system of autonomous differential equations in $\mathbb{R}^{n+1}$.

A solution of (1) is a curve $x \to y(x) \in \mathbb{R}^n$ such that

$$\frac{dy^i}{dx}(x) = \omega^i(x, y(x)) \tag{5}$$

and a solution of (4) is a curve $t \to y(t)$ in $\mathbb{R}^{n+1}$ such that

$$\frac{dy}{dt}(t) = Y(y(t)) \qquad \Leftrightarrow \qquad \begin{cases} \dfrac{dy^0}{dt}(t) = 1, \\ \dfrac{dy^i}{dt}(t) = \omega^i(y^0(t), y^i(t)). \end{cases} \tag{6}$$

Here I am using a parameter t to emphasize that the parameter in the $(n+1)$-dimensional space can be different from x, but of course, because the first coordinate of Y is 1, we will have $\frac{dy^0}{dt} = 1$ and we can use $y^0 = x$ to parametrize the solutions of Y.

The systems (1) and (6) are equivalent: from the solution of the first we can have a solution of the second by considering its graph

$$\{x \to y(x)\} \qquad \Leftrightarrow \qquad \{x \to (x, y(x))\}$$

and vice versa, with the solution of (6) one can find a solution of (1).

2 1-Jet Space of Curves

Let us consider parametrised curves in $\mathbb{R}^n$. We say that two curves γ_1 and γ_2 ($\gamma_1, \gamma_2 : \mathbb{R} \to \mathbb{R}^n$) are equivalent to order k at x_0 if their Taylor series coincide up to order k. This means that γ_1 and γ_2 are equivalent to order k at x_0 iff

$$\gamma_1(x_0) = \gamma_2(x_0), \qquad \frac{d\gamma_1}{dx}\Big|_{x_0} = \frac{d\gamma_2}{dx}\Big|_{x_0}, \qquad \ldots \qquad \frac{d^k\gamma_1}{dx^k}\Big|_{x_0} = \frac{d^k\gamma_2}{dx^k}\Big|_{x_0}.$$

The equivalence to order k at x_0 is actually an equivalence relation. The equivalence class of the curve γ is denoted by $j_{k,x_0}(\gamma)$. The set of all equivalence classes (for all x_0 and γ) is denoted as $J_k(\mathbb{R}, \mathbb{R}^n)$. An element $j_{k,x_0}(\gamma)$ of $J_k(\mathbb{R}, \mathbb{R}^n)$ can be characterized by the following data: the value of x_0, the value of the first derivatives at x_0, ..., the value of the k^{th} derivatives at x_0. In this manner, $(x, y^i, \dot{y}^i, \ddot{y}^i, \ldots)$ is a coordinate system for $J_k(\mathbb{R}, \mathbb{R}^n)$, where x is the source coordinate, y^i are the target coordinates, $\dot{y}^i$ give the first derivatives at x_0 etc. In particular, $J_k(\mathbb{R}, \mathbb{R}^n)$ is a finite-dimensional space.

Because we are dealing with a system of first-order ODEs, we need the first-order jet space $J_1(\mathbb{R}, \mathbb{R}^n)$. Two curves γ_1 and γ_2 determine the same 1^{st}-order jet at x_0 if and only if $\gamma_1(x_0) = \gamma_2(x_0)$ and $\dot{\gamma}_1(x_0) = \dot{\gamma}_2(x_0)$.

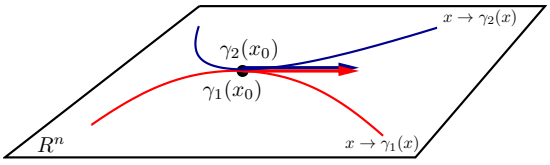


Fig. 1: γ_1 and γ_2 are equivalent to order 1 at x_0

The 1-jet space of curves in $\mathbb{R}^n$ is a $2n+1$-dimensional space

$$J_1(\mathbb{R}, \mathbb{R}^n) = \{(x, y^i, \dot{y}^i) \mid x, y^i, \dot{y}^i \in \mathbb{R}\},$$

where $(x, y^i, \dot{y}^i)$ are the coordinate functions.

2.1 Prolongation of a Curve

If γ is a curve ($\gamma : x \to y(x)$), then one can consider its *prolongation* which will be denoted by $j_1(\gamma)$. It is a curve in $J_1(\mathbb{R}, \mathbb{R}^n)$ and it is easy to find it: one has to complete the original curve by its derivative at every x:

$$j_1(\gamma) : x \;\longrightarrow\; (x, y^i(x), \dot{y}^i(x)) = (x, y^i(x), \tfrac{dy^i}{dx}(x)) \tag{7}$$

This prolongation can be visualized in the following way:

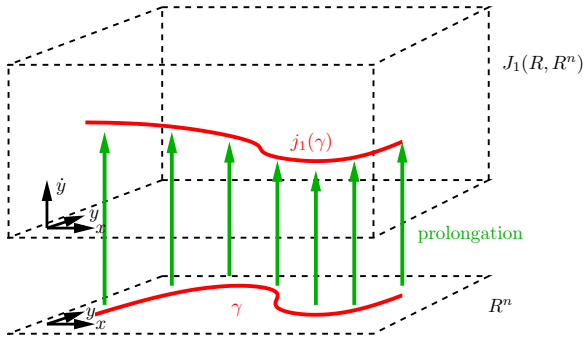


Fig. 2: Prolongation of a curve γ

3 First-Order System of ODEs as a Subset of the 1-Jet Space

Since the 1-jet space of curves contains information about the curves and their derivatives, the first-order system (1) can be interpreted as a subset Σ of this space. This subset is characterised by the equations $\Sigma^j(x, y^i, \dot{y}^i) = 0$, where $\Sigma^j = \dot{y}^j - \omega^j(x, y)$. These are *not differential but algebraic equations in* $J_1(\mathbb{R}, \mathbb{R}^n)$ between the independent coordinates $(x, y^i, \dot{y}^i)$. This is why it is so convenient to use them.

Just to have numbered equations we formally define the above introduced quantities:

Definition 1. The subset $\Sigma \subset J_1(\mathbb{R}, \mathbb{R}^n)$ associated with the system of first-order ODEs (1) is defined by

$$\Sigma := \{(x, y, \dot{y}) \mid \Sigma^j(x, y, \dot{y}) = 0\}$$

where $\Sigma^j : J_1(\mathbb{R}^{n+1}) \to \mathbb{R}$ are the functions

$$\Sigma^j(x, y^i, \dot{y}^i) := \dot{y}^j - \omega^j(x, y), \qquad j = 1, ..., n. \tag{8}$$

A curve $\gamma : x \to y(x)$ is a solution of (1) if and only if

$\Leftrightarrow$ $\frac{dy^j}{dx}(x) = \omega^j(x, y(x))$;
$\Leftrightarrow$ $\frac{dy^j}{dx}(x) - \omega^j(x, y(x)) = 0$;
$\Leftrightarrow$ $\Sigma^j(x, y^i(x), \frac{dy^i}{dx}(x)) = 0$, because that is how we introduced these functions;
$\Leftrightarrow$ $\Sigma^j(x, y^i(x), \dot{y}^i(x)) \equiv 0$, because we introduced the coordinates $(x, y^i, \dot{y}^i)$ for a curve γ in this way;
$\Leftrightarrow$ $j_{1,x}(\gamma) \in \Sigma$, because Σ was defined as the common zeros of the Σ^j functions.

Therefore we have the

Property 2. The curve γ is a solution of (1) if and only if its prolongation lies identically in the subset Σ, that is

$$\Sigma^j(x, y^i(x), \dot{y}^i(x)) \equiv 0 \tag{9}$$

for $j = 1, ..., n$.

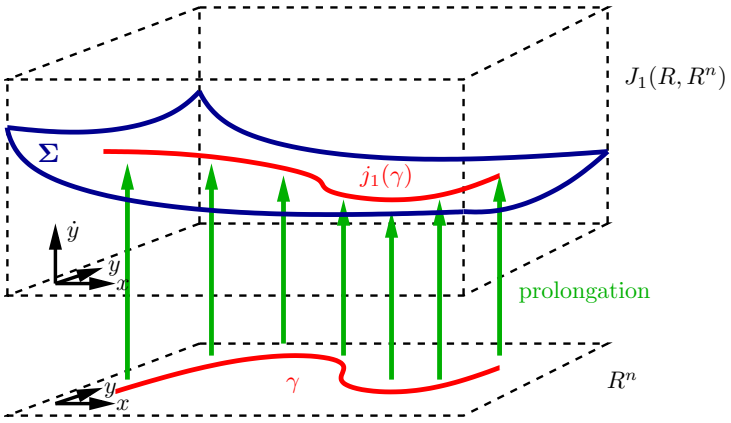


Fig. 3: Prolongation of a **solution**

4 The Prolongation of a Diffeomorphism in the Jet Space

If there is a diffeomorphism S (for example a symmetry) transforming $\mathbb{R}^{n+1}$ into itself, then every object (e.g. a curve) is moving and eventually changing under this transformation. If $x \to \gamma(x)$ is a curve, then the new curve will be $\bar{x} \to \bar{\gamma}(\bar{x})$. For both γ and $\bar{\gamma}$ the prolongated curves ($j_1\gamma$ and $j_1\bar{\gamma}$) can be computed. That shows that there is a map of the jet space into itself *induced* by S. This induced map is called the *first prolongation* of S and it will be denoted by S^1. In this section we will determine this map.

The diagram shows what we are looking for:

$$\begin{array}{ccc} J_1(\mathbb{R},\mathbb{R}^n) & \xrightarrow{S^1=???} & J_1(\mathbb{R},\mathbb{R}^n) \\ \text{prolongation}\uparrow & & \uparrow\text{prolongation} \\ \mathbb{R}\times\mathbb{R}^n & \xrightarrow{S} & \mathbb{R}\times\mathbb{R}^n \end{array}$$

First we have to compute how a curve and its derivative change under the transformation.

Let $S{:}\mathbb{R}^{n+1}\to\mathbb{R}^{n+1}$ be a diffeomorphism, $S(x, y^i) = (\bar{x}, \bar{y}^i)$, where

$$\begin{aligned} \bar{x} &= \varphi(x, y) \\ \bar{y}^i &= \psi^i(x, y) \end{aligned}$$

Let $\gamma : x \to y(x)$ be a curve in $\mathbb{R}^n$ and $\bar{\gamma} : \bar{x} \to \bar{y}(\bar{x})$ its image. Then

$$\frac{d\bar{y}^i}{d\bar{x}} = \frac{d\bar{y}^i}{dx}\frac{dx}{d\bar{x}} = \frac{d\bar{y}^i}{dx}\frac{1}{d\bar{x}/dx} = \frac{\psi^i_x + \psi^i_{y^j}\dot{y}^j}{\varphi_x + \varphi_{y^j}\dot{y}^j} = \frac{D\psi^i}{D\varphi} \tag{10}$$

where D denotes the total derivative with respect to x.

The formula (10) shows how the derivatives of the curve γ are changing when the diffeomorphism S transforms γ. Therefore the prolongation S^1 of a diffeomorphism $S{:}\mathbb{R}^{n+1}\to\mathbb{R}^{n+1}$ is given by

$$\begin{aligned} J_1(\mathbb{R},\mathbb{R}^n) &\xrightarrow{S^1} J_1(\mathbb{R},\mathbb{R}^n) \\ (x, y^i, \dot{y}^i) &\xrightarrow{S^1} (\bar{x}, \bar{y}^i, \dot{\bar{y}}^i) = (\varphi, \psi^i, \tfrac{D\psi^i}{D\varphi}) \end{aligned} \tag{11}$$

5 A 1-Parameter Family of Diffeomorphisms and Infinitesimal Generators

If $\{S_\varepsilon\}_{\varepsilon\in\mathbb{R}}$ is a 1-parameter group of diffeomorphisms (1-parameter family of symmetries) of $\mathbb{R}^{n+1}$, then two important objects can be associated to it: the *infinitesimal generator* and *the prolongation*

of the infinitesimal generator of S_ε.

The idea is the following:

Infinitesimal generator: for every point p the map $\varepsilon \to S_\varepsilon(p)$ is a curve. Differentiating with respect to ε we get a vector X_p. Since we can do it at every point, we can get a vector field X. The vector field X is called the *infinitesimal generator of* S_ε.

Prolongation of the infinitesimal generator: The idea is the same as in the case of the infinitesimal generator, but here we use S^1_ε instead of S_ε.
Indeed, for every ε the map S_ε is a diffeomorphism. From the previous chapter we know that we can consider its prolongation, for which S_ε will be a map in the 1-jet space. Doing this for every ε we get a 1-parameter family of diffeomorphisms $\{S^1_\varepsilon\}_{\varepsilon\in\mathbb{R}}$ of the 1-jet space. Differentiating it with respect to ε at a point (of the 1-jet space) we get a vector. Doing this at every point, we can obtain a vector field, which will be denoted by X^1. The vector field X^1 is called the *first prolongation of the infinitesimal generator* X.

Let us find how we can compute the coordinate expression of the infinitesimal generator and the prolongation of the infinitesimal generator.

Let $\{S_\varepsilon\}_{\varepsilon\in\mathbb{R}} = \{(\varphi_\varepsilon, \psi^i_\varepsilon)\}_{\varepsilon\in\mathbb{R}}$ be 1-parameter family of diffeomorphism of $\mathbb{R}^{n+1}$ such that for $\varepsilon = 0$ we have $S_0 = id$, the identity map. Then for $\varepsilon = 0$ we have

$$\begin{aligned} &\varphi_0(x,y) = x, \quad \frac{\partial}{\partial\varphi_0}x = 1, \quad \frac{\partial}{\partial\varphi_0}y^j = 0,\\ &\psi^i_0(x,y) = y^i, \quad \frac{\partial}{\partial\psi^i_0}x = 0, \quad \frac{\partial}{\partial\psi^i_0}y^j = \delta^i_j, \end{aligned} \tag{12}$$

where δ^i_j is the Kronecker delta.

5.1 The Infinitesimal Generators

The infinitesimal generator of $\{S_\varepsilon\}$ will be denoted by X. X is a vector field on $\mathbb{R}^{n+1}$ such that $\frac{dS_\varepsilon}{d\varepsilon} = X$ at every point $(x,y) \in \mathbb{R}\times\mathbb{R}^n$. The components of X will be denoted as it is usual by (ξ, η^i), that is,

$$\boxed{X = \xi\frac{\partial}{\partial}x + \eta^i\frac{\partial}{\partial}y^i,} \tag{13}$$

where

$$\frac{d\varphi_\varepsilon}{d\varepsilon}\Big|_{\varepsilon=0} = \xi, \qquad \frac{d\psi^i_\varepsilon}{d\varepsilon}\Big|_{\varepsilon=0} = \eta^i, \quad i = 1,...,n. \tag{14}$$

5.2 The Prolongation of the Infinitesimal Generators

According to (11), for every ε the map S_ε induces a map S^1_ε of the jet space into itself. That way we have a 1-parameter family of maps:

$$\begin{aligned} J_1(\mathbb{R},\mathbb{R}^n) &\xrightarrow{S^1_\varepsilon} J_1(\mathbb{R},\mathbb{R}^n)\\ (x, y^i, \dot y^i) &\longrightarrow (\bar x_\varepsilon, \bar y^i_\varepsilon, \dot{\bar y}^i_\varepsilon) = (\varphi_\varepsilon, \psi^i_\varepsilon, \tfrac{D\psi^i_\varepsilon}{D\varphi_\varepsilon}) \end{aligned} \tag{15}$$

The infinitesimal generator X^1 of the prolongation is just the derivative of S^1_ε with respect to the parameter ε. Let us compute it:

$$X^1(x,y^i,\dot y^i) = \frac{d}{d\varepsilon}\Big|_{\varepsilon=0} S_\varepsilon(x,y^i,\dot y^i) = \frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\left(\bar x_\varepsilon, \bar y^i_\varepsilon, \dot{\bar y}^i_\varepsilon\right) = \frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\left(\varphi_\varepsilon,\ \psi^i_\varepsilon,\ \tfrac{D\psi^i_\varepsilon}{D\varphi_\varepsilon}\right) \tag{16}$$

Differentiating the coordinates term-by-term and using (14) we have

$$\begin{aligned}
\frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\varphi_\varepsilon &= \xi,\\
\frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\psi^i_\varepsilon &= \eta^i,\\
\frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\Big(\tfrac{D\psi^i_\varepsilon}{D\varphi_\varepsilon}\Big) &= \frac{d}{d\varepsilon}\Big|_{\varepsilon=0}\frac{\partial_x\psi^i_\varepsilon + \dot y^j(\partial_{y^j}\psi^i_\varepsilon)}{\partial_x\varphi_\varepsilon + \dot y^k(\partial_{y^k}\varphi_\varepsilon)}\\
&= \frac{(\eta^i_x + \dot y^j\eta^i_{y^j})(\partial_x\varphi_0 + \dot y^k(\partial_{y^k}\varphi_0)) - (\partial_x\psi^i_0 + \dot y^j(\partial_{y^j}\psi^i_0))(\xi_x + \dot y^k\xi_{y^k})}{\big(\partial_x\varphi_0 + \dot y^k(\partial_{y^k}\varphi_0)\big)^2}\\
&= \frac{(\eta^i_x + \dot y^j\eta^i_{y^j})(1+0) - (0 + \dot y^j\delta^i_j)(\xi_x + \dot y^k\xi_{y^k})}{(1+0)^2} = (\eta^i_x + \dot y^j\eta^i_{y^j}) - \dot y^i(\xi_x + \dot y^k\xi_{y^k})\\
&= D\eta^i - \dot y^i D\xi
\end{aligned}$$

where we also used (12). Therefore we get the coordinate expression of X^1:

$$\boxed{X^1 = \xi\frac{\partial}{\partial}x + \eta^i\frac{\partial}{\partial}y^i + (D\eta^i - \dot y^i D\xi)\frac{\partial}{\partial}\dot y^i} \tag{17}$$

6 Symmetry

The map $S:\mathbb{R}^{n+1}\to\mathbb{R}^{n+1}$ is a symmetry for (6) /and therefore a symmetry for (1) because they are equivalent/ if S transforms solutions of (6) into solutions of (6); that is, if $\gamma : x \to y(x)$ is a solution to (6), then the map $\bar\gamma : \bar x \to \bar y(\bar x)$ is also a solution to (6).

We can characterise solutions of (1) in two different ways: as integral curves of the vector field Y, and as a map such that its prolongation lies in Σ (see Proposition 2). Therefore we can characterize symmetries also in two different ways: by their action on the vector field Y, and by their action on Σ.

Property 3. (Using the vector field Y): S is a symmetry of (6) if and only if

$$\frac{d\gamma}{dx}(x) = Y(\gamma(x)) \quad\Rightarrow\quad \frac{d\bar\gamma}{d\bar x}(\bar x) = Y(\bar\gamma(\bar x)); \tag{18}$$

that is, S leaves the vector field invariant.

Property 4. Using the subset Σ in the 1-jet space: S is a symmetry of (6), if and only if

$$\Sigma^i(x, y(x), \dot y(x)) \equiv 0 \quad\Rightarrow\quad \Sigma^i(\bar x, \bar y(x), \dot{\bar y}(x)) \equiv 0; \tag{19}$$

that is, S^1 (the prolongation of S) leaves the set Σ invariant.

The coordinate expression of the prolongation of a symmetry was already computed in (11). Therefore we have the following

Property 5. The map S is a symmetry of (6) if and only if

$$\Sigma^i(x, y^j(x), \dot y^j(x)) \equiv 0 \quad\Rightarrow\quad \Sigma^i(\varphi, \psi^j(x), D\psi^j/D\varphi(x)) \equiv 0 \tag{20}$$

7 Linearised Symmetry Condition

In this chapter we will derive the linearised symmetry condition for a system of first-order ODEs. The idea is exactly the same as it is in the case of a single first-order ODE: we consider $\{S_\varepsilon\}_{\varepsilon\in\mathbb{R}}$ a 1-parameter group of symmetries. From Property 4 we know that $\{S_\varepsilon\}_{\varepsilon\in\mathbb{R}}$ leaves invariant the subset Σ. Therefore its infinitesimal generator X^1 (which is the prolongation of X) is tangent to Σ. Consequently the derivative of the functions of Σ with respect to X^1 are identically zero. Since we already calculated the coordinate expression of X^1 and we know the functions Σ^j we can calculate these derivatives. The expression that we get this way involves the coordinate $\dot{y}^i$. But we are doing our calculation on Σ and here (again using the expression $\Sigma^i = 0$) these coordinates can be expressed with ω^i. What we get at the end of this is just the linearised symmetry condition.

The calculation: let $\{S_\varepsilon\}_{\varepsilon\in\mathbb{R}} = \{\bar{x}_\varepsilon, \bar{y}_\varepsilon)\}_{\varepsilon\in\mathbb{R}} = \{(\varphi_\varepsilon, \psi^i_\varepsilon)\}_{\varepsilon\in\mathbb{R}}$ be a 1-parameter family of symmetries of (6) such that for $\varepsilon = 0$ we have $S_0 = id$. According to (20), for every ε we have

$$\Sigma^j(\bar{x}_\varepsilon, \bar{y}^i_\varepsilon, \dot{\bar{y}}^i_\varepsilon) \equiv 0 \tag{21}$$

whenever the equation $\Sigma^j(x, y^i(x), \dot{y}^i(x)) \equiv 0$ is satisfied, that is if

$$\dot{y}^j = \omega^j(x, y) \tag{22}$$

holds. By differentiating (21) with respect to ε we obtain

$$\frac{\partial}{\partial\Sigma^j}x\frac{\partial}{\partial\bar{x}_\varepsilon}\varepsilon + \frac{\partial}{\partial\Sigma^j}y^i\frac{\partial}{\partial\bar{y}^i_\varepsilon}\varepsilon + \frac{\partial}{\partial\Sigma^j}\dot{y}^i\frac{\partial}{\partial\dot{\bar{y}}^i_\varepsilon}\varepsilon \equiv 0. \tag{23}$$

Using (16) we get that equation (23) is just

$$X^1(\Sigma^j) \equiv 0,$$

where X^1 is the first prolongation of the infinitesimal generator of S_ε. Using the coordinate expression (17) of X^1 we get

$$\xi\frac{\partial}{\partial\Sigma^j}x + \eta^i\frac{\partial}{\partial\Sigma^j}y^i + (D\eta^i - \dot{y}^i D\xi)\frac{\partial}{\partial\Sigma^j}\dot{y}^i \equiv 0$$

and substituting the expression (8) of Σ^i in it, we can obtain

$$-\xi\frac{\partial}{\partial\omega^j}x - \eta^i\frac{\partial}{\partial\omega^j}y^i + (D\eta^i - \dot{y}^i D\xi)\delta^j_i \equiv 0$$

that is

$$(D\eta^j - \dot{y}^j D\xi) - \xi\frac{\partial}{\partial\omega^j}x - \eta^i\frac{\partial}{\partial\omega^j}y^i \equiv 0.$$

Writing the coordinate expression for the total derivatives and expressing $\dot{y}^j$ by using (22) we get the following:

Theorem 6. *(Linearised symmetry condition) A vector field $X = \xi\frac{\partial}{\partial}x + \eta^i\frac{\partial}{\partial}y^i$ is an infinitesimal symmetry of the system of first order differential equations (1) if and only if it is a solution of the system of equations (24).*

$$\boxed{\Big(\frac{\partial}{\partial\eta^j}x + \frac{\partial}{\partial\eta^j}y^k\omega^k\Big) - \omega^j\Big(\frac{\partial}{\partial\xi}x + \frac{\partial}{\partial\xi}y^k\omega^k\Big) - \xi\frac{\partial}{\partial\omega^j}x - \eta^i\frac{\partial}{\partial\omega^j}y^i \equiv 0. \quad j = 1, ..., n} \tag{24}$$

Remark: there are summations for the index i and k in the formula.

8 Decomposition of the Lie Algebra of Infinitesimal Symmetries into Trivial and Time-Preserving Subalgebras

As we will see, the Lie algebra of infinitesimal symmetries can be decomposed into the direct sum of two components (Theorem 12). The first component is the trivial one, determined by the direction of the flow of the vector field associated to the system of ODEs. The elements of this component will be called trivial infinitesimal symmetries. The elements of the second component will be called time-preserving infinitesimal symmetries, because their action on the time coordinate is trivial. The elements of this second component can be characterised by the Lie bracket condition (see Lemma 11).

Let us consider the system of first-order ODEs given by (1). The vector field associated with (1) is $Y = \frac{\partial}{\partial}x + \omega^i \frac{\partial}{\partial} y^i$. We have the following observations

Lemma 7. *For an arbitrary function $\xi = \xi(x, y)$, the vector field $\xi Y = \xi \frac{\partial}{\partial} x + \xi \omega^i \frac{\partial}{\partial} y^i$ is an infinitesimal symmetry.*

Proof. One can easily check that (24) is satisfied. □

Lemma 8. *A vector field $X = \eta^i \frac{\partial}{\partial} y^i$ is an infinitesimal symmetry for (1) if and only if*

$$[X, Y] \equiv 0.$$

Proof. $X = \eta^i \frac{\partial}{\partial} y^i$ is an infinitesimal symmetry for (1) if and only if it is a solution of (24). Since for X we have $\xi \equiv 0$, the condition (24) is

$$\Big(\frac{\partial}{\partial \eta^j} x + \frac{\partial}{\partial \eta^j} y^i \omega^i\Big) - \eta^i \frac{\partial}{\partial \omega^j} y^i \equiv 0, \quad j = 1, ..., n \tag{25}$$

On the other hand, computing the Lie bracket of Y and X we have

$$\begin{aligned} [X, Y] &= \Big[\eta^j \frac{\partial}{\partial} y^j, \frac{\partial}{\partial} x + \omega^i \frac{\partial}{\partial} y^i\Big] = \\ &= \eta^j \frac{\partial}{\partial \omega^i} y^j \frac{\partial}{\partial} y^i - \Big(\frac{\partial}{\partial \eta^j} x + \omega^i \frac{\partial}{\partial \eta^j} y^i\Big) \frac{\partial}{\partial} y^i \\ &= \Big(\eta^i \frac{\partial}{\partial \omega^j} y^i - \frac{\partial}{\partial \eta^j} x - \omega^i \frac{\partial}{\partial \eta^j} y^i\Big) \frac{\partial}{\partial} y^j \end{aligned} \tag{26}$$

which shows that the coefficients of $\frac{\partial}{\partial} y^j$ are exactly the functions given by the left-hand side of (25). Therefore we have (25) if and only if we have $[X, Y] \equiv 0$. □

We introduce the following

Definition 9. An infinitesimal symmetry X is called a

1. *trivial infinitesimal symmetry*, if there exists a function $\xi = \xi(x, y)$ such that $X = \xi Y$,
2. *time-preserving infinitesimal symmetry*, if its action on the time coordinate is trivial; that is, for the Lie derivatives of the x-coordinate we have $\mathcal{L}_X x \equiv 0$.

Notation 10. We denote by

- $\mathcal{S}ym(Y)$ the Lie algebra of infinitesimal symmetries of (1) or equivalently, the Lie algebra of infinitesimal symmetries of (6),
- $\mathcal{S}ym_0(Y)$ the Lie subalgebra of time-preserving symmetries.

Lemma 11. *A vector field* $X = \xi\frac{\partial}{\partial}x + \eta^i\frac{\partial}{\partial}y^j$ *is a time-preserving infinitesimal symmetry of (1) if and only if* $\xi \equiv 0$*, and* $[X, Y] \equiv 0$*.*

Proof. If X is a time-preserving infinitesimal symmetry, that is $X \in \mathcal{S}ym_0(Y)$, then $\mathcal{L}_X x \equiv 0$. On the other hand we have $\mathcal{L}_X x = \xi\frac{\partial}{\partial x}x + \eta^i\frac{\partial}{\partial x}y^j = \xi$ which shows that ξ must be identically zero. Moreover, we can get the Lie bracket condition by using Lemma 8. □

Theorem 12. *The Lie algebra of infinitesimal symmetries can be decomposed as a direct sum of trivial infinitesimal symmetries and time-preserving infinitesimal symmetries, that is*[1]

$$\boxed{\mathcal{S}ym(Y) = Span(Y) \oplus \mathcal{S}ym_0(Y)} \tag{27}$$

Proof. Let $X = \xi\frac{\partial}{\partial}x + \eta^i\frac{\partial}{\partial}y^j$ be an infinitesimal symmetry of (1), and set $X_0 := X - \xi Y$. According to Theorem 6, the vector field X is a solution of (24). From Lemma 7, we have that ξY is also a solution of (24). Therefore X_0 is a solution of (24) and therefore an infinitesimal symmetry. Since X_0 has no component in the direction of $\frac{\partial}{\partial}x$ we have $\mathcal{L}_{X_0}x = 0$ and X_0 is therefore a time-preserving infinitesimal symmetry. Hence we get the decomposition of X as

$$X = \xi Y + X_0$$

where $\xi Y \in Span(Y)$ and $X_0 \in \mathcal{S}ym_0(Y)$. □

References

1. Olver, P.J.: *Applications of Lie Groups to Differential Equations*, Springer, 2000.

[1] $Span(Y) = \{fY$, where f is an arbitrary function$\}$

Classification of Semisimple Lie Algebras

Károly Podoski

Institute of Mathematics
University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
k.podoski@gmail.com

Abstract. The general method of Lie algebras is very essential in many parts of mathematics and physics. This talk will give a brief review on the structures of finite-dimensional Lie algebras over the complex field. The theory of semisimple Lie algebras consists of very nice and attractive results utilizing various tools of linear algebra, abstract algebra and graph theory, including the adjoint representation, the Killing form, root space decomposition and Dynkin diagrams.

1 Introduction to Lie Algebras

The most natural Lie algebras can be obtained from associative algebras, e.g. from matrix algebras such that a new bilinear operation called **Lie bracket** or **commutator** is defined: $[x, y] = xy - yx$. Let us start first with a natural example: the general linear algebras and the special linear algebras for 2×2 matrices.

Example $(gl(2,\mathbb{C}), sl(2,\mathbb{C}))$**.** Let the **general linear algebra** $gl(2,\mathbb{C})$ over the complex numbers $\mathbb{C}$ consist of the 2×2 matrices:

$$gl(2,\mathbb{C}) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{C} \right\}. \tag{1}$$

The subalgebra called the **special linear algebra** $sl(2,\mathbb{C})$ consists of the matrices in $gl(2,\mathbb{C})$ with zero trace. Recall that the **trace** is a linear map from the square matrices to the base field, assigning the sum of the diagonal elements to every matrix. Moreover, $\mathrm{Tr}(xy) = \mathrm{Tr}(yx)$, hence $\mathrm{Tr}([x, y]) = 0$ and thus $sl(2,\mathbb{C})$ is closed under the Lie bracket and

$$sl(2,\mathbb{C}) = \left\{ \begin{pmatrix} a & b \\ c & -a \end{pmatrix} : a, b, c, \in \mathbb{C} \right\}. \tag{2}$$

These algebras are vector spaces over the complex field $\mathbb{C}$ and $\dim gl(2,\mathbb{C}) = 4$ with the natural basis

$$\left\{ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right\}. \tag{3}$$

Moreover, $\dim sl(2,\mathbb{C}) = 3$ with the natural basis

$$\left\{ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \right\}. \tag{4}$$

In general, Lie algebras consisting of the matrices with Lie-bracket operator $[x, y] = xy - yx$ are called **linear Lie algebras**. It is possible, however, to define general Lie algebras abstractly using a few axioms.

Definition 1. A vector space L over a field F with a bilinear operation $L \times L \to L$ denoted by $(x, y) \mapsto [x, y]$ (called Lie bracket) is called a **Lie algebra** over F if the following axioms are satisfied:

1. $[x, y] = -[y, x]$ for all $x, y \in L$,
2. $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$ for all $x, y, z \in L$. This is called the **Jacobian identity**.

Note, that we consider the Lie bracket as part of the signature of the algebra. That is, we define subalgebras, ideals, etc. in the usual universal algebraic sense, such that the Lie bracket is considered an operation of the Lie algebra. In particular

Definition 2. A subset S of a Lie algebra L is called a **Lie subalgebra** if S is a subspace which is closed under the Lie bracket.
If S is a subalgebra of L, it is denoted by $S \leq L$.

It is important if a subspace is a subalgebra, or not. Later on, we will decompose Lie algebras into the direct sum of *subspaces*, and we will note that this direct decomposition is *not* a decomposition to subalgebras.

Definition 3. Let $S_1, \ldots, S_n$ be subspaces or subalgebras of a Lie algebra L. Then L is the **direct sum** of S_i, that is

$$L = \oplus_{i=1}^{n} S_i,$$

if and only if every element of L can be uniquely written as $\sum_{i=1}^{n} s_i$, where $s_i \in S_i$.

Homomorphisms are defined in the usual way: as operation-preserving maps from one Lie algebra to another. Let L_1, L_2 be Lie algebras over the field F. That is, a **homomorphism** $\phi\colon L_1 \to L_2$ is a linear map satisfying $\phi([x, y]) = [\phi(x), \phi(y)]$. The image of the homomorphism is $\operatorname{Im}(\phi) = \{\, \varphi(x) \mid x \in L_1 \,\} \subseteq L_2$, the kernel of the homomorphism is $\ker \phi = \{\, x \in L_1 \mid \phi(x) = 0 \,\} \subseteq L_1$.

The image and the kernel of a homomorphism are clearly subalgebras. Moreover, kernels have a special property, namely that if $x \in L_1$, $y \in \ker \phi$ then $[x, y] \in \ker \phi$. Such subalgebras are called ideals:

Definition 4. A subalgebra I of a Lie algebra L is called an **ideal** of L (denoted by $I \lhd L$) if $x \in L$, $y \in I$ implies $[x, y] \in I$.

Thus, every kernel of a homomorphism is an ideal. The converse holds, as well, which is usually proved by constructing the quotient Lie algebra. Let I be an ideal of L. Then the **quotient Lie algebra** L/I is the factor space (as quotient vector space) with elements $\{\, x + I \mid x \in L \,\}$, and let us define the Lie bracket as $[x + I, y + I] = [x, y] + I$. It is easy to see that this definition makes the quotient

vector space a Lie algebra, and the map $L \to L/I$, $x \mapsto x + I$ is a homomorphism (called **natural homomorphism**).

Similarly to other algebraic structures (such as vector spaces, groups, rings) the Homomorphism theorem holds, that is,

Theorem 5 (Homomorphism theorem)**.** *Let L_1, L_2 be Lie algebras over the field F and let $\phi\colon L_1 \to L_2$ be a homomorphism. Then*

$$L_1/\ker\phi \simeq Im(\phi).$$

If ϕ is a bijection then ϕ is called an **isomorphism**. If ϕ is a bijection and $L_1 = L_2$ then ϕ is called an **automorphism**. A **representation** of L is a homomorphism to a linear lie algebra $\phi\colon L \to gl(V)$. Finally, L is the direct sum of subalgebras S_i if and only if L is the direct sum of S_i as subspaces, and each subalgebra S_i is an ideal in L.

Ideals play an important role in the theory of Lie algebras. There are certain subsets which always form an ideal, e.g. the centre or the derived subalgebra.

2 Semisimple Lie-algebras

Definition 6. Let L be an arbitrary Lie algebra. The unique maximal solvable ideal is called the **radical of** L and denoted by $\operatorname{Rad} L$. If L does not contain any non-zero solvable ideal then L is called **semisimple**.

Some properties of semisimple Lie algebras:

1. For an arbitrary Lie algebra L, $L/\operatorname{Rad} L$ is semisimple.
2. An L is semisimple if and only if L is a direct sum of simple Lie algebras.
3. Let L be semisimple. Then there exists ideals $L_1, \dots, L_t$ of L which are simple such that $L = L_1 \oplus \cdots \oplus L_t$. Every simple ideal of L coincides with one of the L_i.

2.1 Cartan decomposition

From now on, **L denotes a (nonzero) semisimple Lie algebra over** $\mathbb{C}$.

Definition 7. An element $x \in L$ is called **semisimple** if ad_x is diagonalizable. A subalgebra of L consisting of only semisimple elements is called **toral subalgebra**.

It is not too hard to show that every toral subalgebra is Abelian.

Lemma 8. *A toral subalgebra of L is Abelian.*

If H is a toral subalgebra of L then $\{ \operatorname{ad}_h \mid h \in H \}$ is a commuting family of semisimple (diagonalizable matrices) endomorphisms of L. According to a well-known fact in linear algebra (namely that commuting linear maps are simultaneously diagonalizable) the elements of $\{ \operatorname{ad}_h \mid h \in H \}$ is simultaneously diagonalizable.

Definition 9. A maximal toral subalgebra H is called a **Cartan subalgebra**.

Some properties of a Cartan subalgebra H:

Although a Cartan subalgebra is not unique, any two Cartan subalgebras are isomorphic to each other by a conjugation.

We can find a basis in L for which every element of $\{ \operatorname{ad}_h \mid h \in H \}$ is diagonalized. For a basis vector $x \in L$ we have $[h, x] = \lambda \cdot x$ (for all $h \in H$), where $\lambda \in \mathbb{C}$ is a scalar depending on $h \in H$. In other words, L is the direct sum of the subspaces

$$L_r = \{ x \in L \mid [h, x] = r(h)x \text{ for all } h \in H \}, \tag{5}$$

where $r \colon H \to \mathbb{C}$ is a linear map.

Definition 10. Let H^* denote the **dual space** of H that is the vector space of all linear functions from H to $\mathbb{C}$

Note that $L_0 = H$, corresponding to the identically 0 function.

Definition 11. The set of all non-zero (linear functions) $r \in H^*$ for which $L_r \neq 0$ is denoted by Φ. The elements of Φ are called **roots** and Φ is called a **root system**.

4. If L is semisimple, then we have **root space decomposition** or **Cartan decomposition**; that is, L is the direct sum (**as subspaces**) of H and of the subspaces L_r:

$$L = H \oplus (\oplus_{r \in \Phi} L_r).$$

H is an Abelian subalgebra called **Cartan subalgebra**

Example $(sl(2, \mathbb{C}))$**.** Let $L = sl(2, \mathbb{C})$. Then

$$H = \left\{ \begin{pmatrix} \alpha & 0 \\ 0 & -\alpha \end{pmatrix} : \alpha \in \mathbb{C} \right\}$$

is a Cartan subalgebra. Note that every Cartan subalgebra can be written as $P^{-1}HP$ where P is an invertible matrix of size 2×2.

Let $h = \begin{pmatrix} \alpha & 0 \\ 0 & -\alpha \end{pmatrix} \in H$ and $x = \begin{pmatrix} a & b \\ c & -a \end{pmatrix}$. Now,

$$[h, x] = \begin{pmatrix} 0 & 2\alpha b \\ -2\alpha c & 0 \end{pmatrix}.$$

That is, ad_h (for $h \in H$) is diagonal in the basis $e_{11} - e_{22}, e_{12}, e_{21}$:

$$ad_h = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2\alpha & 0 \\ 0 & 0 & -2\alpha \end{pmatrix}.$$

Consequently, the Cartan decomposition of L is

$$L = H \oplus L_r \oplus L_{-r},$$

where

$$L_r = \{\, \lambda e_{12} \mid \lambda \in \mathbb{C} \,\},$$
$$L_{-r} = \{\, \lambda e_{21} \mid \lambda \in \mathbb{C} \,\},$$

and

$$r \colon H \to \mathbb{C} \qquad r(h) = r\left(\begin{pmatrix} \alpha & 0 \\ 0 & -\alpha \end{pmatrix} \right) = 2\alpha.$$

Now, $\dim H = 1$, $\dim L_r = 1$, $\dim L_{-r} = 1$ and the root system is $\Phi = \{\, r, -r \,\}$.

2.2 Semisimplicity by the Killing form

To classify simple Lie algebras we will need a bilinear form with an unusual associative property.

Definition 12. Let L be any Lie algebra.
If $x, y \in L$, define $\kappa(x, y) = \mathrm{Tr}\,(\mathrm{ad}_x \,\mathrm{ad}_y)$. Then κ is a symmetric bilinear form on L, called the **Killing form**.

Lemma 13. *The Killing form κ is associative in the sense that $\kappa\,([x, y]\,, z) = \kappa\,(x, [y, z])$.*

Theorem 14. *Let L be a Lie algebra. Then L is semisimple if and only if its Killing form is nondegenerate.*

2.3 Properties of a root space decomposition

Let L be a semisimple Lie algebra and take its Cartan decomposition:

$$L = H \oplus (\oplus_{r \in \Phi} L_r),$$

where Φ is the corresponding root system.

Since the restriction of the Killing form κ to H is nondegerate we can identify H with H^*: to every $r \in H^*$ there corresponds a unique element $t_r \in H$ satisfying

$$r(h) = \kappa(t_r, h) \text{ for all } h \in H.$$

Now, we can introduce a scalar product $(.,.)$ on H^* such that

$$(r, s) = \kappa(t_r, t_s) \text{ for all } r, s \in H^*.$$

Definition 15. A root system Φ is called **indecomposable** (or **irreducible)** if it cannot be partitioned into the union of two proper subsets such that each root in one set is orthogonal to every root in the other set by the scalar product $(.,.)$.

Lemma 16. *A semisimple L Lie algebra is simple if and only if the corresponding root system Φ is indecomposable.*

In the next section we will define the root systems in an abstract way using five axioms. The main advantage of this definition is that the simple (or semisimple) Lie algebras can be classified via abstract root systems and their properties.

3 Abstract root systems

Let V be a real Eucliden space of finite dimension l with scalar product $(.,.)$.

Let w_r denote the reflection to the hyperplane orthogonal to the non-zero vector r. That is, for $x \in V$ we have

$$w_r(x) = x - \frac{2(r,x)}{(r,r)}r.$$

Now, $w_r(r) = -r$ and $w_r(y) = y$ for all y with $(r,y) = 0$.

1. Φ is a set of non-zero vectors.
2. Φ spans V.
3. If $r, s \in \Phi$ then $w_r(s) \in \Phi$.
4. If $r, s \in \Phi$ then $2(r,s)/(r,r)$ is an integer.
5. If $r, \lambda r \in \Phi$, where $\lambda \in \mathbb{R}$ then $\lambda = \pm 1$.

The **rank of a root system** Φ is the dimension of V (spanned by Φ): $\operatorname{rank}\Phi = \dim V = l$.

3.1 Properties of a root system

Let Φ be a root system. For all $r, s \in \Phi$ let

$$A_{sr} = \frac{2(r,s)}{(s,s)} = 2\frac{\|r\|}{\|s\|}\cos\theta.$$

Now, A_{sr} and A_{rs} are integers having the same signs and $A_{sr}A_{rs} = 4\cos^2\theta$. This only leaves a few possibilities for A_{sr} and A_{rs}. Suppose that $r \neq \pm s$ and $\|r\| \geq \|s\|$ (if $\pm r \neq s$ then r and s are called **nonproportional roots**). Then $|A_{rs}|$ is either 0 or 1, moreover $A_{rs} \cdot A_{sr} = 4\cos^2\theta < 4$. If $A_{rs} = 0$, then $\cos\theta = 90\circ$, and therefore $A_{sr} = 0$, as well. If $|A_{rs}| = 1$, then A_{sr} has the same sign with $|A_{sr}| \leq 3$. Consequently, there are only a finite number of possibilities for the degrees of two vectors, and for the ratio of their lengths. In each case one can then generate the corresponding rank 2 root system by applying the appropriate reflections after each other. The rank 2 root systems can be found in Figure 1.

6. If $A_{rs} = 0$ then $A_{sr} = 0$ and $\theta = 90^o$, the corresponding root system is $A_1 \times A_1$.
7. If $A_{rs} = 1$ and $A_{sr} = 1$ then $\theta = 60^o$ and $\|r\| = \|s\|$, the corresponding root system is A_2.

8. If $A_{rs} = 1$ and $A_{sr} = 2$ then $\theta = 45^o$ and $\|r\| = \sqrt{2}\,\|s\|$, the corresponding root system is B_2.
9. If $A_{rs} = 1$ and $A_{sr} = 3$ then $\theta = 30^o$ and $\|r\| = \sqrt{3}\,\|s\|$, the corresponding root system is G_2.
10. If $A_{rs} = -1$ and $A_{sr} = -1$ then $\theta = 120^o$ and $\|r\| = \|s\|$, the corresponding root system is A_2.
11. If $A_{rs} = -1$ and $A_{sr} = -2$ then $\theta = 135^o$ and $\|r\| = \sqrt{2}\,\|s\|$, the corresponding root system is B_2.
12. If $A_{rs} = -1$ and $A_{sr} = -3$ then $\theta = 150^o$ and $\|r\| = \sqrt{3}\,\|s\|$, the corresponding root system is G_2.

Classifying root systems of rank 2 we obtain only four different root systems : $A_1 \times A_1$ is a reducible root system, A_2, B_2 and G_2 are irreducible root systems.

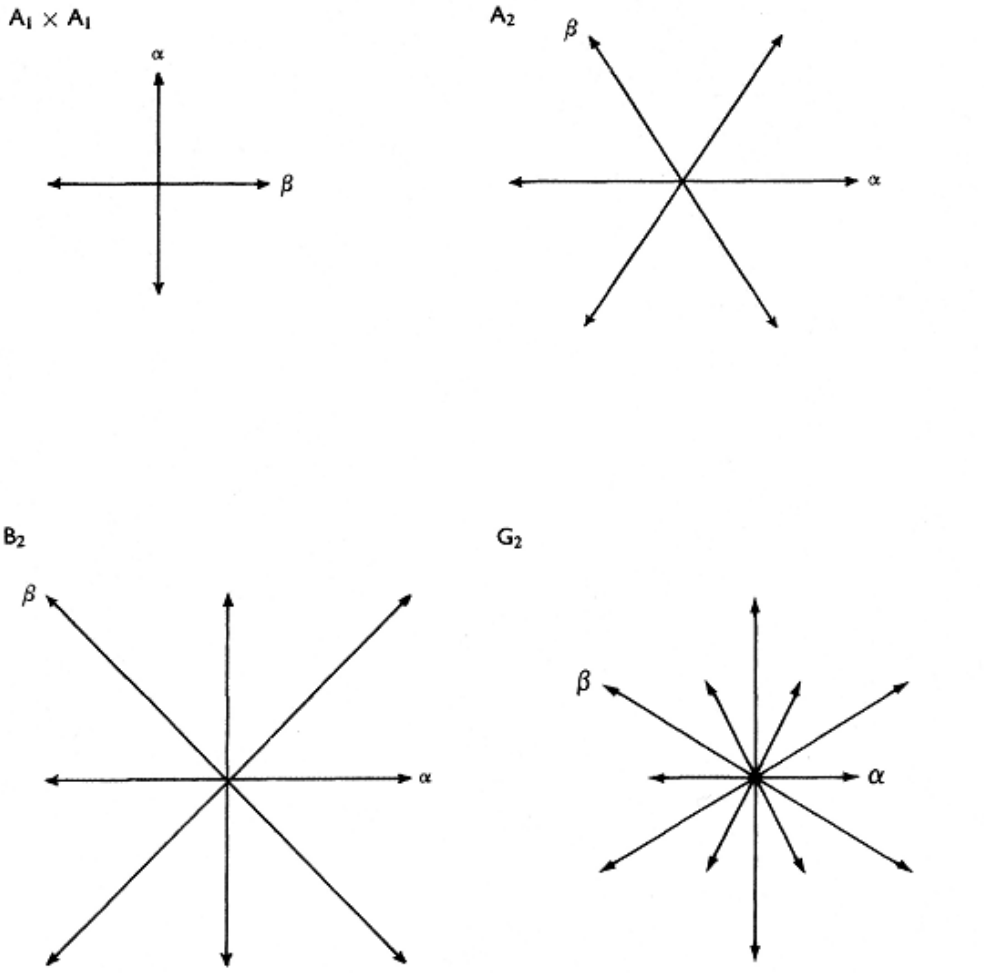


Fig. 1: Rank two root systems

Lemma 17. *Let r, s be nonproportional roots. If $(r, s) > 0$ then $r - s$ is a root. If $(r, s) < 0$ then $r + s$ is a root.*

Theorem 18 (Existence theorem)**.** *Let Φ be an indecomposable root system. Then there exists a simple Lie algebra over $\mathbb{C}$ which has a root system equivalent to Φ.*

Theorem 19 (Isomorpism theorem)**.** *Any two simple Lie algebras over $\mathbb{C}$ with equivalent root systems are isomorphic.*

3.2 Fundamental root systems (or bases)

Definition 20. Let Φ be a system of roots. A subset $\Pi \subset \Phi$ is called a **fundamental root system (or base)** if the following axioms are satisfied.

1. Π is linearly independent.
2. Every root in Φ is a linear combination of roots in Π with coefficients which are either all non-negative or all non-positive.

Lemma 21. *If Π is a fundamental system in Φ then $(r, s) \leq 0$ for all different elements r, s in Π.*

Theorem 22. *Every root system contains several fundamental systems.*[1] *If Π_1 and Π_2 are fundamental systems in Φ then $|\Pi_1| = |\Pi_2| = l$. This number is called the* ***rank of the root system****. Moreover, Π_1 is equivalent to Π_2.*

Let Π be a fundamental system in Φ. It is easy to see that Π is indecomposable if and only if Φ is indecomposable.

4 Coxeter graphs and Dynkin diagrams

Definition 23. Let Φ be a root system and $\Pi = \{ r_1, r_2, \ldots, r_l \}$ be a fundamental system in Φ. Let the **Coxeter graph of Φ** be a graph having l vertices, such that the ith vertex is joined to the jth vertex (for $i \neq j$) by $A_{r_i r_j} A_{r_j r_i}$-many edges.

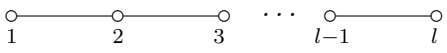
Then it is easy to see that

3. the Coxeter graph of a root system Φ is connected if and only if Φ is indecomposable.

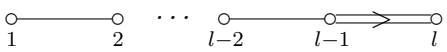
Definition 24. The **Dynkin diagram of a root sytem** Φ can be obtained from the Coxeter graph of Φ such that whenever a double or a triple edge occurs then an arrow is added pointing to the shorter of the two roots. The matrix consisting of the elements A_{rs} for every $r, s \in \Pi$ is called the **Cartan matrix**.

Theorem 25. *If Φ is an indecomposable root system of rank l its Dynkin diagram is one of the following (l vertices in each cases):*

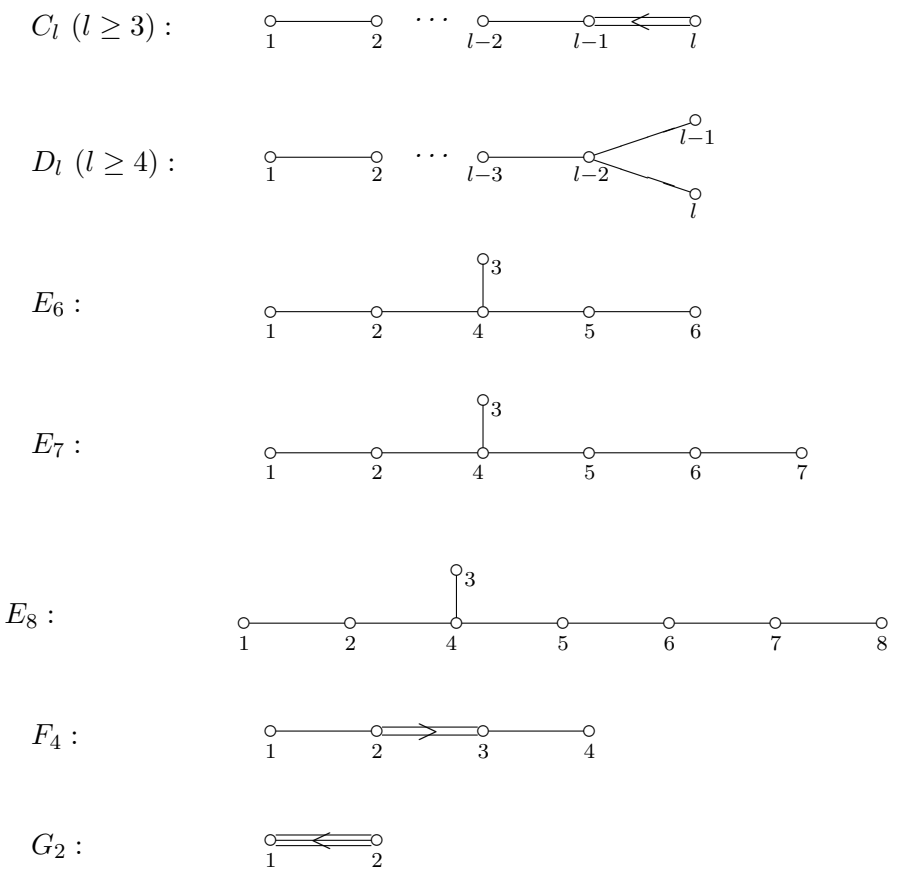
A_l $(l \geq 1)$:



B_l $(l \geq 2)$:



[1] If Π_1 and Π_2 are fundamental systems then there exist reflections $w_{r_1}, \ldots, w_{r_t}$ such that $\Pi_2 = w_{r_1}(\ldots(w_{r_t}(\Pi_1)\ldots)$, where $r_1, \ldots, r_t \in \Pi_1$.



References

1. R. W. Carter. *Simple groups of Lie type*. Wiley Classics Library. John Wiley & Sons Inc., New York, 1989. Reprint of the 1972 original, A Wiley-Interscience Publication.
2. J. E. Humphreys. *Introduction to Lie algebras and representation theory*, volume 9 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1978. Second printing, revised.
3. D. J. S. Robinson. *A course in the theory of groups*, volume 80 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition, 1996.

Octonions

Ágota Figula

Institute of Mathematics
University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
figula@science.unideb.hu

Abstract. In this talk I would like to describe the algebra $\mathbb{O}$ of octonions or Cayley numbers. $\mathbb{O}$ is an 8-dimensional real algebra which can be constructed from the skew field of quaternions using an involutory antiautomorphism $a \mapsto \bar{a}$, called *conjugation* (Cayley-Dickson process). $\mathbb{O}$ is non-associative, but alternative, biassociative division algebra such that the Lie algebra of its derivations forms a 14-dimensional simple Lie algebra A. Using the classification of the simple Lie algebras, A is isomorphic to the Lie algebra of the exceptional compact Lie group G_2.

Key words and phrases: the classical division algebras, group of automorphisms of these algebras, orthogonal groups, the exceptional simple Lie group G_2

1 Introduction

Within the frame of the BIOMICS project we have studied the simple Lie algebras over an algebraically closed field of characteristic 0. Our point of view was to investigate their root systems and Dynkin diagrams. In particular, we have dealt with the exceptional Lie algebra $\mathbf{g}_2$. This Lie algebra has dimension 14 and it has a strong connection to the algebra $\mathbb{O}$ of octonions or Cayley numbers. Namely, the set of derivations of $\mathbb{O}$ forms a simple Lie algebra isomorphic to $\mathbf{g}_2$ (cf. [3], [5]).

Here we wish to follow the discussion of [7] about the classical division algebras. First we explain the Cayley-Dickson process which is an inductive construction of a sequence $\mathbb{F}_m$, $m \in \mathbb{N} \cup \{0\}$, of real algebras. This construction describes the relations between the fields $\mathbb{F}_0 = \mathbb{R}$ of real numbers, $\mathbb{F}_1 = \mathbb{C}$ of complex numbers, the skew field $\mathbb{F}_2 = \mathbb{H}$ of quaternions and the non-associative algebra $\mathbb{F}_3 = \mathbb{O}$ of octonions. The complex numbers can be represented as ordered pairs of real numbers, quaternions can be viewed as ordered pairs of complex numbers and octonions can be written as ordered pairs of quaternions, such that the addition is the usual vector addition, each algebra has an involutory antiautomorphism, called conjugation, and the multiplication is defined in a uniform way given in (1). We shows that the algebras $\mathbb{F}_m$, $m \leq 3$, are normed quadratic division algebras. Among these algebras only the algebra of octonions does not hold the associative law. But as a consequence of alternativity it is biassociative, i.e. every two elements of $\mathbb{O}$ belong to an associative subalgebra containing 1.

Following [7] in section 3 we deal with the group Aut $(\mathbb{F})$ of automorphisms of the algebras $\mathbb{F} = \{\mathbb{H}, \mathbb{O}\}$. The Lie algebra of Aut $(\mathbb{F})$ is precisely the Lie algebra of the derivations of $\mathbb{F}$. Using the multiplication of the algebras $\mathbb{F}_m$, $1 \leq m \leq 3$, we describe the orthogonal groups $O_n(\mathbb{R})$, $SO_n(\mathbb{R})$, $n = 2^m$, as the group of $\mathbb{R}$-linear transformations of $\mathbb{F}_m$ which preserve the norm form. We show that the group of automorphisms of $\mathbb{H}$ coincides with the group of all inner automorphisms of $\mathbb{H}$, both are the group $SO_3(\mathbb{R})$. Every automorphism of $\mathbb{F}_m$, $m \in \{2, 3\}$, leaves invariant the subspace $\mathrm{Pu}\mathbb{F}_m$ of $\mathbb{F}_m$ orthogonal to $\mathbb{R}$, commutes with conjugation and is orthogonal with respect to the inner product of $\mathbb{F}_m$. To investigate the group Aut $(\mathbb{O})$ first we prove some important transitivity property of Aut $(\mathbb{O})$ and show that it is generated by involutions. After this we prove that the group Aut $(\mathbb{O})$ is a 14-dimensional simple Lie subgroup of $O_8(\mathbb{R})$. Using the classification of simple Lie groups it turns out that Aut $(\mathbb{O})$ is the exceptional compact Lie group G_2.

2 The Classical Division Algebras

Here we apply the Cayley-Dickson process to the field $\mathbb{R}$ of real numbers in order to construct the field $\mathbb{C}$ of complex numbers, the skew field $\mathbb{H}$ of quaternions and the non-associative field $\mathbb{O}$ of octonions. They are called the classical division algebras. We characterize them.

The Cayley-Dickson construction produces a sequence $\mathbb{F}_m$ of algebras over $\mathbb{R}$ such that each of them has an involutory antiautomorphism $a \mapsto \bar{a}$, called *conjugation*. One starts with $\mathbb{F}_0 = \mathbb{R}$, $\overline{a} = a$ for $a \in \mathbb{R}$. We assume that $\mathbb{F}_{m-1}$, $m \geq 1$, with its conjugation has been constructed. One puts $\mathbb{F}_m := \mathbb{F}_{m-1} \times \mathbb{F}_{m-1}$ and defines the addition, the multiplication and the conjugation in $\mathbb{F}_m$ as follows:

$$(a,b) + (c,d) := (a+c, b+d)$$

$$(a,b)(c,d) := (ac - \overline{d}b, da + b\overline{c}) \tag{1}$$
$$\overline{(a,b)} := (\overline{a}, -b),$$

where $a, b, c, d \in \mathbb{F}_{m-1}$. The dimension of $\mathbb{F}_m$ over $\mathbb{R}$ is 2^m. The map $\mathbb{F}_{m-1} \to \mathbb{F}_m : a \mapsto (a, 0)$ shows that $\mathbb{F}_{m-1}$ may be identified with a subalgebra of $\mathbb{F}_m$. $\mathbb{R} = \mathbb{F}_0$ is a central subfield of all the algebras $\mathbb{F}_m$. The conjugation is an antiautomorphism of $\mathbb{F}_m$ and the set of its fixed elements is $\{x \in \mathbb{F}_m |\ \bar{x} = x\} = \mathbb{R}$. The algebra $\mathbb{F}_1 := \mathbb{C}$ of complex numbers is commutative and associative. The algebra $\mathbb{F}_2 := \mathbb{H}$ of Hamilton's *quaternions* is associative. For $m = 3$ we get the algebra of *octonions* (or Cayley numbers) $\mathbb{F}_3 := \mathbb{O}$. It is not associative, but alternative and consequently biassociative.

Proposition 1. *For $m \leq 3$ the algebra $\mathbb{F}_m$ has no zero divisors and therefore it is a division algebra in the sense that, for $a \neq 0$, the $\mathbb{R}$-linear maps $x \mapsto ax$ and $x \mapsto xa$ are bijective.*

This assertion follows from the fact that for $m \leq 3$ the algebra $\mathbb{F}_m$ is alternative. To the proof of this fact we introduce the notion of the norm of the algebras $\mathbb{F}_m$. For $x = (a,b) \in \mathbb{F}_m$, $a, b \in \mathbb{F}_{m-1}$ one has

$$x\overline{x} = (a,b)(\overline{a}, -b) = (a\overline{a} + b\overline{b}, 0),\ \ \overline{x}x = (\overline{a}, -b)(a,b) = (\overline{a}a + b\overline{b}, 0).$$

By induction on m, one gets $x\overline{x} = \overline{x}x$ and for $x \neq 0$ this is a positive element $||x||^2$ of the subfield $\mathbb{R}$. The map $x \mapsto ||x||^2$ is a positive definite quadratic form (the so-called *norm form*) on the $\mathbb{R}$-vector space $\mathbb{F}_m$. The associated bilinear form is $\langle x|y\rangle := ||x+y||^2 - ||x||^2 - ||y||^2 = \overline{x}y + \overline{y}x$. It is a positive definite inner product such that $\langle x|x\rangle = 2||x||^2$. Since conjugation is involutory we obtain $||\bar{x}||^2 = ||x||^2$, $\langle \bar{x}|\bar{y}\rangle = \langle x|y\rangle$. Moreover, for $a, b \in \mathbb{F}_{m-1}$ we have $||(a,b)||^2 = ||a||^2 + ||b||^2$. The subspaces $\mathbb{F}_{m-1} \times \{0\}$ and $\{0\} \times \mathbb{F}_{m-1}$ of $\mathbb{F}_m$ are orthogonal with respect to the inner product. For $0 \neq x \in \mathbb{F}_m$ we obtain that $x^{-1} := ||x||^{-2}\bar{x}$ is a two-sided multiplicative inverse and that $||x^{-1}||^2 = (||x||^2)^{-1}$.

$\mathbb{F}_m$ is a **quadratic algebra**, i.e. every element of $\mathbb{F}_m$ satisfies a quadratic equation with real coefficients. For every $x \in \mathbb{F}_m$

$$x + \bar{x} \in \mathbb{R} \text{ so that } \bar{x} \in \mathbb{R} + \mathbb{R}x.$$

Consequently, $x^2 = (x + \bar{x})x - ||x||^2 \in \mathbb{R} + \mathbb{R}x$.

The orthogonal complement of the subspace $\mathbb{R}$ of $\mathbb{F}_m$ with respect to the inner product is the subspace

$$\mathrm{Pu}\mathbb{F}_m := \{x \in \mathbb{F}_m | \bar{x} = -x\}$$

of *pure* elements. From the above quadratic equation it follows that this subspace can be written as

$$\mathrm{Pu}\mathbb{F}_m := \{x \in \mathbb{F}_m |\ x^2 \in \mathbb{R},\ x^2 = -||x||^2 \leq 0\}. \tag{2}$$

For $x \in \mathbb{F}_m$ the equality

$$x^2 = -||x||^2 \text{ holds if and only if } x \in \mathrm{Pu}\mathbb{F}_m. \tag{3}$$

For $u, v \in \mathrm{Pu}\mathbb{F}_m$ we have

$$\langle u|v\rangle = 0 \text{ precisely if } uv = -vu \text{ precisely if } uv \in \mathrm{Pu}\mathbb{F}_m. \tag{4}$$

Now we deal with the associative properties of the algebras $\mathbb{F}_m$. Every algebra $\mathbb{F}_m$, $m \geq 0$, is *mono-associative.*

Proposition 2. *For $x \in \mathbb{F}_m \setminus \mathbb{R}$, the span $\mathbb{R}+\mathbb{R}x$ of 1 and x is an associative and commutative subalgebra of $\mathbb{F}_m$ and this is isomorphic to $\mathbb{C}$.*
Proof. The intersection of the 2-dimensional subspace $A = \mathbb{R} + \mathbb{R}x$ with the hyperplane $\mathrm{Pu}\mathbb{F}_m$ is a 1-dimensional subspace $\mathbb{R}u$ with $u \in \mathrm{Pu}\mathbb{F}_m$ and $||u||^2 = 1$. Hence $\{1, u\}$ is a basis of A and $u^2 = -||u||^2 = -1 \in \mathbb{R}$ (cf. (2)). Hence A is a subalgebra which is associative and commutative. It is isomorphic to $\mathbb{C}$. □

To get stronger associative properties of $\mathbb{F}_m$ for $m \leq 3$ we use alternativity.

Proposition 3. *Let $\mathbb{F}_{m-1}$ be an associative algebra. Then for all $x, y \in \mathbb{F}_m$ one has*

$$\bar{x}(xy) = (\bar{x}x)y = ||x||^2 y \tag{5}$$

and by conjugation

$$(yx)\bar{x} = y(x\bar{x}) = ||x||^2 y. \tag{6}$$

Equivalently, $\mathbb{F}_m$ is alternative, i.e. the following identities hold:

$$x(xy) = x^2 y \quad \text{and} \quad (yx)x = yx^2.$$

As a consequence of alternativity of $\mathbb{F}_m$ it is flexible, i.e. the following identities hold: $x(yx) = (xy)x$.
Proof. Let $x = (a, b)$, $y = (c, d)$ for $a, b, c, d \in \mathbb{F}_{m-1}$. Direct computation and the associativity of $\mathbb{F}_{m-1}$ yield that

$$\bar{x}(xy) = (\bar{a}ac - \bar{a}\bar{d}b + \bar{a}\bar{d}b + c\bar{b}b, da\bar{a} + b\bar{c}\bar{a} - b\bar{c}\bar{a} + b\bar{b}d).$$

As $a\bar{a} = \bar{a}a = ||a||^2$ and $b\bar{b} = \bar{b}b = ||b||^2$ are scalars and $||a||^2 + ||b||^2 = ||x||^2$ we get $\bar{x}(xy) = (||a||^2 c + ||b||^2 c, ||a||^2 d + ||b||^2 d) = ||x||^2(c, d) = ||x||^2 y$ which proves (5). Applying conjugation (6) follows. The alternative laws are identical to (5) and (6) if x is a pure element, i.e. if $x = -\bar{x}$. Decomposing $x \in \mathbb{F}_m$ into a scalar in $\mathbb{R}$ and a pure element the general case can be proved.
The flexibility of $\mathbb{F}_m$ follows from alternativity by computing the two sides of the equation $(x + y)((x + y)x) = (x + y)^2 x$. □

Proposition 4. Biassociativity. *For all $x \in \mathbb{O} \setminus \mathbb{R}$ and $y \in \mathbb{O} \setminus (\mathbb{R} + \mathbb{R}x)$, the span $\mathbb{R} + \mathbb{R}x + \mathbb{R}y + \mathbb{R}xy$ is an associative subalgebra of $\mathbb{O}$ isomorphic to $\mathbb{H}$.*

Proof. Intersecting $\mathbb{R} + \mathbb{R}x$ and $\mathbb{R} + \mathbb{R}x + \mathbb{R}y$ with the hyperplane $\mathrm{Pu}\mathbb{O}$, we obtain elements $u, v \in \mathrm{Pu}\mathbb{O}$ for which $||u||^2 = 1 = ||v||^2$ and $\langle u|v\rangle = 0$ such that $x = r_0 + r_1 u$, $y = s_0 + s_1 u + s_2 v$ for suitable scalars $r_\nu, s_\nu \in \mathbb{R}$, $\nu = 0, 1, 2$ with $r_1 \neq 0$, $s_2 \neq 0$. By (2) we have $u^2 = -||u||^2 = -1$ and therefore $\mathbb{R} + \mathbb{R}x + \mathbb{R}y + \mathbb{R}xy = \mathbb{R} + \mathbb{R}u + \mathbb{R}v + \mathbb{R}uv$. Hence it suffices to prove the following assertion:

Proposition 5. *Assume that* $2 \leq m \leq 3$ *and let* $u, v \in Pu\mathbb{F}_m$ *be such that*

$$||u||^2 = 1 = ||v||^2 \quad \textit{and} \quad \langle u|v\rangle = 0. \tag{7}$$

Then the product $w := uv$ *satisfies*

$$w \in Pu\mathbb{F}_m, \quad \langle u|w\rangle = 0 = \langle v|w\rangle, \ \textit{and} \ ||w||^2 = 1. \tag{8}$$

Moreover, we have the following multiplication table:

	u	v	w
u	-1	w	$-v$
v	$-w$	-1	u
w	v	$-u$	-1.

The span $\mathbb{R} + \mathbb{R}u + \mathbb{R}v + \mathbb{R}w$ *is an associative subalgebra of* $\mathbb{F}_m$ *isomorphic to* $\mathbb{H}$.

Definition 6. A triple u, v, w with these properties is called a *Hamilton triple.*

Proof. At a first stage we shall assume in addition that $\mathbb{F}_m$, $m = 2, 3$, is already known to be alternative. This assumption will later prove. From (7), (2) and (4) we get $u^2 = -1 = v^2$ and $w \in \mathrm{Pu}\mathbb{F}_m$. Applying alternativity one obtains $uw = u(uv) = u^2v = -v$ and $wv = (uv)v = uv^2 = -u$. By (4) it follows that u, v and w are mutually orthogonal with respect to $\langle \ | \ \rangle$ and anticommuting. Using alternativity we get that $w^2u = w(wu) = wv = -u$. Since w^2 is a scalar by (3), this means that $w^2 = -1$ and $||w||^2 = 1$. Thus the multiplication table is proved and it follows that

$$A := \mathbb{R} + \mathbb{R}u + \mathbb{R}v + \mathbb{R}w$$

is a flexible subalgebra.
To prove the associativity of A it suffices to show that triple products composed of u, v and w are associative. If two of the factors of such a triple product coincide, then this is true by alternativity and flexibility. For triple products with different factors we have, for instance,

$$(uv)w = w^2 = -1 = u^2 = u(vw) \ \text{ and } \ (vu)w = -w^2 = -1 = -v^2 = v(uw).$$

All the other triple products with different factors are obtained from these by cyclic permutation of u, v and w under which the corresponding equalities remain valid because of the symmetry of the multiplication table. Thus A is associative.
Now we show our supplementary hypothesis of alternativity. Since $\mathbb{F}_1 = \mathbb{C}$ is associative, $\mathbb{F}_2$ is alternative by Proposition 3 and so the above arguments are valid in $\mathbb{F}_2$. In this case $1, u, v, w$ span the 4-dimensional algebra $\mathbb{F}_2 = \mathbb{H}$ since they are linearly independent by (7) and (8), thus $\mathbb{F}_2 = A$. In particular, from the above, $\mathbb{F}_2 = \mathbb{H}$ is associative.
Proposition 3 yields that also $\mathbb{F}_3 = \mathbb{O}$ is alternative. Thus our previous arguments apply here we get that the subalgebra A spanned by 1 and by elements u, v, w with the properties (7) and (8) is associative. Any two such algebras are isomorphic since multiplication is entirely determined by the given multiplication table. As $\mathbb{H}$ is also spanned by such a basis every such subalgebra of $\mathbb{F}_3 = \mathbb{O}$ is isomorphic to $\mathbb{H}$. □

Since there are multiplicative inverses of the algebras $\mathbb{F}_m$ we obtain that $\mathbb{C}$ is a commutative field and $\mathbb{H}$ is a skew field. As $\mathbb{F}_3 = \mathbb{O}$ is alternative the assertion of Proposition 1 follows because if $xy = 0$, then $||x||^2 y = \bar{x}(xy) = 0$ so that $x = 0$ or $y = 0$.

Summarizing this discussion we obtain:

Proposition 7. $\mathbb{O}$ *is an alternative field, i.e. an alternative division algebra and it is biassociative.*

Proposition 8. Multiplicativity of the norm. *If* $m \leq 3$*, then for* $x, y \in \mathbb{F}_m$ *one has*

$$||xy||^2 = ||x||^2||y||^2. \tag{9}$$

Moreover, for each $a \in \mathbb{F}_m$*,*

$$\langle x|\bar{a}y\rangle = \langle ax|y\rangle \quad \textit{and} \quad \langle x|y\bar{a}\rangle = \langle xa|y\rangle. \tag{10}$$

Proof. All the algebras $\mathbb{F}_m$, $m \leq 3$ are at least biassociative (cf. Proposition 4). Hence one has $||xy||^2 = \overline{(xy)}(xy) = \bar{y}(\bar{x}x)y = ||x||^2\bar{y}y = ||x||^2||y||^2$ which proves (9). For (10) we may assume $a \neq 0$. Put $z = a^{-1}y$, so that $az = a(a^{-1}y) = y$. Using (9), we obtain $\langle ax|y\rangle = \langle ax|az\rangle = ||ax + az||^2 - ||ax||^2 - ||az||^2 = ||a||^2(||x + z||^2 - ||x||^2 - ||z||^2) = ||a||^2\langle x|z\rangle = \langle x|||a||^2 z\rangle = \langle x|(\bar{a}a)z\rangle = \langle x|\bar{a}(az)\rangle = \langle x|\bar{a}y\rangle$. The first equality of (10) is thus proved. The second is obtained by applying conjugation which preserves the inner product (see the properties of the *norm form*). □

The following result is a motivation of the Cayley-Dickson process.

Proposition 9. Constructing $\mathbb{O}$ from the quaternion subfields. *Let* H *be a subalgebra of* $\mathbb{O}$ *isomorphic to* $\mathbb{H}$ *and let* $z \in Pu\mathbb{O}$ *be of unit length* $||z||^2 = 1$ *and orthogonal to* H*. Then the* 4*-dimensional* $\mathbb{R}$*-linear subspace* Hz *is orthogonal to* H *so that the* $\mathbb{R}$*-vector space* $\mathbb{O}$ *decomposes into the direct sum*

$$\mathbb{O} = H \oplus Hz.$$

For $a, b, c, d \in H$ *we have* $(a + bz)(c + dz) = (ac - \bar{d}b) + (da + b\bar{c})z$*.*
Proof. H is invariant under conjugation (see the part $\mathbb{F}_m$ is a **quadratic algebra**). By (10) we have $\langle a|bz\rangle = \langle \bar{b}a|z\rangle = 0$. Hence H and Hz are orthogonal. Using alternativity we obtain for $x, y, w \in \mathbb{O}$,

$$||x + y||^2 w = (x + y)((\bar{x} + \bar{y})w) = (||x||^2 + ||y||^2)w + x(\bar{y}w) + y(\bar{x}w).$$

Now $||x + y||^2 = ||x||^2 + ||y||^2$ whenever $\langle x|y\rangle = 0$. Hence if $\langle x|y\rangle = 0$, then $x(\bar{y}w) = -y(\bar{x}w)$. For $w = 1$ if $y \in \mathrm{Pu}\mathbb{O}$ and $\langle x|y\rangle = 0$, then $xy = y\bar{x}$. Using these and alternativity we may compute the terms on the left hand side of the asserted product formula as follows.

$$(bz)c = \bar{c}(bz) = \bar{c}(z\bar{b}) = -\bar{z}(c\bar{b}) = -(b\bar{c})\bar{z} = (b\bar{c})z,$$

$$a(dz) = a(z\bar{d}) = -\bar{z}(\bar{a}\bar{d}) = z(\bar{d}a) = (da)z,$$

$$(bz)(dz) = -\bar{d}((\bar{z}\bar{b})z) = -\bar{d}((b\bar{z})z) = -\bar{d}((b||z||^2) = -\bar{d}b.$$

□

Definition 10. *Cayley triples* are triples $u, v, z \in \mathrm{Pu}\mathbb{O}$, $||u||^2 = ||v||^2 = ||z||^2 = 1$ such that u and v are mutually orthogonal and such that z is orthogonal to u, v and uv.

This notion is symmetric in the sense that a permutation of a Cayley triple is another Cayley triple. For example by Proposition 9 and Proposition 5 we obtain the following identities: $u(vz) = (vu)z = -(uv)z$ (which shows that $\mathbb{O}$ is non-associative), $(vz)u = -(vu)z$. Thus one gets $u(vz) = -(vz)u$. From (4) we get that the pure element vz is orthogonal to u. By Proposition 5 and Proposition 4 for a given Cayley triple u, v, z the pure octonions u, v and $w := uv$ form a Hamilton triple and together with 1 they span a subalgebra H isomorphic to $\mathbb{H}$. Furthermore, by Proposition 9 the elements 1, u, v, w, z,

uz, vz, wz constitute a basis of $\mathbb{O}$ as a vector space over $\mathbb{R}$. From Proposition 5 and Proposition 9 one can compute a multiplication table for this basis. The following assertion is true.

For all Cayley triples u, v, z one obtains the same multiplication table with respect to the basis 1, u, v, w, z, uz, vz, $(uv)z$.

This shows that for any two Cayley triples the $\mathbb{R}$-linear transformation of $\mathbb{O}$ mapping the basis obtained in this way from the first Cayley triple onto the basis corresponding to the second Cayley triple is an automorphism of $\mathbb{O}$. Hence we get the following:

Corollary 11. *For any two Cayley triples of $\mathbb{O}$, there is a unique automorphism of $\mathbb{O}$ mapping the first Cayley triple onto the second.*

The *standard Cayley triple* and the associated basis of $\mathbb{O}$ are obtained as follows. Let $i \in \mathbb{C}$ be an imaginary unit, that is $i \in \mathrm{Pu}\mathbb{C}$ with $i^2 = -1$. In $\mathbb{H} = \mathbb{C} \times \mathbb{C}$ one considers the basis

$$1 = (1,0), \quad i = (i,0), \quad j := (0,1), \quad k := ij = (i,0)(0,1) = (0,i).$$

The triple i, j, k is clearly a Hamilton triple as in Proposition 5. In $\mathbb{O} = \mathbb{H} \times \mathbb{H}$ by identify $\mathbb{H}$ with the subalgebra $\mathbb{H} \times \{0\}$ we find these elements again, namely

$$i \hat{=} (i,0), \quad j \hat{=} (j,0), \quad k \hat{=} (k,0).$$

They are elements of $\mathrm{Pu}\mathbb{O}$. Putting $l = (0,1) \in \mathrm{Pu}\mathbb{O}$, we note that i, j, l is a Cayley triple and we compute the following products.

$$il = (i,0)(0,1) = (0,i), \ jl = (j,0)(0,l) = (0,j), \ kl = (k,0)(0,1) = (0,k).$$

Following an idea of Freudenthal [4] 1.5.13, p. 19, one may represent this multiplication table graphically using the projective plane with 7 points.

3 The Group of Automorphisms of Octonions

Now we study the group $\mathrm{Aut}(\mathbb{F}_m)$, $m \leq 3$ of automorphisms of the ring $\mathbb{F}_m$, that is we consider automorphisms with respect to addition and multiplication in $\mathbb{F}_m$, linearity over $\mathbb{R}$ is not presupposed. The only automorphism of $\mathbb{F}_0 = \mathbb{R}$ is the identity. The only automorphism of the field $\mathbb{F}_1 = \mathbb{C}$ which fixes the elements of $\mathbb{R}$ are the identity and conjugation. They map i onto i or $-i$.

Proposition 12. *Let $m \geq 2$. Then an automorphism $\alpha \in Aut(\mathbb{F}_m)$ fixes every element of the centre $\mathbb{R}$ of $\mathbb{F}_m$ and so is $\mathbb{R}$-linear. Moreover, α leaves $Pu\mathbb{F}_m$ invariant, commutes with conjugation and is orthogonal with respect to the inner product of $\mathbb{F}_m$.*
Proof. The center $\mathbb{R}$ of $\mathbb{F}_m$ is invariant under α and $\alpha|_{\mathbb{R}} = \mathrm{id}$ because $\mathbb{R}$ has no other automorphism. Hence α is $\mathbb{R}$-linear. By (2) the subspace $\mathrm{Pu}\mathbb{F}_m$ is invariant under α. Therefore conjugation which on $\mathrm{Pu}\mathbb{F}_m$ induces $-\mathrm{id}$, commutes with α. α is orthogonal since $||x^\alpha||^2 = x^\alpha \overline{x^\alpha} = x^\alpha \overline{x}^\alpha = (x\overline{x})^\alpha = (||x||^2)^\alpha = ||x||^2$. □

Let $\mathbb{F}$ be one of the algebras $\mathbb{F}_m$ for $0 \leq m \leq 3$. Using the multiplication in $\mathbb{F}$ we can describe of certain groups of orthogonal transformations acting on the $\mathbb{R}$-vector space $\mathbb{F} = \mathbb{R}^n$, where $n = 2^m$, $m \in \{0,1,2,3\}$. The group of $\mathbb{R}$-linear transformations of $\mathbb{F}$ which are orthogonal with respect to the inner product $\langle \ | \ \rangle$ on $\mathbb{F}$ is denoted by $O_n(\mathbb{R}) = \{C : \mathbb{F} \to \mathbb{F} | C \text{ is } \mathbb{R}-\text{linear}, \ \forall x \in \mathbb{F} : \ ||Cx||^2 = ||x||^2\}$. The normal subgroup $SO_n(\mathbb{R}) = \{C \in O_n(\mathbb{R}) | \det C = 1\}$ of $O_n(\mathbb{R})$ has index 2.

Lemma 13. *(a) For each $a \in \mathbb{F}$ with $||a||^2 = 1$ the transformations $x \mapsto ax$ and $x \mapsto xa$ of $\mathbb{F} = \mathbb{R}^n$ belong to $O_n(\mathbb{R})$ and even to $SO_n(\mathbb{R})$ for $\mathbb{F} \in \{\mathbb{C}, \mathbb{H}, \mathbb{O}\}$.*
(b) The group $SO_n(\mathbb{R})$ is generated by the transformations $\mathbb{F} \to \mathbb{F} : x \mapsto axa$, where $a \in \mathbb{F}$, $||a||^2 = 1$.
Proof. As the norm is multiplicative (cf. (9)) the $\mathbb{R}$-linear transformations in (a) with $||a|| = 1$ are elements of $O_n(\mathbb{R})$.
For $\mathbb{R}$ one has $SO_1(\mathbb{R}) = \{\mathrm{id}\}$. In general the transformations considered in (b) belong to $SO_n(\mathbb{R})$ by (a). In order to show that they generate $SO_n(\mathbb{R})$ we use reflections in hyperplanes. These reflections do not belong to $SO_n(\mathbb{R})$. For any $a \in \mathbb{F}$ with $||a||^2 = 1$, the reflection in the hyperplane orthogonal to a is the mapping

$$\rho_a : x \mapsto x - 2a \cdot \frac{\langle x|a\rangle}{\langle a|a\rangle} = x - 2a \cdot \frac{(\bar{x}a + \bar{a}x)}{2||a||^2} = x - a(\bar{x}a + \bar{a}x) = -a\bar{x}a.$$

Every element of $SO_n(\mathbb{R})$ is the product of an even number of such hyperplane reflections, i.e. the product of transformations having the form $\rho_a\rho_b$ for $a, b \in \mathbb{F}$ with $||a||^2 = 1 = ||b||^2$. Now $x^{\rho_a\rho_b} = -b(\overline{-a\bar{x}a})b = b(\bar{a}x\bar{a})b$ so that $\rho_a\rho_b$ can also be written as the composition of the transformations $x \mapsto \bar{a}x\bar{a}$ and $x \mapsto bxb$. □

Corollary 14. *For $\mathbb{F} = \{\mathbb{C}, \mathbb{H}\}$ the group $SO_n(\mathbb{R})$ consists precisely of the transformations $\mathbb{F} \to \mathbb{F} : x \mapsto b^{-1}xa$, where $a, b \in \mathbb{F}$, $||a||^2 = 1 = ||b||^2$.*

Proof. These transformations belong to $SO_n(\mathbb{R})$ (cf. Lemma 13) and they form a subgroup of $SO_n(\mathbb{R})$. (Here the associativity of $\mathbb{F}$ is required). This subgroup contains the transformations of Lemma 13 (b) (put $b = a^{-1}$) which generate $SO_n(\mathbb{R})$. □

Using $\mathbb{F} = \mathbb{H}$ we obtain a description of $SO_3(\mathbb{R})$ from Corollary 14. The subspace $\mathrm{Pu}\mathbb{H} \cong \mathbb{R}^3$ of the $\mathbb{R}$-vector space $\mathbb{H}$ is the orthogonal complement of $\mathbb{R}$ in $\mathbb{H}$ (cf. definition of *pure* elements). The group of $\mathbb{R}$-linear transformations of $\mathbb{H}$ which are orthogonal with respect to the inner product and which fix 1 may be identified with the group of orthogonal transformations of $\mathrm{Pu}\mathbb{H} \cong \mathbb{R}^3$. We denote both groups by $O_3(\mathbb{R})$. The normal subgroup $SO_3(\mathbb{R})$ of elements of determinant 1 has index 2. Now the transformation $\mathbb{H} \to \mathbb{H} : x \mapsto b^{-1}xa$ in Corollary 14 fixes 1 precisely if $b = a$. Therefore we have:

Corollary 15. *$SO_3(\mathbb{R})$ consists precisely of the transformations $int(a) : \mathbb{H} \to \mathbb{H} : x \mapsto a^{-1}xa$, where $a \in \mathbb{H}$ satisfies $||a||^2 = 1$.*

Hence $SO_3(\mathbb{R})$ is an epimorphic image of the unit sphere $\mathbb{S}^3 = \{a \in \mathbb{H} |\ ||a||^2 = 1\}$ of $\mathbb{H} = \mathbb{R}^4$. By multiplicativity of the norm it is a subgroup of the multiplicative group $\mathbb{H}^*$.

Corollary 16. *The kernel of the epimorphism $int : \mathbb{S}^3 \to SO_3(\mathbb{R}) : a \mapsto int(a)$ is $\{1, -1\}$.*
Proof. An element $a \in \mathbb{S}^3$ belongs to the kernel if and only if $a^{-1}xa = x$ for all $x \in \mathbb{H}$. This is the case if a belongs to the centre of $\mathbb{H}$, which is $\mathbb{R}$. Now one has $\mathbb{R} \cap \mathbb{S}^3 = \{1, -1\}$. □

Now we determine $\mathrm{Aut}(\mathbb{H})$. Because of associativity of $\mathbb{H}$ conjugation by an element $a \in \mathbb{H} \setminus \{0\}$ is an automorphism of the skew field $\mathbb{H}$, the *inner automorphism* $\mathrm{int}(a)$. Let $\mathrm{Int}(\mathbb{H})$ be the group of all

these inner automorphisms. Since $||a||$ belongs to the centre $\mathbb{R}$ of $\mathbb{H}$ we have $\mathrm{int}\left(\frac{1}{||a||}a\right) = \mathrm{int}(a)$. Now $\frac{1}{||a||}a$ has norm 1, so that

$$\mathrm{Int}(\mathbb{H}) = \left\{\mathrm{int}(a) | a \in \mathbb{H}, ||a||^2 = 1\right\}.$$

By Proposition 12 and Corollary 15 we know that $\mathrm{Aut}(\mathbb{H})$ is contained in the group $O_3(\mathbb{R})$ and that $\mathrm{Int}(\mathbb{H}) = SO_3(\mathbb{R})$.

Proposition 17. *$Aut(\mathbb{H}) = Int(\mathbb{H}) = SO_3(\mathbb{R})$.*
Proof. $SO_3(\mathbb{H})$ has index 2 in $O_3(\mathbb{H})$ whence if $\mathrm{Aut}(\mathbb{H})$ were bigger then $SO_3(\mathbb{R})$ we would have $\mathrm{Aut}(\mathbb{H}) = O_3(\mathbb{R})$. In particular, conjugation (which on Pu $\mathbb{H}$ induces $-\mathrm{id} \in O_3(\mathbb{R})$) would have to be an element of $\mathrm{Aut}(\mathbb{H})$, but conjugation is an antiautomorphism and not an automorphism, as $\mathbb{H}$ is not commutative. □

Now we study $\mathrm{Aut}(\mathbb{O})$. We start our discussion with transitivity properties which follows from the fact that $\mathrm{Aut}(\mathbb{O})$ acts sharply transitively on the set of Cayley triples (see Corollary 11).

Lemma 18. *(a) $Aut(\mathbb{O})$ acts transitively on the set*

$$\left\{(u, v) |\ u, v \in Pu\ \mathbb{O},\ u \perp v,\ ||u||^2 = 1 = ||v||^2\right\},$$

i.e. on the 6-sphere $\{u \in Pu\ \mathbb{O}, ||u||^2 = 1\}$ and the stabilizer $Aut(\mathbb{O})_i$ is transitive on the 5-sphere $\{u \in Pu\ \mathbb{O}, ||u||^2 = 1, u \perp i\}$.
(b) The stabilizer $Aut(\mathbb{O})_{i,j}$ is sharply transitive on the 3-sphere $\{(0, b) \in \mathbb{H} \subset \mathbb{H} | |b||^2 = 1\}$.
Proof. To verify (a) we use the fact that for any two pure orthogonal elements u, v with $||u||^2 = 1 = ||v||^2$ there is a Cayley triple having u, v as the first two elements. The elements of the 3-sphere given in (b) are precisely those elements z of Pu$\mathbb{O}$ with unit length which satisfy the property i, j, z is a Cayley triple because i, j, ij span Pu$\mathbb{H} \times \{0\}$ and $\{0\} \times \mathbb{H}$ is the orthogonal space of Pu $\mathbb{H} \times \{0\}$ in Pu$\mathbb{O}$. □

Lemma 19. *(a) The automorphism group $\Lambda = Aut(\mathbb{O})$ acts transitively on the set $\mathcal{H}$ of subalgebras H of $\mathbb{O}$ with $H \cong \mathbb{H}$ so that the stabilizers Λ_H of such subalgebras are conjugate. These stabilizers cover Λ.*
(b) The stabilizer of $\mathbb{H} = \mathbb{H} \times \{0\} \subset \mathbb{H} \times \mathbb{H} = \mathbb{O}$ is

$$\Lambda_{\mathbb{H}} = \{(x, y) \mapsto (a^{-1}xa, b^{-1}ya) |\ a, b \in \mathbb{H}, ||a||^2 = 1 = ||b||^2\},$$

where $(x, y) \in \mathbb{O} = \mathbb{H} \times \mathbb{H}$. It is isomorphic to $SO_4\mathbb{R}$.
(c) The stabilizer of i and j is $\Lambda_{i,j} = \{(x, y) \mapsto (x, b^{-1}y) |\ b \in \mathbb{H}, ||b||^2 = 1\}$.
(d) All involutions of Λ are conjugate and Λ is generated by them.
(e) $Aut(\mathbb{O})$ is a subgroup of $SO_8\mathbb{R}$.

Proof. A subalgebra $H \cong \mathbb{H}$ intersects Pu$\mathbb{O}$ in a 3-dimensional subspace and hence contains orthogonal pure elements u, v of $\mathbb{O}$ such that $||u||^2 = 1 = ||v||^2$. H is a span of $1, u, v$ and uv (see Proposition 5) and $\Lambda = \mathrm{Aut}(\mathbb{O})$ is transitive on the set $\mathcal{H}$ of such subalgebras (see Lemma 19 (a)). In particular the stabilizers Λ_H of $H \in \mathcal{H}$ are conjugate. It remains to show that every element $\lambda \in \Lambda$ leaves some subalgebra $H \in \mathcal{H}$ invariant. Since λ is orthogonal with respect to the inner product of $\mathbb{O}$ and leaves invariant the 7-dimensional subspace Pu$\mathbb{O}$ of $\mathbb{O}$ (cf. Proposition 12), Pu$\mathbb{O}$ decomposes into 1- and 2-dimensional invariant subspaces. Moreover, there is a 2-dimensional subspace spanned by orthogonal pure vectors u, v of norm

1. The subalgebra H spanned by $1, u, v, uv$ is isomorphic to $\mathbb{H}$ (see Proposition 5) and invariant under λ. This proves assertion (a).
The definition of the multiplication in $\mathbb{O} = \mathbb{H} \times \mathbb{H}$ yields that the right hand side of $\Lambda_{\mathbb{H}}$ in (b) is a subgroup M of $\mathrm{Aut}(\mathbb{O})$. The stabilizer $M_{i,j}$ fixes every element on the subalgebra $\mathbb{H} \times \{0\}$ which is spanned by $1, i, j, ij$. Thus, $M_{i,j}$ consists of the transformations described in assertion (b) with $a = \pm 1$ (cf. Corollary 16). This proves assertion (c). Moreover, $M_{i,j}$ acts sharply transitively on the unit sphere of $\{0\} \times \mathbb{H}$. By Lemma 18 the larger group $\Lambda_{i,j} \supseteq M_{i,j}$ is sharply transitively on this unit sphere. Hence one has $\Lambda_{i,j} = M_{i,j}$. M leaves $\mathbb{H} \times \{0\}$ invariant and induces the full automorphism group $\mathrm{Int}(\mathbb{H}) = SO_3(\mathbb{R})$ (see Proposition 17). Moreover, M is transitive on the set of pairs of pure orthogonal elements of norm 1 contained in $\mathbb{H} = \mathbb{H} \times \{0\} \subset \mathbb{H} \times \mathbb{H} = \mathbb{O}$ and this set is invariant under $\Lambda_{\mathbb{H}}$ (cf. Proposition 12). Hence one gets $\Lambda_{\mathbb{H}} = M$. Restriction to $\{0\} \times \mathbb{H}$ gives an epimorphism of $\Lambda_{\mathbb{H}}$ onto $SO_4(\mathbb{R})$ (cf. Corollary 14). If $b^{-1}ya = y$ for all $y \in \mathbb{H}$, then $a = b$. Hence we get an isomorphism and assertion (b) is proved.
Let $\iota \in \Lambda$ be an involution. As ι is orthogonal (see Proposition 12), the $\mathbb{R}$-vector space $\mathbb{O}$ is the orthogonal sum of the eigenspaces F_+ and F_- of ι corresponding to the eigenvalues 1 and -1 and ι is uniquely determined by the fixed space F_+. This is a subalgebra of $\mathbb{O}$. Multiplication by an element $a \in F_- \setminus \{0\}$ is a vector space isomorphism between F_+ and F_-. Hence the subalgebra F_+ has dimension 4 over $\mathbb{R}$ and therefore F_+ is isomorphic to $\mathbb{H}$ (cf. Propositions 4 and 5). As Λ acts transitively on the set $\mathcal{H}$ of such subalgebras (see Lemma 19 (a)) it follows that all involutions are conjugate. The fact that the stabilizer $\Lambda_{\mathbb{H}} \cong SO_4(\mathbb{R})$ is generated by its involutions is proved in [2] (Chap. II, ÃĆÂğ 6, no. 1, p. 51). Then the same holds for the conjugates Λ_H, $H \in \mathcal{H}$ and for their union Λ (see Lemma 19 (a)). Hence the assertion (d) is proved.
According to Proposition 12 $\mathrm{Aut}(\mathbb{O})$ is a subgroup of $O_8(\mathbb{R})$. Now assertions (a) and (b) imply that every automorphism has determinant 1, which proves assertion (e). □

Theorem 20. *$Aut(\mathbb{O})$ is a compact simple Lie group of dimension 14. By the classification of simple Lie groups it is isomorphic to the exceptional compact Lie group G_2.*

Proof. We know that $\mathrm{Aut}(\mathbb{O}) \subseteq O_8(\mathbb{R})$. As multiplication in $\mathbb{O}$ is continuous, $\mathrm{Aut}(\mathbb{O})$ is closed in the usual topology of $O_8(\mathbb{R}) \subseteq GL_8(\mathbb{R})$. $\mathrm{Aut}(\mathbb{O})$ is compact because $O_8(\mathbb{R})$ is compact (cf. [6], Proposition 17.8, p. 337). Hence $\mathrm{Aut}(\mathbb{O})$ is a Lie group since it is a closed linear subgroup. Now we prove that $\mathrm{Aut}(\mathbb{O})$ is a simple group. Assume that $N \neq \{\mathrm{id}\}$ be a normal subgroup of $\Lambda = \mathrm{Aut}(\mathbb{O})$. We have to prove that $N = \mathrm{Aut}(\mathbb{O})$. By Lemma 19 (d) it suffices to show that N contains an involution. From Lemma 19 (a) we get that N intersects some and therefore every of the conjugate subgroup Λ_H, $H \in \mathcal{H}$ non-trivially. Now we prove that every non-trivial normal subgroup of $SO_4(\mathbb{R}) \cong \Lambda_{\mathbb{H}}$ contains an involution. One has $SO_4(\mathbb{R}) = \{\mathbb{H} \to \mathbb{H} : y \mapsto b^{-1}ya |\ a, b \in \mathbb{H}, ||a||^2 = 1 = ||b||^2\}$ (cf. Corollary 14). The subgroups $A := \{\mathbb{H} \to \mathbb{H} : y \mapsto ya |\ a \in \mathbb{H}, ||a||^2 = 1\}$ and $B := \{\mathbb{H} \to \mathbb{H} : y \mapsto b^{-1}y |\ b \in \mathbb{H}, ||b||^2 = 1\}$ are normal in $SO_4(\mathbb{R})$ and $AB = SO_4(\mathbb{R})$. The centralizer $C_{SO_4(\mathbb{R})}(B) = A$ and $C_{SO_4(\mathbb{R})}(A) = B$. If N is normal in $SO_4(\mathbb{R})$ such that $N \cap B = \{\mathrm{id}\}$, then one has $N \subseteq C_{SO_4(\mathbb{R})}(B) = A$. As A and B are isomorphic to $\mathbb{S}^3 = \{a \in \mathbb{H}, ||a||^2 = 1\}$ it suffices to prove that every non-trivial normal subgroup H of $\mathbb{S}^3$ contains an involution. We consider the epimorphism $\mathrm{int} : \mathbb{S}^3 \to SO_3(\mathbb{R})$ with kernel $\{1, -1\}$ (cf. Corollary 16). If H contains the involution -1 the proof is finished. If not, then H is mapped isomorphically onto a non-trivial normal subgroup of $SO_3(\mathbb{R})$ by int. As $SO_3(\mathbb{R})$ is simple (see [1], p. 57) we have $\mathrm{int}(H) = SO_3(\mathbb{R})$. Hence H must contain involutions because $SO_3(\mathbb{R})$ does. (This latter case does not occur, but this does not affect the argument).
The dimension of $\mathrm{Aut}(\mathbb{O})$ can be computed by applying the dimension formula for stabilizers to the transitive actions given in Lemma 18. We have $\dim(\mathrm{Aut}(\mathbb{O})_{i,j}) = \dim(\mathbb{S}^3) = 3$ (cf. Lemma 19 (c) or Lemma 18 (b). $\mathrm{Aut}(\mathbb{O})_{i,j}$ is a stabilizer of the transitive action of $\mathrm{Aut}(\mathbb{O})_i$ on the 5-sphere of pure unit quaternions orthogonal to i. Hence we have $\dim(\mathrm{Aut}(\mathbb{O})_i) = \dim(\mathrm{Aut}(\mathbb{O})_{i,j}) + 5 = 8$. As $\mathrm{Aut}(\mathbb{O})$ acts transitively on the 6-dimensional unit sphere of Pu $\mathbb{O}$ (cf. Lemma 18 (a)) we obtain that $\dim(\mathrm{Aut}(\mathbb{O})) = \dim(\mathrm{Aut}(\mathbb{O})_i) + 6 = 14$. There is precisely one almost simple Lie group of dimension 14, the exceptional Lie group G_2 (cf. the classification of the almost simple Lie groups in [7], 94.32 and 94.33). □

References

1. Artin, E. Geometric algebra, New York: Interscience.
2. Dieudonné, J. La géométrie des groupes classiques, Berlin etc.: Springer (3rd edition).
3. English McLewin, K. Octonions and the Exceptional Lie Algebra $\mathbf{g}_2$, Master Thesis of Science in Mathematics, Virginia Polytechnic Institute, Blacksburg, Virginia, 2004.
4. Freudenthal, H. Oktaven, Ausnahmegruppen und Oktavengeometrie, Utrecht 1951, reprinted in Geom. Dedicata **19**, 7-63.
5. Jacobson N. Cayley Numbers and Normal Simple Lie Algebras of type G, Duke Math. Journal, **775**, Vol. 5, 1939, 775-783.
6. Porteous, I. Topological geometry, Cambridge Univ. Press (2nd edition).
7. Salzmann, H. etc. Compact Projective Plane, Berlin-New York: Walter de Gruyter **21**, 1995.

Groups Associated to Simple Lie Algebras

Zoltán Halasi

Department of Algebra and Number Theory
Institute of Mathematics, University of Debrecen
H-4032 Debrecen, Egyetem tér 1, Hungary
zhalasi@science.unideb.hu

Abstract. Simple Lie algebras play a crucial role both in the theory of Lie groups and in finite group theory. On the one hand, simple Lie groups can generally be found as automorphism groups of simple Lie algebras. On the other hand, by the classification theorem of finite simple groups (CFSG), most of the finite simple groups are related to simple Lie algebras over finite fields. The goal of this note is to highlight the relation between Lie algebras and groups.

1 Connection between Lie Groups and their Lie Algebras

In the following we define the concept of Lie groups and show the connection between a Lie group and its associated Lie algebra. We remark that when we speak about a Lie group we always mean it is a *real* Lie group; Note that it is also possible to speak about complex Lie groups. In order to define Lie groups, we first need to understand what differentiable manifolds in general are.

1.1 Differentable Manifolds

We save the reader here from the precise abstract definition of what (smooth or differentable) manifolds in general are. Roughly speaking, an n dimensional (smooth) manifold is a set, such that in each point of the set locally it looks like the n-dimensional space. This means that one can take a picture of the space by giving an atlas on it. The atlas consists of charts, each chart represents a subset of the manifold, such that the charts cover the whole manifold and if the subsets represented by the charts have non-empty intersection, than if we move from one of the charts to the manifold and then toward to the other chart, then this composition is a smooth (i.e. infinitely many times differentable) function between two open subsets of $\mathbb{R}^n$. To imagine this, just think of the Earth and an atlas covering every segment of the Earth. Of course, in order to make calculations, we need to define the above charts (maps) precisely. A possibility to achieve this is to define embedded manifolds.

Definition 1 (embedded manifold)**.** A k-dimensional embedded manifold of $\mathbb{R}^n$ is a subset $M \subset \mathbb{R}^n$ given as the common roots of smooth functions $f_1, \dots, f_{n-k} \in \mathbb{R}$ such that $\operatorname{grad} f_1(p), \dots, \operatorname{grad} f_{n-k}(p)$ are linearly independent for each $p \in M$.

One can show by using the Inverse function theorem that in this way one really gets a k-dimensional manifold. Furthermore, we remark that although at first sight the definition of embedded manifolds is more special than the concept of general manifolds, by Whitney's theorem every abstract manifold can be embedded into some higher dimensional space.

Example. As an example, let us consider the sphere $S^2 \subset \mathbb{R}^3$ with radius one centered at the origin. This subset of $\mathbb{R}^3$ can be given as the solutions of the equation $f(x,y,z) = x^2 + y^2 + z^2 - 1 = 0$. Now, $f(x,y,z) : \mathbb{R}^2 \to \mathbb{R}$ is a smooth function and its gradient $\operatorname{grad} f(x,y,z) = (2x, 2y, 2z)$, which is zero only at the origin, so it is nowhere zero on S^2. As a result, S^2 is a two dimensional embedded manifold in $\mathbb{R}^3$.

In the following all of our examples will be embedded manifolds. For such manifolds $M \subseteq \mathbb{R}^m$ and $N \subseteq \mathbb{R}^n$ it is meaningful to speak about that a function $f : M \to N$ is differentable (or smooth) as well as about its derivative $f' : M \to \mathbb{R}^{n \times m}$.

Now, we can turn to the definition of Lie groups.

Definition 2 (Lie group)**.** An n-dimensional Lie group G is a smooth manifold together with a multiplication map $m : G \times G \to G$ such that m is a smooth function and G is a group with the multiplication given by m. The product of two elements $g, h \in G$ will be simply denoted by gh instead of $m(g,h)$.

It is easy to see, that having a Lie group G the maps $(g,h) \to gh^{-1}$ and $g \to g^{-1}$ are smooth functions. In the following, for any fixed $g \in G$, we will denote by L_g and R_g the left and right translations by g, i.e. $L_g(h) = gh$ and $R_g(h) = hg$ for every $h \in G$. Both left and right translations are diffeomorphisms $G \to G$.

In the following our goal is to define the Lie algebra of a Lie group. To achieve this, first we need to speak about the tangent vectors and the tangent space of a manifold X at a point $p \in X$. As we only bother with embedded manifolds, we can use geometric approach to define them.

Definition 3 (tangent vector, tangent space)**.** If $M \subseteq \mathbb{R}^n$ is an embedded k-dimensional manifold (given by equations $f_1 = f_2 = \ldots = f_{n-k} = 0$) and $p \in M$ then we say that $v \in \mathbb{R}^n$ is a *tangent vector* of M at p if there is a smooth curve $\gamma : [-\varepsilon, \varepsilon] \to M, (\varepsilon > 0)$ such that $\gamma(0) = p$ and $\gamma'(0) = v$. The set of all tangent vectors of M at p is called the tangent space of M at p and it is denoted by $T_p(M)$.

Proposition 4. *If the manifold $M \subseteq \mathbb{R}^n$ is given with the system of equations $f_1 = f_2 = \ldots = f_{n-k} = 0$ (with $\operatorname{grad} f_1(p), \ldots \operatorname{grad} f_{n-k}(p)$ linearly independent for each $p \in M$!), then the tangent space of M at p is simply the orthogonal component of the subspace $\langle \operatorname{grad} f_1(p), \ldots \operatorname{grad} f_{n-k}(p) \rangle$, so it is a subspace of dimension k.*

Although the above definition of tangent space is expressive, it cannot be applied for abstract manifolds and sometimes it is hard to use them by making calculations. Fortunately, it is possible to define it in an abstract different way which is more algebraic.

Definition 5 (derivation, tangent space)**.** For a manifold M, let $C^\infty(M)$ be the set of all smooth functions $M \to \mathbb{R}$. This is a real associative algebra with pointwise product and sum of functions and scalar multiplication. We say that a linear function $D : C^\infty(M) \to \mathbb{R}$ is a derivation at the point $p \in M$ if $D(fg) = D(f)g(p) + f(p)D(g)$ holds for every $f, g \in C^\infty(M)$ (This equality is called the Leibniz rule). Te set of all derivations at p is the tangent space $T_p(M)$ of M at p.

Although the two definitions of the tangent space $T_p(M)$ seems to be completely different, they are in fact the same (or, at least, there is a one-to one correspondence between them).

Proposition 6. *Given a curve $\gamma : [-\varepsilon, \varepsilon] \to M$ with $\gamma(0) = p$ and tangent vector $\gamma'(0)$ there is a derivation $D_{\gamma'(0)} : C^\infty(M) \to \mathbb{R}$ at p given as $D_{\gamma'(0)}(f) = (f \circ \gamma)'(0)$. For two curves, this definition gives the same if and only if their tangent vectors at p are the same. Furthermore, every derivation at p can be written in the form $D_{\gamma'(0)}$ for some curve γ.*

Having defined the tangent space of the manifold M at each point $p \in M$ we can think of all the tangent spaces at the same time to get the concept of tangent bundle $T(M)$.

Definition 7 (tangent bundle)**.** The tangent bundle of M is the set $T(M) = \cup_{p \in M} T_p(M) = \{(p, v) \mid p \in M\ v \in T_p(M)\}$. It also has a manifold structure of dimension twice as much as the dimension of M.

Example. As an example, let us choose the two-dimensional manifold $S^2 \subseteq \mathbb{R}^3$ defined in Example 1.1. If $p = (p_1, p_2, p_3) \in S^2$ (i.e. $f(p) = p_1^2 + p_2^2 + p_3^2 - 1 = 0$), then, taking a curve γ on S^2 through p, its (derivative) speed will be a vector tangential on the sphere S^2. We know from elementary geometry that the vectors tangential to the sphere S^2 at a point $p \in S^2$ are exactly the vectors perpendicular to the radius vector r drawn from the centre of the sphere (i.e. the origin) to p. Since $r = (p_1, p_2, p_3)$, we get $x = (x_1, x_2, x_3) \in T_p(S^2) \iff x_1p_1 + x_2p_2 + x_3p_3 = 0$. On the other hand, $\operatorname{grad} f(p) = (f'_x(p), f'_y(p), f'_z(p)) = (2p_1, 2p_2, 2p_3) = 2r$. So $x \in T_p(M) \iff x \perp r \iff x \perp \operatorname{grad} f(p)$.
Finally, the tangent bundle of S^2 is just the union of those planes in $\mathbb{R}^3$ which are tangential to the sphere S^2 at some point $p \in S^2$, that is, the set of planes having distance one from the origin.

Having defined tangent spaces and the tangent bundle, next we define the concept of vector fields.

Definition 8 (vector field)**.** A (smooth) vector field over a manifold M is a smooth function $X : M \to TM$ such that $X(p) \in T_p(M)$ for each $p \in M$.

Since tangent vectors at each point $p \in M$ are identified with derivations at p, a vector field X can be identified with differential operators $X : C^\infty(M) \to C^\infty(M)$ by the formula $(X(f))(p) = (X(p))(f)$. With this definition, a vector field is a linear map $C^\infty(M) \to C^\infty(M)$ satisfying the Leibniz rule $X(fg) = X(f)g + fX(g)$.

(Comparing with usual multivariable calculus, derivations at a point p corresponds to tangential derivatives, and vector fields corresponds to differential operators whith smoothly varying direction.)

As vector fields are $C^\infty(M) \to C^\infty(M)$ functions, they can be composed, so we can define the Lie bracket of vector fields as $[X, Y] = X \circ Y - Y \circ X$. It turns out that the Lie bracket of two vector fields is a vector field itself. The set of all vector fields on a manifold M with pointwise addition, scalar multiplication and Lie bracket form a Lie algebra. (This is an infinite dimensional Lie algebra.)

1.2 Lie Groups and their Lie Algebras

Now, turn back to Lie groups. As a Lie group G is also a manifold, we can see the Lie algebra of all vector fields over G. The problem with this huge manifold that it does not reflect the structure of the Lie group. Therefore, we consider only left invariant vector fields. It turns out that the set of left invariant vector spaces form a Lie subalgebra in the Lie algebra of all vector fields over the group. This Lie algebra is called the Lie algebra of the group G and is denoted by $\mathfrak{g}$. Every left invariant vector field is uniquely defined by its value at e (the identity element of G). In that way $\mathfrak{g}$ can be identified with the tangent space $T_e(G)$. Thus, $\mathfrak{g}$ is a finite dimensional Lie algebra with dimension equal to the dimension of G.

Some of the most important Lie groups can be defined as subgroups of the algebra $M^{n\times n}(\mathbb{R})$ of $n\times n$ real matrices. In the following, we show some examples.

Example. 1. The general linear group $GL_n(\mathbb{R})$ is the group of all invertable $n\times n$ real matrices. It is a dense open subset of the space of all $n\times n$ real matrices, so it has dimension n^2. Its Lie algebra is the Lie algebra of all $n\times n$ real matrices $\mathfrak{gl}_n(\mathbb{R})$ with Lie bracket $[A,B]=AB-BA$. Note that $GL_n(\mathbb{R})$ is not an embedded manifold in the n^2 dimensional space, but it can be embedded into the n^2+1 dimensional real vector space.

2. The special linear group $SL_n(\mathbb{R})=\{A\in GL_n(\mathbb{R})\,|\,\det A=1\}$ is an embedded n^2-1 dimensional Lie group in the space of $n\times n$ matrices given as the set of roots of the determinant minus one function. (Note that the determinant can be given as a polynomial function with n^2 variables.) Its Lie algebra $\mathfrak{sl}_n(\mathbb{R})$ is the space of all $n\times n$ matrices of trace 0.

3. We can define a series of Lie groups and their Lie algebras in the following way. Let K be either $\mathbb{R}$ or $\mathbb{C}$ and $(\ )':M^{n\times n}(K)\to M^{n\times n}(K)$ be a linear map satisfying $I'=I$ and $(AB)'=B'A'$ for each $A,B\in M^{n\times n}(K)$. (Here $(\ )'$ will be either the transpose $(\ )^T$ or the conjugate transpose $(\ )^*$ operation.) Given an arbitrary matrix $J\in M^{n\times n}(K)$ the set

$$G_J=\{A\in GL_n(K)\mid AJA'=J\}$$

forms a Lie subgroup of $GL_n(K)$. Its associated Lie algebra is

$$\mathfrak{g}_J=\{A\in M^{n\times n}(K)\mid AJ+JA'=0\}.$$

Choosing special J-s we get the so-called classical Lie groups.

(a) Let $K=\mathbb{R}$ and $(\ )'=(\ )^T$. By taking $J=I$ (the identity matrix) we get $G_J=O_n(\mathbb{R})=\{A\in GL_n(\mathbb{R})\mid AA^T=I\}$ the group of orthogonal matrices. Its Lie algebra is $\mathfrak{o}_n(\mathbb{R})=\{A\in M^{n\times n}(\mathbb{R})\mid A^T=-A\}$ the Lie algebra of all antisymmetric $n\times n$ matrices. (As a result, $\dim O_n(\mathbb{R})=\binom{n}{2}$.)

(b) A similar example can be found as in *(a)* if one chooses J to be a diagonal matrix with k ones and $n-k$ minus ones in the main diagonal. In that way one gets the pseudoorthogonal group $G_J=O_{k,n-k}(\mathbb{R})$. Note that $\dim O_{k,n-k}(\mathbb{R})=\binom{n}{2}$. As a special case we get the Lorentz group $O_{n-1,1}(\mathbb{R})$.

(c) For $K=\mathbb{R}$, $(\ )'=(\ )^T$, $n=2k$ even and $J=\begin{pmatrix}0 & I\\ -I & 0\end{pmatrix}$ we get the symplectic group $G_J=Sp_n(\mathbb{R})$ of dimension $2k^2+k$. Its Lie algebra is

$$\mathfrak{sp}_n(\mathbb{R})=\left\{\begin{pmatrix}A & B\\ C & -A^T\end{pmatrix}\mid A,B,C,D\in M^{k\times k}(\mathbb{R}),\ B=B^T,\ C=C^T\right\}.$$

(d) For choosing $K=\mathbb{C}$, $(\ )'=(\ )^*$ and $J=I$ we get the uniary group $G_J=U(n)=\{A\in GL_n(\mathbb{C})\mid AA^*=I\}$. Its Lie algebra is $\mathfrak{u}(n)=\{A\in M^{n\times n}(\mathbb{C})\mid A^*=-A\}$. As a *real* Lie group, $\dim U(n)=n^2$.

Now, we define the exponential map from a Lie algebra $\mathfrak{g}$ of a Lie group G into G. Recall that an element X of the Lie algebra $\mathfrak{g}$ is just a left-invariant vector field $X : G \to TM$, which is uniquely determined by its value $X(e) \in T_e(G)$. Now, one can show that there is a unique smooth diffeomorphism $\gamma_X : \mathbb{R} \to G$ such that $\gamma_X(0) = X(e)$. (For this γ_X one can show that $\gamma'_X(t) = X(\gamma_X(t))$ for each $t \in \mathbb{R}$ i.e. the speed of the curve defined by γ_X equals to the value of the vector field X at each point of G.) This γ_X is called the one-parameter subgroup of G related to X. Now, $\exp : \mathfrak{g} \to G$ defined as $\exp(X) = \gamma_X(1)$ for each $X \in \mathfrak{g}$ is a well-defined map with remarkable properties. With its help one can prove a number of results about the connection between Lie algebras and Lie groups.

Theorem 9. *Let G be a Lie group with Lie algebra $\mathfrak{g}$.*

1. *There is a one-to-one correspondence between connected (immersed) Lie subgroups of G and the subalgebras of the Lie algebra $\mathfrak{g}$.*
2. *For every finite dimensional Lie algebra $\mathfrak{g}$ there is a connected and simply connected Lie group whose Lie algebra is $\mathfrak{g}$.*
3. *If G and H are connected and simply connected Lie groups with Lie algebras $\mathfrak{g}$ and $\mathfrak{h}$ respectively, then G and H are isomorphic Lie groups if and only if $\mathfrak{g}$ and $\mathfrak{h}$ are isomorphic Lie algebras.*

It is not necessarily true that if the Lie algebras of two Lie groups are isomorphic, then the Lie groups are isomorphic. Instead, starting from an arbitrary Lie group G, first we can take its identity component G_0. Thereafter, it can be shown that there exists a connected, simply connected universal covering group of G_0 denoted by $\hat{G}$ and it is unique up to isomorphism. It is also true that the covering map $\hat{G} \to G_0$ is a group homomorphism whose kernel is a discrete subgroup of $\hat{G}$ contained in the kernel of $\hat{G}$. Now, G, G_0 and $\hat{G}$ have isomorphic Lie algebra. Moreover, two Lie group have isomorphic Lie algebras if and only if the universal covering groups of their identity components are isomorphic Lie groups.

Example. As an example, the orthogonal group $O_3(\mathbb{R})$ is not connected, it has two connected components: The sets of orthogonal matrices with determinant 1 and with determinant -1, respectively. Thus, the special orthogonal group $SO_3(\mathbb{R})$ is a connected Lie group. However, $SO_3(\mathbb{R})$ is not simply connected. Its universal covering group is SU_2, and the kernel of the covering map $SU_2 \to SO_3(\mathbb{R})$ is the subgroup $\{\pm I\} = Z(SU_2)$. Thus, $O_3(\mathbb{R}), SO_3(\mathbb{R})$ and $SU_2(\mathbb{R})$ have isomorphic Lie algebras; As we mentioned in Example 1.2, this Lie algebra is the Lie algebra of all 3×3 antisymmetric matrices.

We close this section by showing, how the image of the exponential map can be calculated, at least, for linear Lie algebras. We say that a Lie algebra $\mathfrak{g}$ is linear if $\mathfrak{g} \leq \mathfrak{gl}_n(\mathbb{R})$ for some n. For such a Lie algebra, if $A \in \mathfrak{g}$ is an arbitrary matrix, then one can calculate $\exp(A)$ by substituting A into the usual power series of the exponential function e^x, so we get

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!} \in GL_n(\mathbb{R}).$$

Note that the above infinite sum convergent for every matrix A. Furthermore, the image $\{\exp(A) \mid A \in \mathfrak{g}\}$ is not a Lie group in general. Instead, $\langle \exp(A) \mid A \in \mathfrak{g} \rangle$ will be a connected Lie group whith Lie algebra $\mathfrak{g}$. Using this, one can check that the Lie algebras of the classical Lie groups are those given in Example 1.2.

We close this section with the remark that in fact every finite dimensional real Lie algebra is isomorphic to some linear Lie algebra by the deep theorem of Ado.

2 Classification of Simple Real Lie algebras and Lie Groups

In this section we would like to give a very rough description on the classification of simple real Lie algebras. As we have seen in the previous section, in order to classify Lie groups, we should first classify real Lie algebras. By the Levi-Mal'cev theorem, every finite dimensional Lie algebra $\mathfrak{g}$ in characteristic zero is of the form $\mathfrak{g} = \mathfrak{r} \rtimes \mathfrak{s}$ where $\mathfrak{r}$ is the radikal of $\mathfrak{g}$ (the largest solvable ideal of $\mathfrak{g}$) and $\mathfrak{s}$ is a semisimple subalgebra of $\mathfrak{g}$. Moreover, in this decomposition $\mathfrak{s}$ (the Levi subalgebra) is unique up to automorphisms of $\mathfrak{g}$.

Now, while the classification of solvable Lie algebras seems to be hopeless, a semisimple Lie algebra is a finite direct sum of simple Lie algebras. Now, we would like to emphasize the method how the classification of complex simple Lie algebras can be used to describe the types of real simple Lie algebras. In this and in the following section we assume that the reader is already familiar with the relevant concepts of the classification theorem of simple Lie algebras over the complex fields. (Cartan decomposition, root system, Dynkin diagrams, etc), as well as how the base field of an algebra can be extended with the concept of tensor product.

Definition 10 (complexification, realification)**.** Starting from a real Lie algebra $\mathfrak{g}$, one can define its *complexification* $\mathfrak{g}(\mathbb{C})$ as $\mathfrak{g}(\mathbb{C}) = \mathfrak{g} \otimes_{\mathbb{R}} \mathbb{C}$. On the other hand, we can define the *realification* $\mathfrak{g}_{\mathbb{R}}$ of any complex Lie algebra $\mathfrak{g}$ by simply "forgetting" the multiplication with non-real scalars.

Note that the above two operations are not inverse to each other. Instead, taking the two operations one after the other to a (complex or real) Lie algebra in the appropriate way, we get a Lie algebra over the original field, with dimension twice as much as the dimension of the original Lie algebra. Moreover, if $\mathfrak{g} = \mathfrak{r} \oplus \mathfrak{s}$ is a Levi decomposition of the real Lie algebra $\mathfrak{g}$ then $\mathfrak{g}(\mathbb{C}) = \mathfrak{r}(\mathbb{C}) \oplus \mathfrak{s}(\mathbb{C})$ is a Levi decomposition of its complexification. In particular, the complexification of a semisimple real Lie algebra will be a semisimple complex Lie algebra. Starting from a simple complex Lie algebra $\mathfrak{g}$ its realification $\mathfrak{g}_{\mathbb{R}}$ is always a simple real Lie algebra. (Hence in this way we already get a family of simple real Lie algebras $(A_l)_{\mathbb{R}}, \dots, (G_2)_{\mathbb{R}}$ of (real) dimension twice as much as the (complex) dimension of $\mathfrak{g}$.). All the other simple real Lie algebras can be found by finding the so-called real forms of the complex simple Lie algebras.

Definition 11 (real form)**.** A real form of a complex Lie algebra $\mathfrak{g}$ is a real subalgebra $\mathfrak{h}$ of $\mathfrak{g}_{\mathbb{R}}$ such that $\mathfrak{h} \otimes \mathbb{C} \simeq \mathfrak{g}$.

It is clear that any real form of a simple complex Lie algebra is a simple real Lie algebra. We also note that although an arbitrary complex Lie algebra does not necessarily admit a real form, a simple complex Lie algebra allways have at least two different real forms, called them the split and the compact real form.

Definition 12 (signature of a simple real Lie algebra)**.** If $\mathfrak{h}$ is a real simple Lie algeba, then its Killing form is a non-degenerate symmetric bilinear function, hence by using Gram-Schmidt orthonalization, one can find a basis in $\mathfrak{h}$ such that the Killing form is diagonal with respect to this basis with only 1-s and -1-s in the main diagonal. If the number of 1-s and -1-s are p and q respectively, then the Lie algebra has signature (p, q).

Definition 13 (compact and split real forms)**.** The *compact real form* of a simple complex Lie algebra is a real form whose Killing form is negative-definite (i.e. it has signature $(0, n)$.) The *split real form* of a simple complex Lie algebra is the opposite: this is the real form being the most far from being compact; that is, if its signature is (p, q) then p is as large as possible. (Note that, in general, the Killing form of a split real simple Lie algebra is not positive definite.)

We note that the split form of a complex simple Lie algebra can be found by choosing a Chevalley basis, and taking the real subspace generated by it.

Example. By choosing the simple Lie algebra $A_1 = \mathfrak{sl}_2(\mathbb{C})$ over $\mathbb{C}$ it has exactly two real forms. Its split real form is $\mathfrak{sl}_2(\mathbb{R})$, while its compact real form is $\mathfrak{su}_2(\mathbb{C})$. For n larger than two, the split and compact real forms of $A_{n-1} = \mathfrak{sl}_n(\mathbb{C})$ are still $\mathfrak{sl}_n(\mathbb{R})$ and $\mathfrak{su}_n(\mathbb{C})$, but there are other real forms, as well.

The various types of real forms of a simple complex Lie algebra $\mathfrak{g}$ can be found by characterizing the equivalence classes of the involutory antilinear automorphisms (conjugations) of $\mathfrak{g}$ under the action of $\operatorname{Aut}\mathfrak{g}$.

Definition 14 (conjugation with respect to a real form)**.** Let $\mathfrak{g}$ be a complex Lie algebra with real form $\mathfrak{h}$. Then every element of $\mathfrak{g}$ can be written uniquely of the form $x+i\cdot y$ where $x, y \in \mathfrak{h}$ and $i \in \mathbb{C}$ is the imaginary unit. Now, we can define the conjugation σ on $\mathfrak{g}$ with respect to $\mathfrak{h}$ with the rule $\sigma(x + i \cdot y) = x - i \cdot y$.

Proposition 15. *Using the previous notation, the following statements can be easily verified*

1. *σ is an involutory (i.e. $\sigma \circ \sigma = \mathrm{id}_{\mathfrak{g}}$) antilinear (i.e. $\sigma(\lambda z) = \overline{\lambda}\sigma(z)$ for each $\lambda \in \mathbb{C},\ z \in \mathfrak{g}$) Lie algebra automorphism of $\mathfrak{g}$.*
2. *$\mathfrak{h} = \mathfrak{g}^\sigma = \{x \in \mathfrak{g} \mid \sigma(x) = x\}$.*
3. *If $\varphi : \mathfrak{g} \to \mathfrak{g}$ is any involutory antilinear Lie algebra automorphism of $\mathfrak{g}$ then $\mathfrak{g}^\varphi = \{x \in \mathfrak{g} \mid \varphi(x) = x\}$ is a real form of $\mathfrak{g}$.*
4. *For two conjugations $\sigma, \varphi : \mathfrak{g} \to \mathfrak{g}$ we have $\mathfrak{g}^\sigma \simeq \mathfrak{g}^\varphi$ if and only if there exists a $\tau \in \operatorname{Aut}\mathfrak{g}$ with $\tau \circ \sigma \circ \tau^{-1} = \varphi$.*

By using the above statements, it is clear that the non-isomorphic real forms of a simple complex Lie algebra are in one-two-one correspondence with the equivalence classes of the involutory antilinear automorphisms (conjugations) of $\mathfrak{g}$ under the action of $\operatorname{Aut}\mathfrak{g}$, so one needs to classify the equivalence classes of conjugations of simple complex Lie algebras which can be managed.

As a final remark, we note that the various real forms of simple complex Lie algebras can be characterized by so-called "Satake diagrams". The Satake diagram of a real form is obtained from the Dynkin diagram of the associated complex simple Lie algebra with some vertices colored to black and some other vertices in pairs connected by arrows.

3 Construction of Finite Simple Groups of Lie Type

Now, we turn to the description of those finite simple groups which are related to Lie groups, called finite simple groups of Lie type.

Note that according to the Classification Theory Of Finite Simple Groups, such simple groups provide us 16 infinite families. Apart from cyclic groups of prime orders, the alternating groups A_n for $n \geq 5$ and 26 so-called sporadic groups, every finite simple group is of Lie type.

3.1 Chevalley Algebras over Finite Fields

Let $\mathfrak{s}$ be a simple Lie algebra over $\mathbb{C}$. Starting from a Cartan decomposition $\mathfrak{s} = \mathfrak{h} \oplus (\oplus_{s\in\Phi}\mathfrak{l}_s)$ of $\mathfrak{s}$ one can show that there is a basis $\mathcal{B}_{Ch} = \{h_r, r \in \Pi; e_s, s \in \Phi\}$ of $\mathfrak{s}$ such that the Lie bracket of every two element of $\mathcal{B}_{Ch}$ is an *integer* multiple of some other element of $\mathcal{B}_{Ch}$. (Such a basis of $\mathfrak{s}$ is called a Chevalley basis of $\mathfrak{s}$.) As a result, taking only the set of integer linear combinations of $\mathcal{B}_{Ch}$ in $\mathfrak{s}$, this set is closed under Lie bracket, so one gets a Lie algebra $\mathfrak{s}(\mathbb{Z})$ over $\mathbb{Z}$. Now, one can get a Lie algebra $\mathfrak{s}(K)$ of type $\mathfrak{s}$ over an arbitrary field K by simply taking the tensor product $\mathfrak{s}(\mathbb{Z}) \otimes_{\mathbb{Z}} K$. In particular, for each prime power $q = p^f \in \mathbb{N}$, one gets the finite Lie algebra $\mathfrak{s}(q) = \mathfrak{s}(\mathbb{Z}) \otimes_{\mathbb{Z}} \mathbb{F}_q$ of type $\mathfrak{s}$ over the finite field $\mathbb{F}_q$. Note that, $\mathfrak{s}(q)$ will not be a simple Lie algebra in general! A further remark is, that one can choose a Chevalley basis in many ways in the simple Lie algebra $\mathfrak{s}$, but the resulting $\mathfrak{s}(\mathbb{Z})$ will always be the same up to isomorphism. This results that the isomorphism type of $\mathfrak{s}(q)$ is also uniquely defined.

3.2 The Chevalley Groups

Now, we define the Chevalley groups $S(K)$ for a Chevalley algebra $\mathfrak{s}(K)$ of type $\mathfrak{s}$ over K. In fact, this group will be just a subgroup of the automorphism group of $\mathfrak{s}(K)$ generated by some special automorphisms related to the Chevalley basis elements $\{e_r; r \in \Phi\}$. First, we show how to construct this for $\mathfrak{s} = \mathfrak{s}(\mathbb{C})$. (In fact, the same argument also works over any field of characteristic 0.) In order to be able to do this we need some general terminology.

A derivation of a Lie algebra $\mathfrak{l}$ is a linear map $\delta\colon \mathfrak{l} \to \mathfrak{l}$ satisfying $\delta([f,g]) = [\delta(f), g] + [f, \delta(g)]$. For example, $\operatorname{ad} x\colon \mathfrak{l} \to \mathfrak{l}$ is a derivation for each $x \in \mathfrak{l}$. (Such derivations are called inner derivations of $\mathfrak{l}$.) As derivations are functions from $\mathfrak{l}$ into itself, they can be composed. In particular, δ^k is a well-defined linear map $\mathfrak{l} \to \mathfrak{l}$ for each $k \in \mathbb{N}$ and derivation δ of $\mathfrak{l}$. We say that the derivation δ is nilpotent if $\delta^n = 0$ for some $n \in \mathbb{N}$. Such a derivation can be substituted into the power series of the usual exponential function $\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$. So we get

$$\exp(\delta) : 1 + \delta + \frac{\delta^2}{2} + \cdots + \frac{\delta^k}{k!} + \cdots$$

Note that the above sum is finite, since $\delta^n = 0$ for some n and it is meaningful, since the characteristic of the base field ($\mathbb{C}$) is zero.

One can prove that $\exp(\delta)\colon \mathfrak{l} \to \mathfrak{l}$ will be in fact a Lie automorphism of $\mathfrak{l}$ for any nilpotent derivation δ of $\mathfrak{l}$.

Now, starting from the simple Lie algebra $\mathfrak{s} = \mathfrak{s}(\mathbb{C})$ and a Chevalley basis $\mathcal{B}_{Ch} = \{h_r, r \in \Pi;\ e_s, s \in \Phi\}$ of $\mathfrak{s}$, it is clear that $\zeta \cdot \operatorname{ad} e_s$ is a nilpotent derivation of $\mathfrak{s}$ for every $s \in \Phi$ and $\zeta \in \mathbb{C}$. Therefore, $x_s(\zeta) := \exp(\zeta \operatorname{ad} e_s)$ is an automorphism of $\mathfrak{s}$. Now, the Chevalley group S of $\mathfrak{s}$ is defined as

$$S = \langle x_s(\zeta) \mid s \in \Phi,\ \zeta \in \mathbb{C} \rangle \leq \operatorname{Aut} \mathfrak{s}.$$

As a remark, compare this with the end of Section 1.

Now, we turn to the case when K is a field of characteristic p. At first sight the above construction cannot be used in general, since we cannot substitute $\operatorname{ad} e_s$ into the power series of the exponential function if p is smaller than the nilpotency class of $\operatorname{ad} e_s$. (Of course, this can only occur for $p \in \{2, 3\}$.) However, calculating the action of $x_r(\zeta) \in S$ on the elements of the Chevalley basis of $\mathfrak{s} = \mathfrak{s}(\mathbb{C})$, we get the following hold for every $\zeta \in \mathbb{C}$ and $r, s \in \Phi$ with r, s linearly independent roots:

$$x_r(\zeta)(e_r) = e_r;$$
$$x_r(\zeta)(e_{-r}) = e_{-r} + \zeta h_r - \zeta^2 e_r;$$

$$x_r(\zeta)h_r = h_r - 2\zeta e_r;$$
$$x_r(\zeta)h_s = h_s - A_{sr}\zeta e_r;$$
$$x_r(\zeta)e_s = \sum_{i=0}^{q} \pm \binom{p+i}{i} \zeta^i e_{ir+s}.$$

That is, the automorphism $x_r(\zeta)$ transforms each element of the Chevalley basis into such a linear combination of the basis elements that the coefficients are all non-negative integral powers of ζ multiplied by rational integers. Now, fixing an arbitrary field K and choosing a $t \in K$ we can define the map $x_r(t)\colon \mathfrak{s}(K) \to \mathfrak{s}(K)$ by simply substituting t in place of ζ in the above equalities. One can show that in this way $x_r(t)$ really defines an automorphism of $L(K)$. Now, the Chevalley group (of adjoint type) of $\mathfrak{s}(K)$ is simply defined as

$$S(K) = \langle x_r(t) \,|\, r \in \Phi,\, t \in K \rangle$$

like it was defined over $\mathbb{C}$.

It turns out that in case of K is a finite field the Chevalley groups $L(K)$ are generally finite simple groups (apart from a few exceptions). In this way one gets 9 classes of finite simple groups of Lie type: $A_l(q)$, $B_l(q)$, $C_l(q)$, $D_l(q)$, $E_6(q)$, $E_7(q)$, $E_8(q)$, $F_4(q)$, $G_2(q)$. Note that these are not all the finite simple group of Lie type; other groups can be obtained as subgroups of some Chevalley groups by defining a special automorphism of a Chevalley group and taking the fixed points of that automorphism. The automorphism defining these subgroups (the so-called twisted groups) can be derived from a non-trivial graph automorphism of the Dynkin diagram related to the Chevalley group. In this way one can get other classes of simple groups, namely, the Steinberg groups ${}^2A_l(q^2)$, ${}^2D_l(q^2)$, ${}^3D_4(q^3)$, ${}^2E_6(q^2)$ and the Suzuki-Ree groups ${}^2B_2(2^{2n+1})$, ${}^2F_4(2^{2n+1})$, ${}^2G_2(3^{2n+1})$. However, the discussion of these groups (as well as many important properties of the Chevalley groups) is beyond the scope of this note.

4 Conclusion

The goal of our note was to demonstrate the connection between simple Lie groups and finite simple groups of Lie type. It turnes out that this connection can be detected through Lie algebras. Starting from a simple (real) Lie group G one can first define its Lie algebra $\mathfrak{g}$. Then $\mathfrak{g}$ will be a simple real Lie algebra. Now, either $\mathfrak{g} \otimes_{\mathbb{R}} \mathbb{C} \simeq \mathfrak{s}$ or $\mathfrak{g} \otimes_{\mathbb{R}} \mathbb{C} \simeq \mathfrak{s} \oplus \mathfrak{s}$ for some simple complex Lie algebra $\mathfrak{s} \in \{A_l, B_l, C_l, D_l, E_6, E_7, E_8, F_4, G_2\}$. Thus, every simple Lie group is uniquely related to a simple complex Lie algebra.

On the other hand, all the finite simple groups of Lie type can also be defined from starting a simple complex Lie algebra. In that way the Dynkin diagrams define a partition both on the set of simple Lie groups and on the set of simple groups of Lie type. Although there is no one-to-one correspondence between types of simple Lie groups and finite simple groups of Lie type, we can say, that a simple Lie group is “similar” to a finite simple group of Lie type if they correspond to the same Dynkin diagram.

What Functors and Categories are Suitable for Relating Mathematical and Biological Models?

Chrystopher L. Nehaniv[1], Maria Schilstra[1], Nicoline den Breems[2], Alastair Munro[2], and Daniel Schreckling[3]

[1] Royal Society Wolfson Biocomputation Research Lab
Centre for Computer Science and Informatics Research
University of Hertfordshire, Hatfield, United Kingdom
[c.l.nehaniv,m.j.1.schilstra]]@herts.ac.uk

[2] Medical Research Institute, Division of Population Health Sciences
University of Dundee, Dundee, United Kingdom
[n.denbreems,a.j.munro]@dundee.ac.uk

[3] Institute of IT-Security and Security Law
University of Passau, Passau, Germany
ds@sec.uni-passau.de

Abstract. Biochemical and biological dynamical systems are frequently described by directed graphs with labelled arrows and nodes. Often no strict convention is followed for the semantics of these graphs, but nevertheless such graphs are often given rigorous mathematical interpretations as specifications of dynamical systems, e.g. giving detailed stoichiometric specification of chemical reactions or other transformations. These directed graphs are typically labelled with kinetic rate constants, such as those involved in, e.g., Michaelis-Menten kinetics or rates of genetic transcription, and these 'constants' are often subject to modulation by many factors. Often graphical representations of these dynamical systems do not uniquely determine them (even for the same author within the same scientific paper), but require additional assumptions that systems biologists can deduce from the text to further constrain the "data flow" expressed in the graphs and processes behind this flow (from which one can, e.g., set up a well-defined system of ordinary differential equations(ODEs)). Systems biologists construct differential equation models, or Petri-net and automata models, based on such directed graphs, augmented by additional knowledge. Conversely, automatic tools exist that produce graphical networks based on ODE descriptions, and increasingly journals and scientific databases include and require SMBL (Systems Biology Markup Language) formulations of biological models. For such graphical models, computational tools support selection of subsets of variables, extension via new reactions, and Petri-net simulation or numerical integration of solutions, etc. To what extent can these constructions be made functorial so that mathematical structure can be used to relate solutions and dynamics in one kind of model to other kinds? For the systems biologist, inclusion, restriction, and joining of models are very simple, natural operations. Their functorial properties in this domain have apparently not been studied, although it is well-known that they can fundamentally affect the character of model dynamics. Directed graphs also have spectral properties associated to the canonical adjacency matrices describing them, as well as automata and transformation semigroups canonically associated to them (whose subgroups may correspond to "natural subsystems"). This raises the questions of what are categories appropriate for achieving functorial correspondences between these domains of interest - what are appropriate choices for defining their objects and morphisms, and what are the functors and adjoint relationships between these categories? While the heuristic use of simplified models in specific contexts of systems biology has certainly been fruitful, one must recognize that the model is a cartoon version of events and, perhaps, not the best basis upon which to draw conclusions concerning general principles of the fundamentals of biological systems. Nevertheless, constructions of combining, merging, and restricting models is carried out routinely, and functorial understanding of these operations would clarify their significance, and might provide a framework for thinking about scaling up complex systems and computation based on interaction. We discuss some initial explorations of these questions and consider some of their implications for the mathematical and biological foundations of interaction computing.

Algebra and Symmetry Structure in Discrete Models of Biochemical and Biological Systems: Natural Subsystems, Permutation Groups, and the Weak Control Hierarchy

Chrystopher L. Nehaniv[1], John Rhodes[2], Attila Egri-Nagy[3,1], Paolo Dini[1] and Maria J. Schilstra[1]

[1] Royal Society Wolfson Biocomputation Research Lab
Centre for Computer Science and Informatics Research
University of Hertfordshire, Hatfield, United Kingdom
[c.l.nehaniv,p.dini,m.j.1.schilstra]]@herts.ac.uk

[2] Department of Mathematics, University of California, Berkeley
Berkeley CA 94720 USA
rhodes@math.berkeley.edu

[3] School of Computing, Engineering and Mathematics
University of Western Sydney, NSW, Australia
a.egri-nagy@uws.edu.au

Abstract. Finite discrete automata models of biological systems such as the lac operon, Krebs cycle, and p53-mdm2 genetic regulatory system constructed directly or from Boolean networks or Petri nets have canonically associated algebraic structures – their transformation semigroups. These often contain permutation groups, which are local substructures exhibiting symmetry, and correspond to "pools of reversibility" or "natural subsystems". These natural subsystems are related to one another in a hierarchically structured manner. This relation is captured by the notion of "weak control" which hierarchically relates these natural subsystems (under which equivalent subsystems are in fact isomorphic as permutation groups). We exhibit examples of natural subsystems arising from several biological examples and describe their weak control hierarchies in detail.

TECtonIc: Tick-Enforceable Control-Flow Integrity for Web-Applications

Eric Rothstein Morris

Institute of IT-Security and Security Law
University of Passau, Passau, Germany
er@sec.uni-passau.de

Abstract. Control-Flow Integrity (CFI) is a basic safety property of systems. Its enforcement prevents the user from arbitrarily controlling the behaviour of an application, and is therefore of practical relevance. CFI enforcement has been addressed for standalone machines using inlined reference monitors and binary rewriting; but for web applications, which are today an increasing trend, CFI violations can be triggered by the user by issuing HTTP requests that can be processed by the web application in an arbitrary order. It is then clear that CFI enforcement needs to be addressed also at the level of HTTP requests/responses. In this work, we present a formal framework for the definition and composition of time-sensitive CFI policies for web applications.

1 Foreword

This paper is a preprint work, and builds theory over the unpublished work [8]. The main objective of these two works is to provide an introduction to the definition and enforcement (via composition) of the enforceable policies introduced by Schneider in [9] and find applications in the security domain of web applications. Neither this work nor [8] is self-contained, but they refer the reader to the relevant literature.

2 Introduction

The Drive-Through Metaphor. Alice is currently working at the local burger restaurant which offers drive-through service, so clients can order and purchase their food without leaving their cars. Today is Alice's turn to take orders from the drive-through, and she knows the protocol:

1. Alice waits for a customer to greet her through the speaker
2. Alice greets the client through the microphone, and asks him or her what he or she would like to order
3. The client states what the order would be
4. Alice confirms the order. If the client would like to change something, then return to previous step.
5. Alice passes the order to the kitchen and tells the client how much the order costs
6. The client pays the order
7. Alice hands the order to the client with a receipt

Evil Bob is hungry. He is willing to test his luck and see if he is able to get free food from the local burger restaurant, but he is unaware that the protocol was designed by the company in order to protect itself from clients that would like to steal food without paying. Bob goes to the drive-through and greets Alice. She greets Bob, according to the protocol, and asks Bob what he would like to eat. Bob orders a big burger and confirms the order. Alice tells him that he should pay $10 for his order, and asks the people in the kitchen to prepare it. Subsequently, Alice asks Bob for the $10. Bob tells her to hand over the food and he would give the money afterwards. Alice refuses, and asks Bob again

for the money. Bob insists unsuccessfully, desists and goes away. In this scenario, Alice prevented the order of actions from changing, in spite of Bob's requests. In this case, the *integrity of the flow of control* or *control-flow integrity* was preserved.

2.1 Control-Flow Integrity for Web Applications

Control-flow integrity for web applications is closely addresses the interaction between two principals: the *user* who issues a series of requests (acting as a master) and the *web application* which processes the requests that the user issues (acting as a slave). Our informal definition of Control-flow integrity (CFI) is inspired by definitions by Braun et al. in [4] and by Jayaraman et al. in [5]: a web application satisfies control-flow integrity if the user is not able to control the behaviour of the web application by arbitrarily issuing requests, and the web application only processes the requests issued by the user if and only if those requests are somehow *expected*. CFI can also be understood as the invariant property that states that the interaction between principals sticks to a particular protocol in spite of the attempts of some (not all) principals to try *not* to stick to it.

Interaction between principals and messages sent among them follow a set of rules established by the developer in the form of a protocol, which requires the developer of the web application to list two main components:

1. A set of *service requests* that can be issued to request the web application to perform services.
2. The *ordering* in which these *service requests* are expected. This ordering is also known as the *defined web application behaviour*.

While we consider web applications to be trusted, we do not trust users of such web applications. Therefore, CFI enforcement mechanisms cannot rely on the users issuing the correct requests. This is a reasonable restriction, as a user might not know the protocol; and even if it[1] did, the user might try not to follow it. As we consider the web application to be trusted, CFI enforcement takes place on the web application side (i.e. server-side). This poses several challenges, predominantly *authentication* of user requests, which is necessary for the purposes of tracking; and *tracking* of each user's previous actions, which is necessary for the purposes of defining the user state consistently. However, this work only addresses the aspects related to the tracking of each user's previous actions in order to consistently obtain the user's state, so we assume the existence of a correct authentication system for user requests.

2.2 Goals of this Work

The motivation behind his work is that we are looking for a way to define global web application time-sensitive policies by composing smaller, simpler policies that address individual processes; and have certainty that their enforcement mechanisms (each one enforcing a single policy) are able work well together without causing any interference among them. In Section 3.1 we present a first insight into the definition of *TECtonIc policies*, short for "Tick-Enforceable Control-Flow Integrity policies". Later, in Section 4 we address the topic of policy composition, and we provide formalisms to show explicitly how TECtonIc policies may be composed either in parallel or in a hierarchy. Section 5 introduces the notion of time, captured by *ticks* in their different types. Finally, as this is not a full work, we present a roadmap that guides research in this direction, hoping that in the future a well-founded theory of enforceable security policies for web applications proves useful for developers in the creation of new web applications.

[1] We refer to the user as *it*, since computers that are able to automatically issue requests may also be considered users of a web application

3 TECtonIc Policy Definition

There are several types of security policies, and they can be classified depending on the nature of the mechanism required to enforce them. Security enforcement mechanisms can be represented as state machines, and they range in complexity from deterministic finite automata to edit automata, as shown by Bielova in [2], where the author classified the different kinds of security properties by their enforcement mechanisms.

We consider CFI policies to represent a safety property based on the work by Ligatti in [6] and also according to the theory presented by Schneider in [9]. Safety properties describe a behaviour in which 'nothing bad ever happens', and in order to enforce them it is enough to enforce that no finite prefix of a valid execution is ever invalid, i.e. any sub-execution satisfies the safety property. TECtonIc policies are redefined versions of the policies presented in [8], work which also presents how a CFI policy can be enforced by means of a finite-state transducer or Mealy automaton.

We consider the actions of a user not to interfere with the current state of another user of the same category. Therefore, TECtonIc policies address only a single category of users, and address only one user of that category. In the case where several users of different categories are involved (i.e. a super-user that enables the account of another user) we consider the users to communicate through the web application, so direct communication occurs only between user and web application, and there is no direct message exchange between users. This simplifies the protocol definition, and is useful for the placement of enforcement mechanisms; as they would monitor messages that are exchanged, acting similarly to a proxy. In this section we provide the definitions and terminology for the definition of TECtonIc policies and a couple of examples to illustrate the definition of such policies.

3.1 TECtonIc Policies

As stated in Section 2.1, TECtonIc policies are closely related to a protocol that describes the way the users and the web application interact. Such protocol is defined[2] by the developer of the web application through two elements mainly:

1. A set of *service requests* that can be issued to request the web application to perform services.
2. The *ordering* in which these *service requests* are expected. This ordering is also known as the *defined web application behaviour*, and is closely related to the *structure* of the TECtonIc Policy.

The following definitions provide more details about the elements mentioned above:

Definition 1 (Set of Service Request Identifiers)**.** The *set of service request identifiers* Σ is a finite set of unique, atomic[3] service requests. We consider Σ to be fixed, in the sense that services are not added or removed dynamically by the policy. A pair $(\sigma, \overrightarrow{p})$ is called a *service request*, with $\sigma \in \Sigma$ and $\overrightarrow{p_i}$ being a vector of parameters with arbitrary finite length (even 0).[4]

The user requests services to the web application by sending a series of service requests, each of which is treated (not necessarily in sequence) by the web application. When a user invokes a service, it usually sends also a vector of parameter values to invoke the service with.

[2] The developer does not have to create a protocol, but rather uses or piggybacks on an existing protocol.
[3] Here we mean unique and atomic in the sense that two different services cannot have the same identifier, and that each identifier maps to one and only one service unambiguously, to avoid confusion.
[4] In this work we do not discuss more restrictions over parameters.

Definition 2 (Service Request Sequence)**.** A *service request sequence* $(\sigma_1, \overrightarrow{p_1})(\sigma_2, \overrightarrow{p_2}) \dots (\sigma_n, \overrightarrow{p_n})$ is a sequence of service requests. Service request sequences are not necessarily finite. In addition, the order in which service requests appear in a sequence is not defined by the user but rather by the server that hosts the web application, as it attempts to handle incoming requests.

Now, in order to define the ordering in which service requests are expected, we use deterministic finite Mealy automata (DFMA) to structure the TECtonIc policies, as they are a useful formalism for describing what happens with each service request whenever it is received. Several DFMAs can be used to express individual processes. For example, logging in, upgrading an account, changing a password, etc. Ideally, the developer defines a TECtonIc policy per linear process, and then composes them, either in parallel or hierarchically. We provide some examples of TECtonIc policies in Section 3.2. The following definition is an extension of the one provided in [8] for CFI policies:

Definition 3 (TECtonIc Policy)**.** A *Tick-Enforceable Control-Flow Integrity Policy* is a (partial) *deterministic finite* Mealy automaton $T = (Q^T, \Sigma_{in}^T, \Psi_{in}^T, \Sigma_{out}^T, \Psi_{out}^T, \gamma^T, \delta^T, \mathbf{start^T})$ that represents a process P where:

- Q^T is the finite set of states. Each state q represents a tracking location of the user inside the process P, and q has the following components:
  1. A unique label used to refer to state q. This label is usually given by the developer i.e. loggedIn, loggedOut; but might also be defined by the values of variables that are relevant to the policy. We refer to states by using their identifiers.
  2. A decidable syntactic condition over parameters used for the purposes of enforcement. One of these conditions could be, for example, 'parameters are sanitised'.
  3. A tick-value $q_\mu \in \mathbb{N}$ used for the purposes of tick-enforcement (see Section 5). If $q_\mu = 0$, then there are no time restrictions for this state.
- Σ_{in}^T is the set of service request identifiers that this policy understands and can be issued by principals that want to use the web application services, usually a user through a user-agent (browser). We state that $\Sigma_{in}^T \subseteq \Sigma$, as a policy might refer only to a subset of services. Σ_{out}^T is a set of messages that can be sent to the user as a response to a service request.[5] We define the *set of messages* $\Sigma^T = \Sigma_{in}^T \cup \Sigma_{out}^T$, and we refer to their members as messages.
- Ψ_{in}^T is a set of internal signals used to let the policy enforcement mechanism know that something happened. Among these signals are the responses issued by the web application as a result of a service request. Similarly, Ψ_{out}^T is the set of signals that are used by the enforcement mechanism to let other entities (except the user) know about occurring events. In order to avoid overhead, usually the service requests issued by the user are forwarded to the web application (i.e. elements in $\Psi_{in}^T \cap \Sigma_{out}^T$), but signals will also prove useful for tick-enforcement (see Section 5). We name the *set of signals* $\Psi^T = \Psi_{in}^T \cup \Psi_{out}^T$, and we refer to their members as signals.
  Both Σ_{in}^T and Ψ_{in}^T are disjoint sets, so a user can never issue an internal signal. Similarly for Σ_{out}^T and Ψ_{out}^T. Elements of Σ_{in}^T and Ψ_{in}^T are called *inputs* and elements from Σ_{out}^T and Ψ_{out}^T are called outputs.
- **start** is the initial state of the policy. It has particular properties:
  1. Its unique identifier is *start*.
  2. There is no condition over parameters as the start state should not have any. **true** is used for purposes of consistency.
  3. The tick-value μ is equal to 0.
- δ^T is an overloaded transition function with one *total* instance $\delta^T : Q^T \times \Sigma_{in}^T \to Q^T$, and another (partial) instance $\delta^T : Q^T \times \Psi_{in}^T \to Q^T$. We overload this function as both instances have the same purpose (together they serve the role of the transition function of the policy), but we need to state that one instance must be total while the other may be partial. Unless specified, the function δ^T refers to the *general transition function of the policy*, defined as $\delta^T : Q^T \times (\Sigma_{in}^T \uplus \Psi_{in}^T) \to Q^T$, which joins both overloaded functions into one.
- γ^T is an overloaded transduction function with one *total* instance $\gamma^T : Q^T \times \Sigma_{in}^T \to \Psi_{out}^{*T}$ and another (partial) instance $\gamma^T : Q^T \times \Psi_{in}^T \to \Sigma_{out}^{*T}$, where Ψ_{out}^{*T} is the set of finite length sequences built using elements of Ψ_{out}^T (similarly for the set Σ_{out}^{*T}).

[5] Note that both Σ_{in}^T and Σ_{out}^T are sets of messages that are part of the protocol defined by the developer for communication between the user and the web application. The elements of Σ_{out}^T must be meaningful for the user agent, the entity that sends and receives messages on behalf of the user.

3.2 TECtonIc Policy Examples

To illustrate how to define TECtonIc policies, we define two policies that are used to describe two common processes in web applications: login/logout process, and upgrading a user account.

Logging-in Policy: Web applications usually handle some sort of authentication mechanism to identify users, allowing users to store some relevant information related to them and their accounts, and to protect such data from other users' attempts to read it. In this paper we present a very simple login/logout policy based on login-password verification:

Let L be the TECtonIc login/logout policy. L's components are:

- The state set $\{\mathbf{start^L}, LoggedIn, NotLoggedIn\}$, where:
  - The state $LoggedIn$ has a tick-value of 100
  - The state $NotLoggedIn$ has a tick-value of 0
- The service identifiers used by the users are $\Sigma^L_{in} = \{login, logout\}$
- The message responses sent to the users are $\Sigma^L_{out} = \{login_ok, logout_ok, error\}$
- The received internal signals are $\Psi^L_{in} = \{TIMEOUT, LOGIN_OK, LOGIN_ERROR\}$
- The sent internal signals are $\Psi^L_{out} = \{LOGIN, LOGOUT\}$
- The functions δ^L and γ^L are shown in Figure 1, where user input messages are coded lowercase blue (Σ^L_{in}), message responses for users are coded lowercase purple (Σ^L_{out}), signals towards the web application are coded uppercase black (Ψ^L_{out}), and signal inputs are coded uppercase green (Ψ^L_{in}).

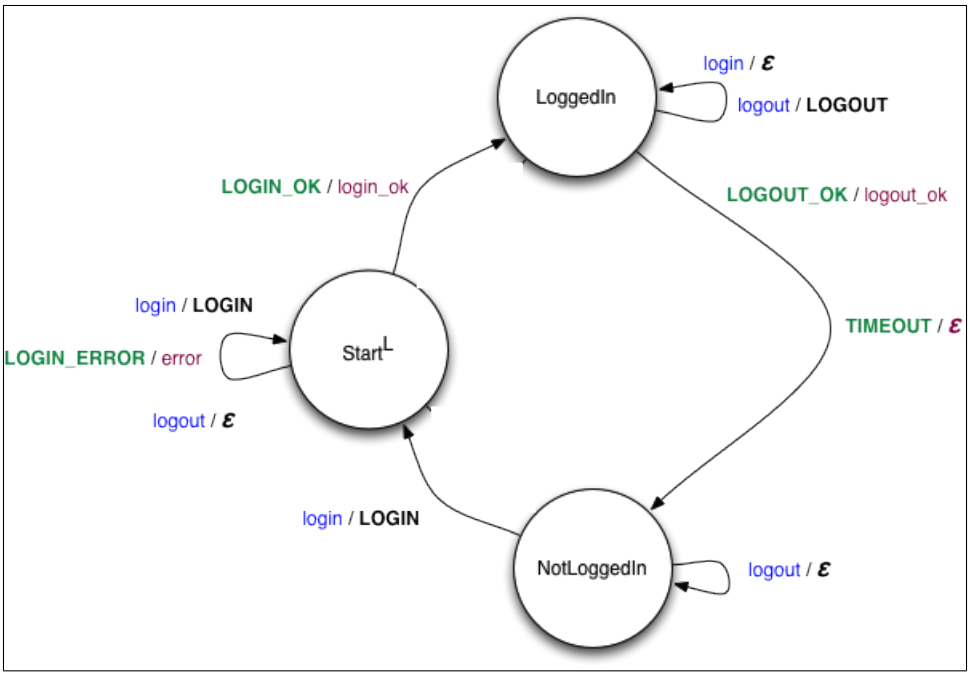


Fig. 1: **Transition and transduction functions for the TECtonIc policy L. Messages received by the automaton appear on the left side and messages sent appear on the right side. The symbol ϵ represents that no signal or message is issued as a response to the input message or signal.**

There are several remarks to be made at this point concerning the transition functions:

- Usually, authentication mechanisms match some input provided by the user against a stored value in a database. Since TECtonIc policies do not have any kind of data storage, the enforcement mechanism needs to wait for a confirmation from the access control mechanism stating that either

the authentication was successful or not, and this confirmation is represented by the $LOGIN_OK$ and $ERROR$ internal signals.
- At the moment of policy definition, the developer should focus on how the state of the user should change when particular messages or signals are received, *not* about how those signals are sent or received. This matter should be addressed by the enforcement architecture. It is advisable that the developer model their processes following state-of-the-art standards.
- It is not necessary to specify all transitions for every internal signal in every state Ψ_{in}^{L}, as those messages are responses issued by the web application at very particular moments (specified by the developers). However, it is necessary to specify all transitions for every message Σ_{in}^{L}, to consider all possible user actions.
- The $TIMEOUT$ input signal is issued by an entity different from the web application (a clock), but it is processed in the same fashion as a signal issued by the web application. Every state with a tick value different from 0 *must* have a $TIMEOUT$ transition (see Section 5).

Upgrading Account Policy. To earn extra privileges, users are usually given the option to upgrade their accounts for a certain price. This policy is interesting to model, as the user is asked to perform actions that take place outside the server that hosts the web application (e.g. payment procedure is handled by a trusted third party cashier service).

Let U be the TECtonIc upgrading policy. U's components are:

- The state set $\{\mathbf{start^U}, Redirect, Upgraded, Waiting\}$, where:
  - The state $Redirect$ has a tick value of 0
  - The state $Upgraded$ has a tick value of 0
  - The state $Waiting$ has a tick value of 1000
  - The state $NotUpgraded$ has a tick value of 0
- The service identifiers used by the users are $\Sigma_{in}^{U} = \{upgrade, show_receipt\}$
- The message responses sent to the users are $\Sigma_{out}^{U} = \{upgrade_ok, go_to_cashier, payment_error, receipt_error\}$
- The received internal signals are $\Psi_{in}^{U} = \{DOWNGRADE, UPGRADE_OK, REDIRECT, RECEIPT_ERROR, TIMEOUT\}$
- The sent internal signals are $\Psi_{out}^{U} = \{CONFIRM_RECEIPT, UPGRADE\}$
- The functions δ^{U} and γ^{U} are shown in Figure 2.

It is important to note that if the user upgraded its account and closed its session, it should be considered upgraded next time it logs in. We also restrict users from upgrading their accounts unless they have logged in. Therefore, there is a clear dependency between the policy U and the policy L that is not covered by the definition of the individual policies, but rather is made explicit via hierarchical composition, which we explain in the following sections.

4 TECtonIc Policy Composition

The motivation for policy composition is that the developer focuses only on an individual procedure and models it through a single TECtonIc policy, which states how messages should be exchanged in order to guide the user into finishing such procedure. In the previous sections we explained how TECtonIc policies are defined without any explicit dependencies among them, like individual building blocks. Now, to make these dependencies explicit, these building blocks can be glued together to create *composed* policies. Policies can be composed in several ways, but we only explore two different methods: parallel composition and hierarchical composition.

To reason about dependencies, let us use the two example TECtonIc policies L and U presented in Section 3.2. We are interested in preventing users from upgrading their account if they are not already

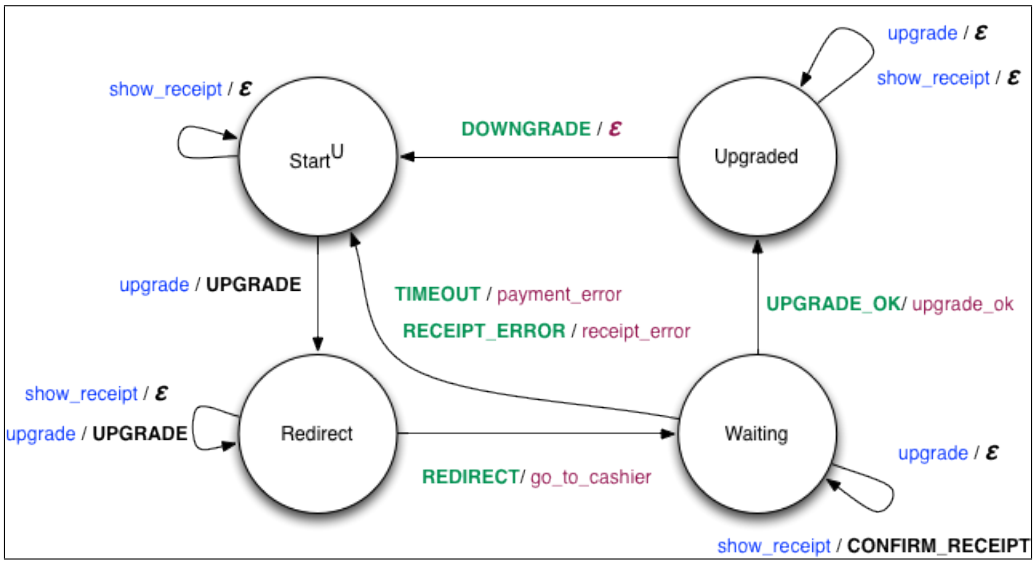


Fig. 2: **Transition and transduction functions for the TEConIc policy U. The separation between the redirect and the waiting state prevents the user from upgrading**

logged in; and one way of enforcing this requirement would be to check whether the state of the user in the login policy L is *LoggedIn* when an *upgrade* message from the user is received, and handle the request accordingly (see Figure 3). In this case, the current state of the policy L is needed by policy U to validate the transitions triggered by the *upgrade* message, so the decision of policy U *depends* on the current state of the process modelled by policy L. As the enforcement of policy L does not really need information about the current state of the process modelled by policy U, we say that L does not depend on U. To capture this notion formally we define the *policy dependence relation* $\lhd$, which is a reflexive and transitive relation.

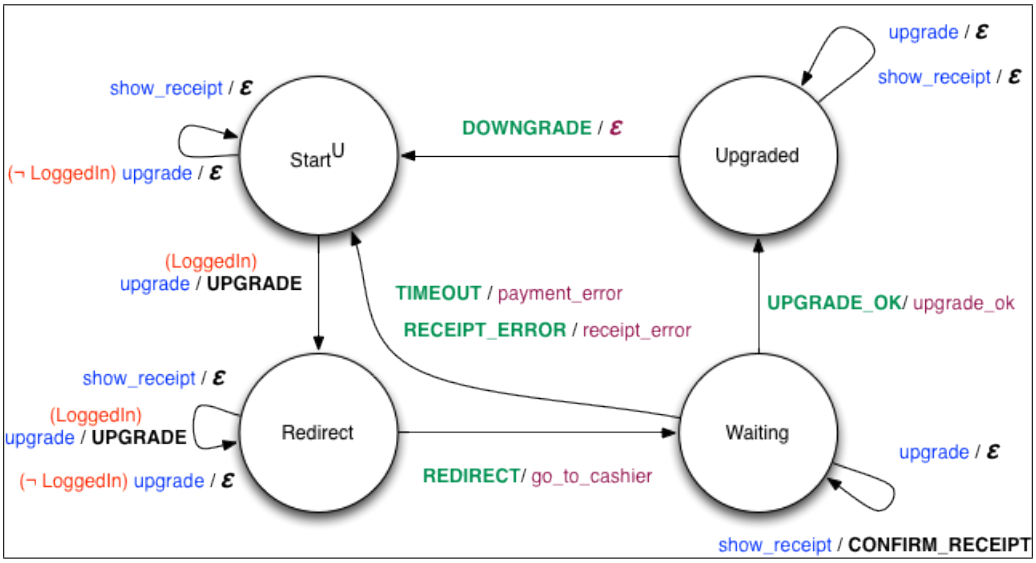


Fig. 3: **Dependencies are made explicit in red for the upgrade transitions that issue signals in the policy U, and they refer to policy L.**

4.1 Policy Dependence

For the set of TEConIc policies of a web application it is possible create a policy dependence graph which displays graphically the *policy dependence relation* $\lhd$, as the one shown in Figure 4. The relation

$\triangleleft$ plays an important role at the time of policy composition, as it gives the developer hints about what to expect when both enforcement mechanisms work together, and provides ground for a unified, global policy.

Consider $\triangleleft^+$ the transitive closure of $\triangleleft$. Now, for two different policies A and B, if $A \triangleleft^+ B$ we claim that A *depends* on B. If **neither** $A \triangleleft^+ B$ nor $B \triangleleft^+ A$, then we say that A and B are *independent* policies.

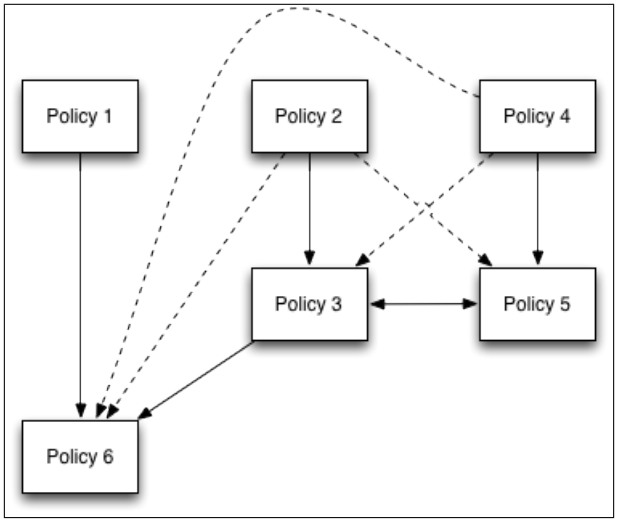


Fig. 4: **Example of a policy dependency graph. Explicit dependencies are represented with solid arrows and transitive dependencies with dotted arrows.**

Whenever two policies A and B are independent, they can be placed at the same level of the dependency graph. If A depends on B, then A is placed on a lower level than B, unless B also depends on A, in which case the two policies are *strongly dependent* and are placed on the same level.[6] In the following sections, we provide formal notions of composed policies, including parallel and hierarchically composed policies.

4.2 Parallel Composition of Policies

When two policies A and B are independent, we define the parallel composition of A and B as the join policy $A \parallel B$. The components of the policy $A \parallel B$ are defined as follows:

- The state set $Q^{A\|B} = Q^A \times Q^B$, and $\mathbf{start^{A\|B}} = (\mathbf{start^A}, \mathbf{start^B})$
- The set of messages issued by the users $\Sigma_{in}^{A\|B} = \Sigma_{in}^A \cup \Sigma_{in}^B$
- The set of message responses sent to the users $\Sigma_{out}^{A\|B} = \Sigma_{out}^A \cup \Sigma_{out}^B$
- The set of received internal signals $\Psi_{in}^{A\|B} = \Psi_{in}^A \cup \Psi_{in}^B$.
- The set of sent internal signals $\Psi_{out}^{A\|B} = \Psi_{out}^A \cup \Psi_{out}^B$

Before defining the transition and transduction functions, we *extend* the transition functions δ^A and δ^B to functions $\delta^{A \triangleleft \emptyset}$ and $\delta^{B \triangleleft \emptyset}$ respectively. This extension is necessary as the combination of policies unites the sets of input messages of each policy, rendering the transition functions of policies A and B not total for policy $A \parallel B$.

[6] In this work we do not study strongly dependent policies.

The values of $\delta^{A \triangleleft \emptyset}(q_a, \sigma)$ and $\delta^{B \triangleleft \emptyset}(q_b, \sigma)$ for $q_a \in Q^A$, $q_b \in Q^B$ and $\sigma \in (\Sigma^A_{in} \cup \Sigma^B_{in}) \uplus (\Psi^A_{in} \cup \Psi^B_{in})$ are defined as follows:

- If $\delta^A(q_a, \sigma)$ is defined, then $\delta^{A \triangleleft \emptyset}(q_a, \sigma) = \delta^A(q_a, \sigma)$, otherwise $\delta^{A \triangleleft \emptyset}(q_a, \sigma) = q_a$
- Similarly, if $\delta^B(q_b, \sigma)$ is defined, then $\delta^{B \triangleleft \emptyset}(q_b, \sigma) = \delta^B(q_b, \sigma)$, otherwise $\delta^{B \triangleleft \emptyset}(q_b, \sigma) = q_b$

Using these new functions, it is possible to define the transition function of the composed policy:

- The function $\delta^{A \| B} : Q^{A \| B} \times (\Sigma^{A \| B}_{in} \uplus \Psi^{A \| B}_{in}) \to Q^{A \| B}$ applied to a state (q_a, q_b) on input σ is equal to the state $(\delta^{A \triangleleft \emptyset}(q_a, \sigma), \delta^{B \triangleleft \emptyset}(q_b, \sigma))$.

We also extend the transduction functions γ^A and γ^B to functions $\gamma^{A \triangleleft \emptyset}$ and $\gamma^{B \triangleleft \emptyset}$, respectively.

The values of $\gamma^{A \triangleleft \emptyset}(q_a, \sigma)$ and $\gamma^{B \triangleleft \emptyset}(q_b, \sigma)$ for $q_a \in Q^A$, $q_b \in Q^B$ and $\sigma \in (\Sigma^A_{in} \cup \Sigma^B_{in}) \uplus (\Psi^A_{in} \cup \Psi^B_{in})$ are defined as follows:

- If $\gamma^A(q_a, \sigma)$ is defined, then $\gamma^{A \triangleleft \emptyset}(q_a, \sigma) = \gamma^A(q_a, \sigma)$, otherwise $\gamma^{A \triangleleft \emptyset}(q_a, \sigma) = \epsilon$
- Similarly, if $\gamma^B(q_b, \sigma)$ is defined, then $\gamma^{B \triangleleft \emptyset}(q_b, \sigma) = \gamma^B(q_b, \sigma)$, otherwise $\gamma^{B \triangleleft \emptyset}(q_b, \sigma) = \epsilon$.

The transduction function of the composed policy:

- The function $\gamma^{A \| B} : Q^{A \| B} \times (\Sigma^{A \| B}_{in} \uplus \Psi^{A \| B}_{in}) \to (\Sigma^{*A \| B}_{out} \uplus \Psi^{*A \| B}_{out})$ applied to a state (q_a, q_b) on input σ is equal to the sequence $\gamma^{A \triangleleft \emptyset}(q_a, \sigma)\gamma^{B \triangleleft \emptyset}(q_b, \sigma)$.

We omit a proof that shows that the parallel composition of two TECtonIc policies is also a TECtonIc policy, as we consider it to be quite natural.

4.3 Hierarchical Composition of Policies

Hierarchical composition is a particular method of composition in which the execution of a machine depends directly on the state or outputs of another machine, and not on the output produced by it after executing.

When policy A depends on policy B, we define the hierarchical composition of A and B as the join policy $A \wr B$. The components of the policy $A \wr B$ are defined as follows:

- The state set $Q^{A \wr B} = Q^A \times Q^B$, and $\mathbf{start}^{\mathbf{A} \wr \mathbf{B}} = (\mathbf{start}^{\mathbf{A}}, \mathbf{start}^{\mathbf{B}})$
- The set of messages issued by the users $\Sigma^{A \wr B}_{in} = \Sigma^A_{in} \cup \Sigma^B_{in}$
- The set of message responses sent to the users $\Sigma^{A \wr B}_{out} = \Sigma^A_{out} \cup \Sigma^B_{out}$
- The set of received internal signals $\Psi^{A \wr B}_{in} = \Psi^A_{in} \cup \Psi^B_{in}$.
- The set of sent internal signals $\Psi^{A \wr B}_{out} = \Psi^A_{out} \cup \Psi^B_{out}$.

To define the functions $\delta^{A \wr B}$ and $\gamma^{A \wr B}$, it is necessary to know the current state of B before applying transitions or issuing messages/signals in A. To formally capture this concept, we define the extended total transition function $\delta^{A \triangleleft B} : Q^B \times Q^A \times \Sigma^A_{in} \uplus \Psi^A_{in} \to Q^A$. Similarly, we define the extended total transduction function $\gamma^{A \triangleleft B} : Q^B \times Q^A \times (\Sigma^A_{in} \uplus \Psi^A_{in}) \to (\Sigma^{*A}_{out} \uplus \Psi^{*A}_{out})$. We explain how to give values to these functions shortly.

In cases where the current state of B ($q_b \in Q^B$) is irrelevant for a transition in the current state of A on σ ($\sigma \in \Sigma_{in}^A \uplus \Psi_{in}^A$) ($q_a \in Q^A$), the value of $\delta^{A \triangleleft B}(q_b, q_a, \sigma)$ matches $\delta^A(q_a, \sigma)$; otherwise, it needs to be given a value, in the form of a state in Q^A. Similarly for the extended transduction function $\gamma^{A \triangleleft B}$. For a rough idea of what this transition function is extended to, consider the current state of B, see the changes to policy U between Figure 2 and Figure 3 as it considers the current state of L on some transitions.

Once the function $\delta^{A \triangleleft B}$ has been defined, we can define a transition function of the composed policy $\delta^{A \wr B} : Q^{A \wr B} \times (\Sigma_{in}^{A \wr B} \uplus \Psi_{in}^{A \wr B}) \to Q^{A \wr B}$. More precisely, $\delta^{A \wr B}$ will be applied to a state (q_a, q_b) on input σ as follows:

- If $\delta^{A \triangleleft B}(q_b, q_a, \sigma)$ and $\delta^B(q_b, \sigma)$ are defined, then $\delta^{A \wr B}((q_a, q_b), \sigma) = (\delta^{A \triangleleft B}(q_b, q_a, \sigma), \delta^B(q_b, \sigma))$
- If $\delta^{A \triangleleft B}(q_b, q_a, \sigma)$ is defined but $\delta^B(q_b, \sigma)$ is not, then $\delta^{A \wr B}((q_a, q_b), \sigma) = (\delta^{A \triangleleft B}(q_b, q_a, \sigma), q_b)$.
- Similarly, if $\delta^B(q_b, \sigma)$ is defined but $\delta^{A \triangleleft B}(q_b, q_a, \sigma)$ is not, then $\delta^{A \wr B}((q_a, q_b), \sigma) = (q_a, \delta^B(q_b, \sigma))$.

We also use the functions $\gamma^{A \triangleleft B}$ to define the transduction functions $\gamma^{A \wr B} : Q^{A \wr B} \times \Sigma_{in}^{A \wr B} \to \Psi_{out}^{*A \wr B}$ and $\gamma^{A \wr B} : Q^{A \wr B} \times \Psi_{in}^{A \wr B} \to \Sigma_{out}^{*A \wr B}$. First, the function $\gamma^{A \wr B} : Q^{A \wr B} \times \Sigma_{in}^{A \wr B} \to \Psi_{out}^{*A \wr B}$ is applied to a state (q_a, q_b) on input σ as follows:

- Since $\gamma^{A \triangleleft B}(q_b, q_a, \sigma)$ and $\gamma^B(q_b, \sigma)$ are defined, $\gamma^{A \wr B}((q_a, q_b), \sigma)$ is equal to $\gamma^B(q_b, \sigma)\gamma^{A \triangleleft B}(q_b, q_a, \sigma)$.

Second, the function $\gamma^{A \wr B} : Q^{A \wr B} \times \Psi_{in}^{A \wr B} \to \Sigma_{out}^{*A \wr B}$ applied to a state (q_a, q_b) on input ψ is defined as follows:

- If $\gamma^A(q_a, \psi)$ and $\gamma^B(q_b, \psi)$ are both defined, then $\gamma^{A \wr B}((q_a, q_b), \psi)$ is equal to $\gamma^B(q_b, \psi)\gamma^{A \triangleleft B}(q_b, q_a, \psi)$.
- If $\gamma^A(q_a, \psi)$ is defined but $\gamma^B(q_b, \psi)$ is not, then $\gamma^{A \wr B}((q_a, q_b), \psi) = \gamma^A(q_a, \psi)$.
- Similarly, if $\gamma^B(q_b, \psi)$ is defined but $\gamma^A(q_a, \psi)$ is not, then $\gamma^{A \wr B}((q_a, q_b), \psi) = \gamma^B(q_b, \psi)$.

Several questions arise naturally concerning the execution of machines composed in a hierarchy; for example: when does A observe the current state of B? Does B report its current state to A? Can the delay of this observation/report be neglected? We hope that future versions of this work are able to formally solve these questions, and many more (see Section 6).

In the following section, we introduce the notion of tick-enforcement, which allows TECtonIc policies to state time-sensitive conditions over states.

5 Tick-Enforcement

Moore's law is a very interesting observation that states that computing power doubles every two years[7], allowing computers to perform tasks in half the time they required before. Consequently, web applications might have to redefine policies that address time restrictions approximately every two years to adapt the new computing capabilities. There is a way to provide a scalable notion of time so that it is possible to define time-sensitive policies that are scalable: by using the concept of *ticks* (short for tick-tocks).

In this paper, we present two types of ticks to define time-sensitive policies:

- A *regular tick* is an event that happens regularly after some time interval e.g. 2 millisecond.

- An *irregular tick* is an event that happens irregularly, usually issued as a response/consequence of an event happening.[7]

Regular ticks should be used to define policies of the form:

> 'Users are considered logged-out after 10 minutes and before 11 minutes of inactivity after their last request was issued'[8]

Irregular ticks should be used to define policies that depend on events happening:

> 'All enforcement monitors synchronise every 10 synchronisation ticks'[9]

Note that several types of regular and irregular ticks can be defined, including synchronisation ticks, maintenance ticks, countdown ticks, etc.

Contrary to signals and messages received and issued by the web application (which can be ignored or rejected by the enforcement mechanisms), ticks *cannot be prevented from happening*. Ticks fall then under the category of *observable actions* while signals and messages fall under the category of *controllable actions*, as presented by Basin et al. in [1].

5.1 Clocks

Enforcement mechanisms of TECtonIc policies are "equipped" with one or more clocks that produce ticks following some rules. It is important to notice that the ticks that are actually "heard" by the enforcement mechanisms are the ones that matter for the purposes of enforcement, so if a clock emits a tick every 60 seconds, then there must be a mechanism that coordinates with relative precision the emission of such 60-second tick. Consequently, ticks appear to the enforcement mechanisms as internal signals (i.e. elements of the set Ψ_{in}).

In this work, only the $TIMEOUT$ signal is studied, which is a signal that is issued once the clock has emitted a number of ticks equal to the tick-value of the current state of the policy. In Section 3.2, we showed two examples of TECtonIc policies, where each state with a tick-value greater than 0 must have a $TIMEOUT$ transition that is triggered when the clock emits the $TIMEOUT$ signal (see Figures 1 and 2).

Future versions of this work will discuss how clocks, enforcement mechanisms and the web application synchronise so that an attacker cannot exploit delays due to communication between these three principals. A first impression suggests that the action of synchronisation must be prioritised over processing of user requests, but considerations have to be made in order to keep the web application efficient, as a slow web application it is not appealing to any of its users.

6 Roadmap

In this section we compile a list of tasks to be completed and interesting research questions that contribute towards the field of enforceable policies and web application security:

[7] Note that some regular ticks can be expressed as irregular ticks that are issued every time a counter is advanced, and such counter advances regularly.
[8] The arbitrary choice of 11 minutes gives an upper bound to the time that can pass before the user is logged out.
[9] Note that this policy does not really address a process of the user, but rather an internal one. However, it still classifies as a TECtonIc policy.

Formalise Execution. When hierarchical composition is compared to serial composition, both can be found quite similar. However, serial composition is usually associated with one machine executing first, and then the output of the first machine is fed to the second as its input. With hierarchical composition, both machines can execute at the same time without any need for synchronisation. This is an interesting capability that is to be considered when defining formalisms for the execution of the enforcement mechanism, which is what binds all the concepts together and allows the enforcement mechanism to be coupled with the web application to provide security.

Tick-equivalence. It is possible to create equivalences between the different types of tick intervals from different regular clocks. It is interesting to explore the advantages and disadvantages of using a single regular clock instead of different regular clocks defined for the enforcement mechanisms.

Knowledgeable Policies. Inspired by capabilities of the principals used in DKAL (see [3]), policies can be provided with a knowledge base so that they can perform operations faster. We consider this very appealing, as enforcement mechanisms can become part of the web application and not just act as a proxy for it.

Advantages of Composition. Parallel composition is a way to compose two independent policies so that they are enforced by a single enforced mechanism. The advantages of enforcing composed policies instead of enforcing several independent policies is still an interesting topic that may prove useful for the definition of policy enforcement architectures.

Hierarchical composition for strongly dependent policies must also be handled with care in order to avoid concurrency problems e.g. deadlocks among policies. However, we believe that several solutions available in the literature can be adapted to solve these problems.

References

1. D. Basin, V. Jugé, F. Klaedtke, and E. Zalinescu. Enforceable security policies revisited. *Principles of Security and Trust*, 2012.
2. N. Bielova. A theory of constructive and predictable runtime enforcement mechanisms. (November), 2011.
3. A. Blass, G. D. E. Caso, and Y. Gurevich. An introduction to dkal. (October):1–23, 2012.
4. B. Braun, P. Gemein, H. P. Reiser, and J. Posegga. Control-Flow Integrity in Web Applications. *ESSOS*, 2012.
5. K. Jayaraman and G. Lewandowski. Enforcing request integrity in web applications. *Data and Applications Security and Privacy XXIV*, pages 225–240, 2010.
6. J. Ligatti and D. Walker. *Policy enforcement via program monitoring.* 2006.
7. G. E. Moore. Excerpts from a Conversation with Gordon Moore: Moore? s Law. *Intel Corporation*, 2005.
8. E. Rothstein, B. Braun, and H. Daniel. Enforceable Control-Flow Integrity Policies for Web Applications. Unpublished.
9. F. B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, Feb. 2000.

Holonomy Decomposition of the 3-Queens Puzzle

Eric Rothstein Morris[1], Paolo Dini[2], Chrystopher L. Nehaniv[2], Daniel Schreckling[1] and Attila Egri-Nagy[3,2]

[1] Institute of IT-Security and Security Law
University of Passau, Passau, Germany
[er,ds]@sec.uni-passau.de

[2] Royal Society Wolfson Biocomputation Research Lab
Centre for Computer Science and Informatics Research
University of Hertfordshire, Hatfield, United Kingdom
[p.dini,c.l.nehaniv]@herts.ac.uk

[3] School of Computing, Engineering and Mathematics
University of Western Sydney, NSW, Australia
a.egri-nagy@uws.edu.au

Abstract. This paper is motivated by our desire to understand how Krohn-Rhodes theory can be used for solving classical problems in computer science, and to identify the advantages and difficulties of applying such theory. For this purpose, we study the 3-queens puzzle, which consists of finding a configuration of as many queens as possible on a 3×3 chessboard where no queen threatens another. We represent the 3-queens puzzle as a graph-search problem, and formalize this graph-search with a transformation semigroup, so that it is possible to analyse it using holonomy decomposition. This yields novel, interpretable coordinate systems for this type of problem, and we provide some insights into how variations in modelling choices yield different decomposition results.

1 Introduction, Notation and Motivation

In the BIOMICS project[4] we are discretizing continuous models of biochemical pathways of the cell in order to analyse their computational properties and use them as a source of inspiration for a new model of computation that we call Interaction Computing. Our analytical perspecive is strongly algebraic, and in particular we employ the algebraic automata theory that originated with Krohn and Rhodes's work [11], which is mathematically powerful and deep. It relies on the study of algebraic objects such as transformation semigroups and wreath products, concepts which are not trivial to understand and require effort and experience to become comfortable with. The complexity and level of abstraction of Krohn-Rhodes theory means that a significant part of the research challenge lies in understanding the theory itself and in interpreting what it is telling us from an applied computer science perspective. Accordingly, this is the main objective of this paper. To this end, whereas most of the BIOMICS activities focus on analysing particular biochemical pathways, in this paper we analyse an extremely simple problem whose algorithmic properties are transparently accessible to anyone who is familiar with the basic rules of Chess, the 3-queens Chess puzzle. We present not a sanitized mathematical analysis which could be presented in two distilled pages to the few people with the requisite mathematical background, but rather we illustrate the process of understanding a particular example.

The N-queens puzzle, the general version of the 3-queens puzzle, consists of finding a configuration of N queens on an $N \times N$ chessboard such that no queen threatens another. We consider a queen to threaten another queen if both queens are located in the same row, column, or on the same diagonal line of the chessboard. Intermediate configurations, i.e. configurations involving $n < N$ queens, are also possible. Regardless of the value of n, configurations where no queen threatens another are called *valid*. When $n = N$, a valid configuration is also called a *solution*. If no valid configurations are

[4] http://www.biomicsproject.eu

possible for $n = N$, then a solution is achieved when n is the largest possible number of queens in a valid configuration. Thus, a possible way to pose the problem could be to start from an empty board and find a sequence of single-queen, then two-queen, ..., up to N-queen configurations such that each configuration is valid, adds a single queen to the previous configuration, and so that the end configuration is a solution.

To cast this problem in terms of automata theory, we identify each valid configuration with a *state* of an automaton, the empty board as the single *starting state*, and each solution with an *final state*. Loosely speaking, the act of placing a queen on the board becomes a transition from one state to another, which is another way to describe a step in an algorithm. From a computer science perspective we can then try to answer questions such as how understanding the algebraic structure underlying the model may help design or improve algorithms that solve the problem. As discussed in more detail in [5], we take the following abstract definition of automaton as a starting point:

Definition 4. A **finite automaton** can be defined as $\mathcal{A} = (A, X, \delta)$, where A is the set of *input symbols*, the *input alphabet*; X is the set of *states*, the *state space*; and δ is the *state transition function* $\delta : X \times A \to X$. Then we say that the state resulting when input symbol $a \in A$ occurs while the automaton is in state $x \in X$ is the new state $\delta(a, x) \in X$. Since we are talking about a finite-state automaton, all the objects involved are finite.

We refer to automata theory terminology as it may be more familiar to a computer science readership. However, most of the theory discussed in this paper is strongly algebraic. Therefore, we need to characterize an automaton also from the point of view of algebraic automata theory, where it corresponds to a transformation semigroup:

Definition 5. A **transformation semigroup** is a set of states X acted upon by a semigroup S of transformations, where each transformation (each element of S) is a function from X to X. Thus, it is a generalization of permutation groups (which require reversible transformations), and is called a transformation semigroup or ts for short. In symbols, we denote a ts as (X, S).

Saying that a set of transformations S is a *semigroup* is just saying that S is closed under composition. So if we have an automaton, we obtain the corresponding transformation semigroup $S = \langle A \rangle$, as the semigroup generated by actions due to input letters $a \in A$. Here we are identifying a with the transformation $x \mapsto \delta(x, a)$ that it induces on the state set X. Thus, the elements of S are those transformations that come from some finite sequence of input letters from A.[5] Note that two different words over input alphabet A might yield the same transformation of X. This must in fact happen frequently, since there are only a finite number of transformations of X, as X is finite, while the number of words is infinite. In case each element of S is a permutation of X, then S is a *group* and (X, S) is called a *permutation group*. In that case, it follows that the group contains the inverse of any of its members, and hence the identity permutation.

The problem we actually analyse in detail here involves only a 3×3 board, the 3-queens puzzle. This puzzle consists in finding a possible solution involving 2 queens on a 3×3 chessboard. A solution involves only 2 queens because there are no solutions involving 3 queens on a 3×3 board, but we still refer to it as the 3-queens puzzle for the sake of consistency.

Before we can analyse and discuss the 3-queens puzzle from the point of view of Krohn-Rhodes theory we need to provide a conceptual and to some extent formal introduction to the latter.

[5] Generally, we can include the identity transformation in our semigroups, and therefore such a semigroup will then also be a *monoid*, i.e., a semigroup with an identity element. By convention the empty word (consisting of zero letters) yields the identity transformation, but note that it might not be present as the transformation coming from a word of positive length. The correspondence between automata and semigroups is studied in detail in the Wild Book [15].

2 Overview of Krohn-Rhodes Theory

In this section we provide an intuitive and mostly conceptual overview of Krohn-Rhodes theory and in particular of its holonomy decomposition variant [16]. Formal and much more detailed presentations can be found in a number of sources, e.g. [15,1,9,12,4].

2.1 Conceptual Description

The fundamental idea of Krohn-Rhodes decomposition theory is that a given finite-state automaton can always be broken down into a cascade of elementary or "atomic" machines that depend on each other in a particular way. Since these atomic machines cannot be further decomposed, they serve a purpose similar to the prime factors in the decomposition of an integer. These *irreducible* components are of two kinds: *primes* – those where the semigroup of transformations is a simple[6] group, and *units* – those whose semigroup is contained in the two-state identity-reset *flip-flop*. Both of these act on subsets of the original state set X. Analogously to what happens with the integers, an automaton or transformation can be decomposed into its prime components together with some units. The set of irreducible components of an automaton or transformation semigroup (X, S) is uniquely determined by its semigroup. Moreover, each irreducible component of (X, S) must occur at least once in every decomposition of (X, S). However, unlike what happens with the integers, the decomposition is not unique: some primes contain others,[7] and the multiplicity of occurrence of the various irreducible components in the decompositions can vary. This is one source of the challenges in understanding the theory. In spite of the non-uniqueness, there will always be ways of arranging the prime components and flip-flops in a hierarchical manner that can emulate the behaviour of the original automaton.

Another challenge is understanding the subtleties of how the pieces are put back together to yield this emulating automaton. The way the irreducible components are wired together is 'loop-free', which means that the dependence is unidirectional. This turns out to be a generalization of the carry rule in normal addition. But even if we know the irreducible "building blocks" needed, there may be many ways in which they can be ordered and combined, and re-used in multiple copies . We must wire them in a loop-free manner, setting up particular dependencies from higher level to lower level components, so that the resulting 'cascade product' is able to emulate the original automaton, but it may not be obvious how to construct even one such cascade product.

Given an ordered sequence of irreducible components, the totality of possible such cascade products is contained in their iterated 'wreath product' (to be defined formally below). Iterated here means that one wreaths the first two components, then takes the results and wreaths that with the third components, and so forth until all the components have been accounted for. The result is a truly huge structure, so that from the point of view of engineering applications we are mainly interested in a particular cascade product, i.e. in what can be called a sub-wreath product. Clearly, awareness of the other cascade decomposition possibilities is important from the point of view of design optimization, so we can't just focus on a single cascade and forget about all the others. However, at this early stage understanding one cascade is already challenging enough, so that's exactly what we aim to do.

A few years after the Krohn-Rhodes theorem was proved, Zeiger proved a variant whose statement is somewhat easier to understand: the holonomy theorem [16]. For many years, however, there was no computational implementation of any version of the theorem and only extremely simple or special examples could be solved, by hand. Finally, holonomy decomposition was implemented within the past few years as the SgpDec package [8] in the Gap [10] computational algebra language. Because the holonomy theorem is a bit easier to understand than the original Krohn-Rhodes theorem (although not to prove!), our efforts are currently focused on precisely that: understanding the holonomy theorem and how it can be used. The holonomy theorem says that any finite-state automaton can be decomposed

[6] A group is called 'simple' if it has no normal subgroups. See [2] for more details.
[7] For example, a non-commutative simple group always contains prime-order cyclic subgroups.

into a cascade product of permutation-reset automata. The permutation automata involved are 'sub-machines' of the original automaton whose state sets are subsets of the original state set X and whose semigroups are permutation groups (which include an identity map, of course),[8] and these are augmented with all possible resets to yield the permutation-reset automata of the decomposition.[9]

To explain what a permutation-reset automaton is, Figure 1 shows different types of actions that can be induced on a set of six states by the elements of a semigroup S. Let's call one such element $s \in S$. In Case 1a, s permutes the states and so is a member of a group that is a subgroup of S; in Case 1b, s is a constant map, which is non-invertible; the remaining two cases are other examples of non-invertible transformations. A permutation-reset automaton has elements that can only be like Case 1a (including the identity permutation) or Case 1b. In Cases 1b, 1c, and 1d, s cannot belong to a group action that permutes the states. At the same time, in Case 1b the degree of non-invertibility is maximum. Thus, permutations and resets could be regarded as "mutually orthogonal" in the sense that no combination of one type can yield a member of the other type. In other words, permutation groups and identity-reset semigroups are analogous to a generalized basis into which an automaton can be decomposed.

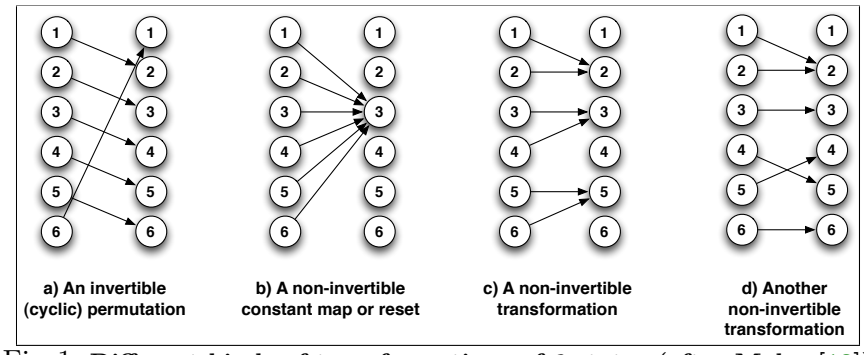


Fig. 1: **Different kinds of transformations of 6 states (after Maler [13])**

Recalling the comment above about the carry rule, the cascade of machines can also be seen from a more abstract point of view as yielding a possible "expansion" of any given state of the original automaton into a positional "number" system. This is best understood by realizing that the notion of a number is actually an abstract concept that has many possible "implementations" or representations: we can choose to count a flock of sheep by an equal number of pebbles, or we can write down a number in positional decimal notation like 89. We could also express 89 in binary notation as 1011001, which is also positional, or in Roman numerals as LXXIX, which is not positional (and therefore much less useful in both practical and mathematical terms). Writing '89' on a piece of paper is easier and more practical than carrying around 89 pebbles, or 89 beans, etc.

However, notice an interesting fact: whereas with numbers we are much more familiar with their (decimal) expansion, to the point that we have to make an effort to notice that '89' is not the "essence" of this number but just one of its possible representations, the opposite is true for an automaton: we are quite familiar with the fact that an automaton is at all times in one of its states, which we can easily visualize, but we have a really hard time thinking about the "expansion" of such a state into a positional notation of some form. This is not just because the base of such an expansion is far from clear and, even if it were clear, in general it changes between positions (levels of the cascade), but because we

[8] Each of these groups arises as a quotient of a group generated by permutations of some subset Z of X, where the generating permutations come from words over the input alphabet that permute the subset Z, or equivalently from the elements of S that permute Z.

[9] As component "building-blocks", permutation-reset automata are *not* irreducible but it is straightforward to refine a cascade of permutation-reset automata further into a full Krohn-Rhodes decomposition of irreducible components by standard methods [15,1,9,4]. For our purposes, holonomy decomposition into a permutation-reset cascade suffices to give a rich enough coordinate system through which to understand a given transformation semigroup (X, S), while avoiding unnecessary detail.

don't really know what to *do* with such an expansion. People generally have no experience in computing in such an expansion, and studying the Krohn-Rhodes or the holonomy theorem is motivated precisely by our desire to develop an intuitive understanding of the positional – or 'coordinatized' – form of the state of an automaton, and also of the action of its semigroup.

Thus, in what follows we will refer to the coordinatized form of a given state as its 'lift'. Analogously, the action of a semigroup element on a given state from the original set X can also be coordinatized or lifted. The lift of the action of a semigroup element can be seen as a generalization of the carry rule that takes the form of a vector of *dependency functions*, to be defined more precisely below. The dependency functions, in turn, are central to the definitions of the cascade product and wreath product. If we switch to a different semigroup generator or input symbol (as one generally does when executing an algorithm), the vector of dependency functions will change, possibly in a radical way. However, the holonomy theorem tells us exactly what these functions are for each semigroup element. Therefore, even if we may not be able to compute state transitions in our head they can certainly be formalized exactly.

The fact that a given automaton can be coordinatized in a certain (range of possible) way(s) and not others means that it has a particular and definite structure. The holonomy decomposition formalizes and reflects this structure in a precise even if very complex way. If the automaton models a particular software application, then what we are saying is that we are exposing the intrinsic static and dynamic structure of the system in a particular way. Although we do not yet know whether holonomy decomposition can have a direct impact on software engineering (cf. [14]), the static and dynamic structure of the system is something that software engineers have been interested in optimizing for a long time, through judicious choice of data structures, classes, methods, and so forth.

Holonomy decomposition gives us the formal and cognitive tools to analyse and understand the global static and dynamic (i.e., behavioural) computational structure of a given finite deterministic automaton. The generality of the method and of the analysis, however, makes this kind of information extremely hard to use; therefore, it is essential that we develop a strong intuitive understanding of the mathematical facts, of the computational possibilities, and of how they relate to different practical examples and software behaviours. This is the main motivation behind this paper.

2.2 Dependency Functions

In this section we reproduce some of the formal concepts presented in [7] which apply to transformation semigroups and permutation groups, because we will need them to analyse the results of the 3-queens puzzle. The holonomy decomposition of a given finite-state automaton yields a *cascade product*, a concept which we shall now describe, that allows us to put together its permutation-reset building-block components. Given a finite automaton, embedded in its transformation semigroup (X, S) as defined above, we seek to decompose it in coordinatized form. A cascade product is a set of machines $(X_1, S_1), (X_2, S_2), ..., (X_n, S_n)$ connected in such a way that the machines that come first (higher levels) influence the state transitions of the machines that come later in this list (lower levels), but not vice versa. The cascade product is itself an automaton or transformation semigroup. Its state set is the Cartesian product of the state sets of the individual machines, whereas the semigroup is given by a rather more complicated structure (shown as a script $\mathcal{C}$ and to be defined more precisely below):

$$(X_1 \times X_2 \times \cdots \times X_n, \mathcal{C}) \tag{1}$$

The basic idea is that states of the higher levels determine which semigroup element will be used at a given lower level. This arrangement is encapsulated in a 'dependency function', which formalizes the notion of carry rule we mentioned above:

$$\begin{aligned} &d_1 : \{\emptyset\} \longrightarrow S_1, \quad \text{i.e.,} \quad d_1 \in S_1 \\ &d_i : X_1 \times X_2 \times \cdots \times X_{i-1} \longrightarrow S_i, \qquad \text{for } \; 1 < i \leq n. \end{aligned} \tag{2}$$

A set of dependency functions for all the levels of the cascade product is called a *transformation cascade*, which should not be confused with the concept of cascade product, and is denoted by $d = (d_1, \ldots, d_n)$. We will need as many such d's as there are generators of the semigroup or, equivalently, as many as there are symbols in the input alphabet of the automaton. Each d will be a *coordinatized form* of an $s \in S$. Each component d_i is a function from the space defined as the Cartesian product of the state sets on the first $i-1$ levels to the semigroup acting on the ith level. Another way to express this concept is that each d_i is a function of $i-1$ parameters, each of which is a state in the $i-1$ levels above level i:

$$s_i = d_i(x_1, x_2, \cdots, x_{i-1}). \tag{3}$$

Notice that in this last equation both the x's and s_i are in lower case, indicating elements x_j of the sets X_j, $j = 1, 2, ..., i-1$, and $s_i \in S_i$. It is important to note that s_i does *not* depend on x_i. A transformation cascade d applied to a state in coordinate form, $(x_1, \ldots, x_n)$, has coordinates $(x'_1, \ldots, x'_n)$ where x'_i is the result of applying transformation s_i to state x_i, a computation in (X_i, S_i). We can think of $(s_1, \ldots, s_n)$ as the coordinatized form of a generator s. But these s_i are not constant, they change according to the higher-level state coordinates $(x_1, \ldots, x_{i-1})$. Thus, we can see that d determines a new function d^* from n-tuples to n-tuples, i.e., a global transformation on the set of n-tuples $X_1 \times X_2 \times \cdots \times X_n$:

$$d^* : X_1 \times X_2 \times \cdots \times X_n \longrightarrow X_1 \times X_2 \times \cdots \times X_n. \tag{4}$$

This function acts on the right of a component state at a given level i as follows:

$$x_i d^* = x_i d_i(x_1, x_2, \cdots, x_{i-1}) = x_i s_i, \tag{5}$$

i.e., $x_i d^*$ is the result in X_i of applying the transformation $d_i(x_1, \ldots, x_{i-1}) = s_i \in S_i$ to state x_i, a computation inside (X_i, S_i). So that globally

$$(x_1, \ldots, x_n)d^* = (x_1 d^*, \ldots, x_n d^*) \tag{6}$$

NB: This shorthand notation hides the dependency on the higher-level states. This mapping d^* determined by the dependency function vector d is called a *cascade transformation* or simply a *cascade*. However, since this is really the function we are interested in, we abuse the notation slightly and drop the star, calling it simply d. Another change to the notation for denoting application of a transformation is to switch from the familiar right multiplication that we just used,

$$y = xs, \tag{7}$$

where x is a state and s is a semigroup element acting on x yielding the new state y, to the less familiar but quite standard exponential notation:

$$y = x^s, \tag{8}$$

which means exactly the same thing. The exponential notation has the advantage of avoiding possible confusion arising from the right vs. left multiplication action, which is rather less straightforward for semigroups than it is for groups. Be that as it may, what we have said until now leads to expressing the action of a dependency function on a state x_i at level i as

$$x_i^d = x_i^{d_i(x_1, x_2, \cdots, x_{i-1})}, \tag{9}$$

whereas globally the transformation d^* acting on a given state $(x_1, \ldots, x_n)$ of the coordinatized automaton is now expressed componentwise as the action of all the component dependency functions d_i in d on all the states x_i of all the machines at the various levels:

$$x^d = (x_1, x_2, \cdots, x_n)^{(d_1, d_2, \cdots, d_n)} = (x_1^{d_1(\emptyset)}, x_2^{d_2(x_1)}, x_3^{d_3(x_1, x_2)}, \cdots, x_n^{d_n(x_1, x_2, \cdots, x_{n-1})}), \tag{10}$$

where $d_1(\emptyset)$ just means that at the top level there are no dependencies on other coordinates, so $d_1(\emptyset) = s_1$, an element of S_1. Or, in the shorthand,

$$x^d = (x_1, x_2, \cdots, x_n)^{(d_1, d_2, \cdots, d_n)} = (x_1^d, \ldots, x_n^d). \tag{11}$$

NB: Again this hides dependency of each x_i^d on $(x_1, \ldots, x_{i-1})$, which is understood.

With these definitions we can now define the *wreath product* as the *Full Cascade Product* (FCP) of transformation semigroups (X_i, S_i), where $1 \leq i \leq n$:

$$FCP = (X_1, S_1) \wr (X_2, S_2) \wr (X_3, S_3) \wr \cdots \wr (X_n, S_n) \tag{12}$$

$$= (X_1 \times X_2 \times \cdots \times X_n, \mathcal{W}), \tag{13}$$

where $\mathcal{W}$ is the set of all possible transformation cascades d. So, finally, we can say that a *Cascade Product* is a sub-transformation semigroup of the FCP of the form

$$(X_1 \times X_2 \times \cdots \times X_n, \mathcal{C}), \tag{14}$$

i.e., $\mathcal{C}$ is some subset of $\mathcal{W}$ closed under composition. Please see [7] for a more exhaustive and complete development and discussion of this formalism.

In constructing decompositions, we will choose the semigroup $\mathcal{C}$, whose elements are transformation cascades, to make the cascade product the coordinatized form for transformation semigroup (X, S) or automaton (X, A, δ). To construct $\mathcal{C}$, we must choose for each state $x \in X$ at least one *lift state* $\hat{x} = (x_1, \ldots, x_n)$ and for each $s \in S$, a *lift cascade* d, given by a dependency function vector $(d_1, \ldots, d_n)$ such that $\hat{x}^d$ will be a lift of state x^s. We must do this so distinct states in X have distinct lifts, i.e. x might have more than one coordinate form $\hat{x}$, but a coordinatized state $(x_1, \ldots, x_n)$ corresponds to at most one $x \in X$. Then it will follow that the cascade product emulates (X, S) since $\hat{x}^d$ will be a lift of x^s. To define $\mathcal{C}$ it suffices to lift generators of S to transformation cascades having this property for all state lifts $\hat{x}$. Thus to decompose an automaton (X, A, δ), after choosing lifts of states, we need only lift the transformations given by input letters $a \in A$. It suffices just to lift generators of S so they satisfy this condition (cf. [15,4]). That is, for $S = \langle A \rangle$, for each input symbol $a \in A$ it is enough to determine at least one lift cascade $\hat{a}$. Then $\mathcal{C}$ can be taken to be the subsemigroup of $\mathcal{W}$ generated by this collection of lifted transformations $\mathcal{C} = \langle \hat{a} : a \in A \rangle$. [10]

If $P \subseteq X$, then $P^s = \{x^s : x \in P\}$ denotes *the image of* P *under* s. In particular, X^s is called the *image set* of s. The holonomy decomposition proceeds by a detailed study of how S acts on inclusion chains of these image sets (of the members of S), and a clever encoding that allows the elimination of much redundancy in the structure due to multiple copies of isomorphic components. For P and Q image sets (or singletons or X), we also write $P \subseteq_S Q$ if $P = Q$ or there is an $s \in S$ with $P \subseteq Q^s$, and we say P is *equivalent* to Q if $P \subseteq_S Q$ and $Q \subseteq_S P$.[11]

We now turn to a more precise formal treatment of the 3-queens puzzle, followed by its holonomy decomposition and representation through lifts of states and transformations. In the final section we discuss the results presented and draw some conclusions of applied computer science relevance.

3 Modelling

Given the state of the art, there is a particular challenge that arises when trying to apply Krohn-Rhodes theory [11] to the 3-queens puzzle: either the decomposition is carried out manually, or we

[10] Furthermore, one can often reduce the state space by only using a proper subset of the full Cartesian product that maps onto the states of original automata and is closed under the action of lifts of transformations. (Examples will be described below.)

[11] This relation $\subseteq_S$ is called the *subduction* relation in the literature. It is relfexive and transitive, and it is crucially used in constructing the skeleton of a transformation semigroup in the holonomy decomposition. In the coordinatization of states, chains of image sets leading to a singleton will be encoded by subduction chains (see below).

rely on the current computational algebra implementation of holonomy decomposition, i.e. SgpDec [8].[12] We choose the latter. In order to work with the holonomy decomposition theorem, we model the 3-queens puzzle as a *transformation semigroup*. To do so, we provide a more formal definition for the *valid configurations* of queens as the set of elements that the semigroup acts on.

Definition 6 (N-Queens States)**.** A *valid configuration* in an N-Queens puzzle is a possible configuration of 0 to N queens on an $N \times N$ chessboard so that no queen is located in the same diagonal, row or column as another queen, i.e. no queen can be attacked by another queen. Valid configurations are also known as *states*. We call X the set of all states.

We specialize to $N = 3$, so X will consist of valid configurations with 0, 1, or 2 queens.

To define the semigroup transformations, well-defined functions on X are required. The action of removing a queen from a state yields a valid configuration while the action of placing a queen may not, so it is necessary to define the action of placing of a queen properly. Before we do that, we need to introduce a way to identify the squares of the board unambiguously. As shown in Figure 2, we adopt the notation used to identify the entries of a matrix, which we call *queen coordinates*; for comparison, the figure also shows the internationally accepted 'algebraic chess notation' (ACN).[13] In ACN each column is identified by a letter, starting at A on the left and increasing towards the right of the board; and each row is identified by a number, with the row numbers increasing upwards. The origin in ACN, therefore, is the bottom-left corner of the board. In queen coordinates notation, entries are similarly identified by ordered pairs but numbers are used for both coordinates and the origin is at the top-left corner of the board. For the case of a 3×3 board, which we analysàďě§e in depth in this paper, however,

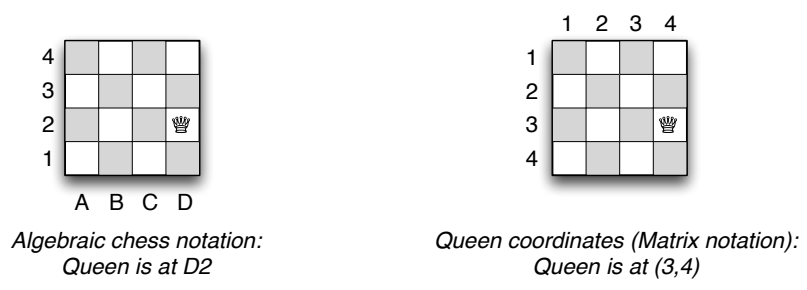


Fig. 2: **Possible coordinate systems for the 4-Queens puzzle**

we follow a *third* convention, which is shown explicitly in Figure 3. Figure 3 shows the 18 states of the 3-queens puzzle. The figure also shows the numbering of the squares of the chessboard. Table 1 shows the various notations we have introduced, along with the one we will use for the 3-queens case in the rest of the paper.

Definition 7 (N-Queens Transformations)**.** The transformation '*Queen at* (a, b)', denoted $Q_{(a,b)}$ according to the queen coordinate notation and Q_i according to the square numbering presented in Table 1, is a function $Q_i : X \to X$, defined for a state x as follows:
If x does not have a queen on square i, the following rules always apply:

1. If placing a queen at position i yields a valid configuration, then $Q_i(x)$ is that valid configuration.
2. If placing a queen at position i does *not* yield a valid configuration, then $Q_i(x)$ is the configuration that would result by removing all the conflicts that arise after placing a queen at i. The valid configuration $Q_i(x)$ should have a queen at position i.

[12] The latter presents technical restrictions, mainly concerning the representation of queen configurations inside the software. Since SgpDec only accepts positive numbers for the elements of the state set, an arbitrary *numbering function* for states (configurations) is used. Using names for states and transformations is possible but still rather cumbersome for users in the current version, requiring mapping to and from such numerical encodings.

[13] http://www.chesscentral.com/Algebraic_Chess_Notation_a/218.htm

Transformations satisfy exactly **one** of the following two rules for placing a queen in an occupied square:

R1: If x already has a queen on square i, then $Q_i(x) = x$.

R2: If x already has a queen on square i, then $Q_i(x)$ is the configuration that results from *only* removing the queen at i.

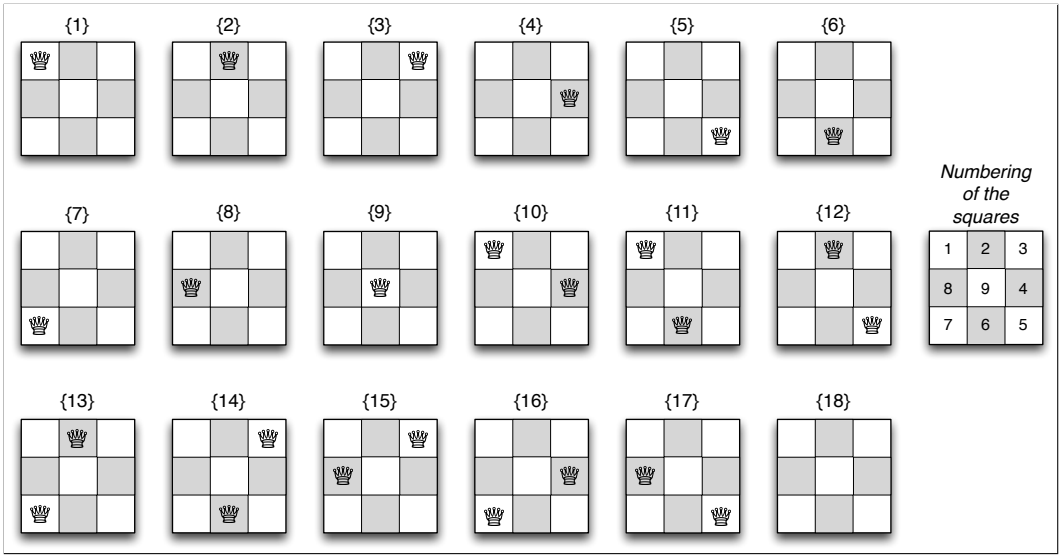


Fig. 3: **States of the 3-queens puzzle**

ACN	Queen Coordinates	Square Number for 3×3 Case
A3	$(1, 1)$	1
B3	$(1, 2)$	2
C3	$(1, 3)$	3
C2	$(2, 3)$	4
C1	$(3, 3)$	5
B1	$(3, 2)$	6
A1	$(3, 1)$	7
A2	$(2, 1)$	8
B2	$(2, 2)$	9

Table 1: **Single-queen coordinates for the 3×3 board**

In our models, either all the Q_i will satisfy rule R1, or all the Q_i will satisfy rule R2. Figure 4 shows an example, using queen coordinates, of what happens when a transformation is applied to a state for the 4×4 board, using rule R2.

Sometimes it helps to look at things from a different perspective. Intuitively, it should be possible for an $N \times N$ board to be rotated, flipped vertically and/or horizontally while preserving the positioning of queens (relative to the other queens). These operations are referred to as *symmetries*. When a symmetry is applied to a state, it always yields a *very similar* state.[14] A symmetry is an invertible transformation that leaves some feature of the structure of a mathematical object invariant. The set of all invertible transformations that leave some structure of a mathematical object invariant always forms a group. Thus, we expect the symmetries of the 3-queens puzzle to act as permutation groups. Formally:

[14] Fortunately, the orientation of the board is irrelevant for the 3-queens puzzle. Also, it does not matter if a queen is placed on a white square or on a black square; otherwise some symmetries could not be applied.

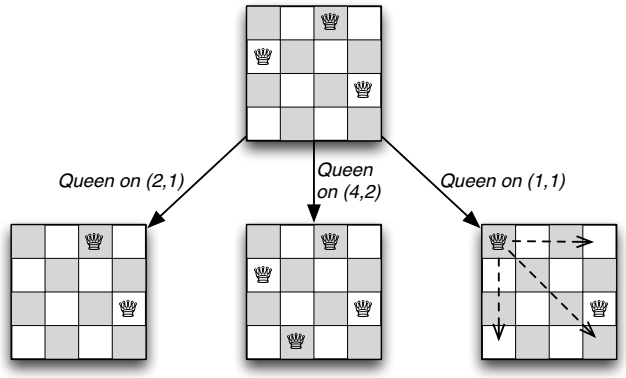


Fig. 4: **Example actions of transformations $Q_{(2,1)}$, $Q_{(4,2)}$ and $Q_{(1,1)}$ following Rule 2 on a state.**

Definition 8 (Symmetries)**.** A *symmetry* is a permutation of the elements of X that preserves structural properties. The function $\rho : X \to X$ is a symmetry if and only if ρ permutes the set X, and for every transformation Q_i there exists a transformation Q_j such that for every $x \in X$ it holds that $Q_j(\rho(x)) = \rho(Q_i(x))$.

Symmetries do not alter the number of queens present in a state. Also, applying a symmetry does not cause a queen to be threatened by another.[15]

For the 3-queens puzzle, symmetries can be divided into two types: rotations and flips; and they permute the state set acting non-trivially as a group on the state set. The intuition behind the definition above is that *any* transformation is a symmetry as long as the structure of the puzzle is *strictly* preserved, except for renaming. Thus, rotations in multiples of 90° are valid, flips about horizontal, vertical or diagonal axes are valid, as is the identity transformation. The reader should notice that the composition of symmetries always yields a symmetry, and a set of symmetries always generates a group. To generate the group of symmetries of the board, we consider the *horizontal flip* H and the *rotation* R:

- The symmetry H flips the board around the horizontal axis that goes through the middle of the board i.e. a line that goes through the central row, leaving it unchanged.[16] Note $H^2 = 1$, the identity transformation. H is symmetry since we have, for all configurations $x \in X$, that $Q_{8-i}(H(x)) = H(Q_i(x))$ for $1 \leq i < 8$, $Q_8(H(x)) = H(Q_8(x))$, and $Q_9(H(x)) = H(Q_9(x))$.
- The symmetry R rotates the board 90° clockwise while fixing the central square. In the very particular case of the 3-queens puzzle, there is an additional special symmetry not occurring for larger board sizes. We call it the *cyclic shift*, with symbol CS. CS rotates the squares of the board by 45° clockwise, simulating an effect on the outer ring of squares similar to a conveyor belt. Figure 6 shows graphical representations of the action of H and CS on the state set. We have for all configurations x that $Q_2(CS(x)) = CS(Q_1(x))$ and similarly $Q_{i+1}(CS(x)) = CS(Q_i(x))$ for $1 \leq i < 8$, and $Q_1(CS(x)) = CS(Q_8(x))$, and of course $Q_9(CS(x)) = CS(Q_9(x))$. Therefore CS is a symmetry. Note that these considerations all hold whether our model is following Rule 1 or following Rule 2. We also have $R = CS^2$, and $CS^8 = R^4 = 1$.

Contrary to Q_i transformations, applying the *same symmetry several times* (say ρ) maps the state set surjectively onto itself, and every symmetry is reversible; it is possible to come up with a sequence of m repetitions of ρ that reverses ρ, i.e. $\rho\rho^m = 1$. For example, to reverse a 45° rotation (named CS) it suffices to rotate the board 7 more times by 45° (see Figure 6). Since ρ is a permutation of the finite set X, ρ^k is the identity transformation for some $k > 1$. We write ρ^{-1} for ρ^{k-1}, which is the *inverse* of ρ, i.e., the permutation undoing ρ, since then $\rho^{-1}\rho = \rho\rho^{-1} = \rho^k$ is the identity.

The concept of symmetries is presented in more detail in the following sections, where we also present different ways to model the 3-queens puzzle as a graph search. These models will be studied and

[15] If it was not threatened already, which is never the case for valid configurations.

[16] This description only applies for an N-Queens puzzle when N is odd, which is our current emphasis since we are analysing the $N = 3$ case. When N is even no square will be left invariant under H.

discussed in detail in Section 5. Some of the transformations we have defined are invertible, whereas others are not. Therefore, the set of all transformations that can act on an N-Queens puzzle is in general a semigroup. As mentioned before, just as an abstract group acting on a set of points faithfully is called a permutation group, an abstract semigroup acting faithfully on a set of states is called a transformation semigroup.

In the following sections, we present different models of the 3-queens puzzle as automata. These four cases share the same state set, but the differences between them are given by slightly changing the transitions between states, using either rule R1 or R2 , and by either including symmetries or not in the semigroup. In each case, we highlight different properties of the model and the expected features that are to be discovered through holonomy decomposition, which will then be discussed in more detail in Section 5.

3.1 Case 1: 3-Queens, R1, No Symmetries

For this case, transformations follow rule R1, so a transformation that places a queen over an existing queen does not affect the current state. Additionally, no symmetries are considered. The motivation behind this case is the following: suppose we are trying to solve an instance of the 3-queens puzzle, and we apply the action of placing a queen on a particular square, e.g. Q_1; after the queen is placed on Square 1, placing another queen on Square 1 will cause no impact, as there is already a queen on that square and no conflicts with other queens will arise. Since this statement can be generalized to all nine Q_i transformations, we can state that applying Q_i after Q_i has *exactly the same impact* as just applying Q_i once, i.e. $Q_i^2 = Q_i$. Therefore, each transformation Q_i is *idempotent*,[17] which is the property that we want to highlight in Case 1.

Figure 5 gives a visual interpretation of the impact that each Q_i transformation has on the board when it is applied, in the form of a finite-state automaton whose transition arcs are labelled with the different possible transitions. There are so many transformations (equivalently, semigroup generators or input symbols) acting on the 18 states of the 3-queens puzzle that in the interest of readability the figure is broken down into 9 sub-figures corresponding to the different actions of the 9 generators. The reader may notice that there is an apparent pattern in 8 of the transformations. This pattern will be studied in detail in Case 2. The different colours in the first 8 cases refer to the starting state when any of the Q_1-Q_8 transformations is applied: if the starting state has 1 queen or less, the arrow is red; if the starting state has 2 queens, the arrow is blue.

There are exactly 41 distinct transformations of X arising from finite sequences of generators (42 if we include the identity). All generator transformations are idempotent, and moreover, it happens that any word in the generators also gives an idempotent transformation. It follows that no non-trivial permutation groups are embedded in the semigroup $S = \langle Q_1, \ldots, Q_9 \rangle$àdĹ. This means that there is no subset of states that is non-trivially permuted under any combination of the Q_i transformations. Therefore, the only components of the cascade product obtained through holonomy decomposition are *identity-resets*. We study the role of these identity-resets in detail in Section 5.1.

Computational exploration for $N = 3, 4$, or 5 queens shows that $s^2 = s$ for every element of the semigroup in this case, which leads us to suspect it is true for all N.

[17] An element e of a semigroup S is *idempotent* if $e^2 = e$. Note that while a group has exactly one idempotent (the identity element), semigroups can have many.

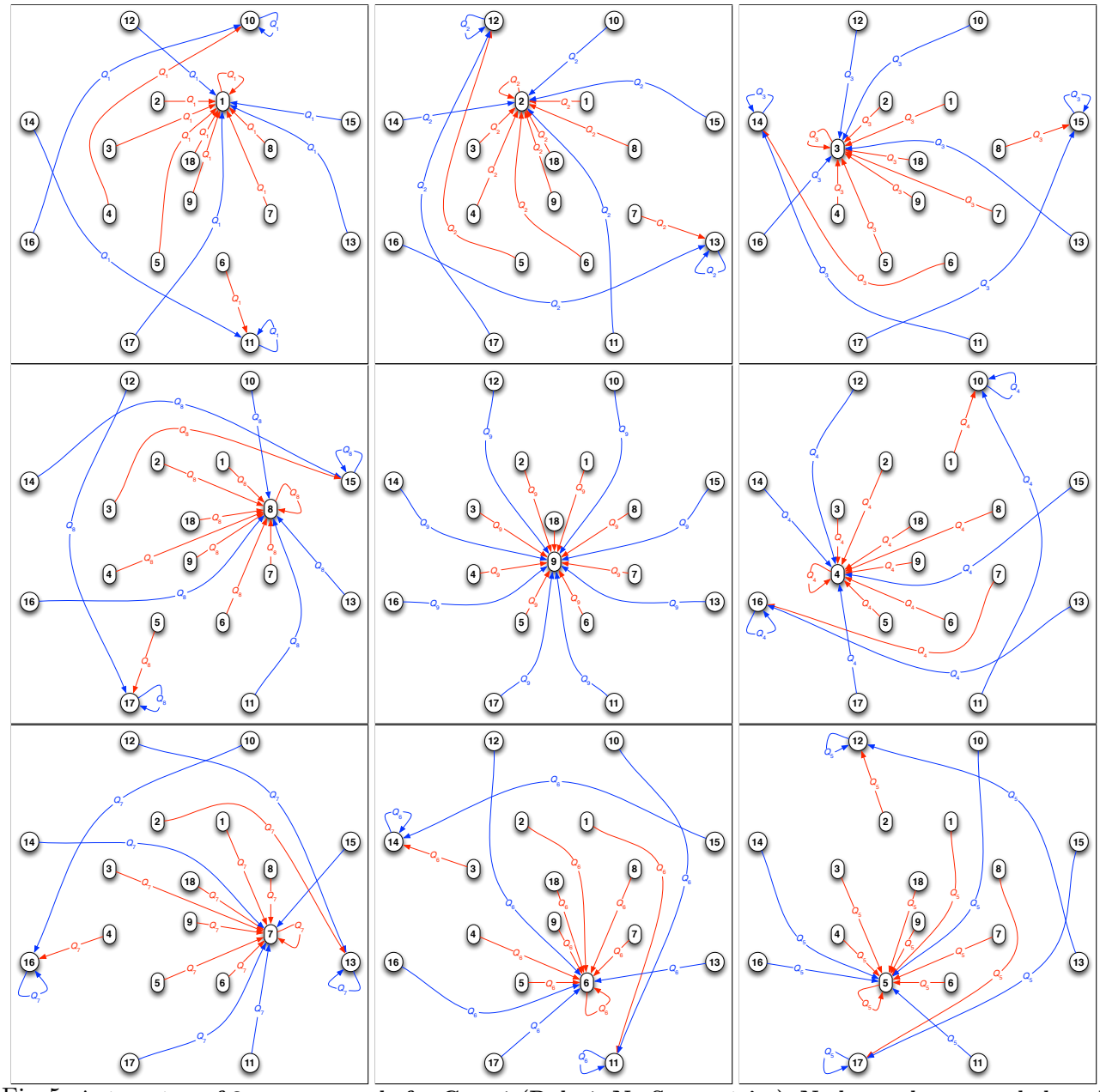


Fig. 5: **Automaton of 3-queens puzzle for Case 1 (Rule 1, No Symmetries). Node numbers encode board configurations (states) according to the enumeration in Figure 3. The 9 different views correspond to the actions of the 9 generators Q_1 through Q_9 (in order from to top-left, reading row-by-row) of the semigroup for this case. Arrows indicate state-transtions due to action of the generator (input symbol) and are labeled by the generator.**

3.2 Case 2: 3-Queens, R1, with Symmetries

As the reader may have noticed, the Q_i transitions (except Q_9) in Figure 5 display a visual pattern. Through careful examination of transformations Q_1 and Q_2, it can be observed that Q_2 is *almost identical* to transformation Q_1, i.e. it is possible to define Q_2 in terms of Q_1 by relabelling according to the inverse of the cyclic shift, applying Q_1, and shifting back. That is, if we rotate by board 45° counterclockwise, apply Q_1 to configuration x, and then rotate back by 45° we obtain $Q_2(x)$. i.e., example, $Q_2(x) = CS(Q_1(CS^{-1}(x)))$. That is $Q_2 = CS^{-1}Q_1CS$, which actually follows immediately from $CSQ_2 = Q_1CS$ as we showed in verifying that CS is a symmetry. From our considerations showing CS is a symmetry above, similarly $Q_{1+k} = CS^{-k}Q_1CS$ for $0 \leq k \leq 7$, and $Q_9(CS(x)) = CS(Q_9(x))$. So in fact, it is possible to express any transformation Q_i (except Q_9) *in terms of* Q_1 and CS. As a consequence, all transformations for Case 2 can be defined in terms of only 4 generators Q_1,

Q_9, H, and CS. Figure 6 shows one such set of 4 generators. We call a minimal set of generators for the semigroup of a given automaton, as in this case as described, a *minimal generating set*.

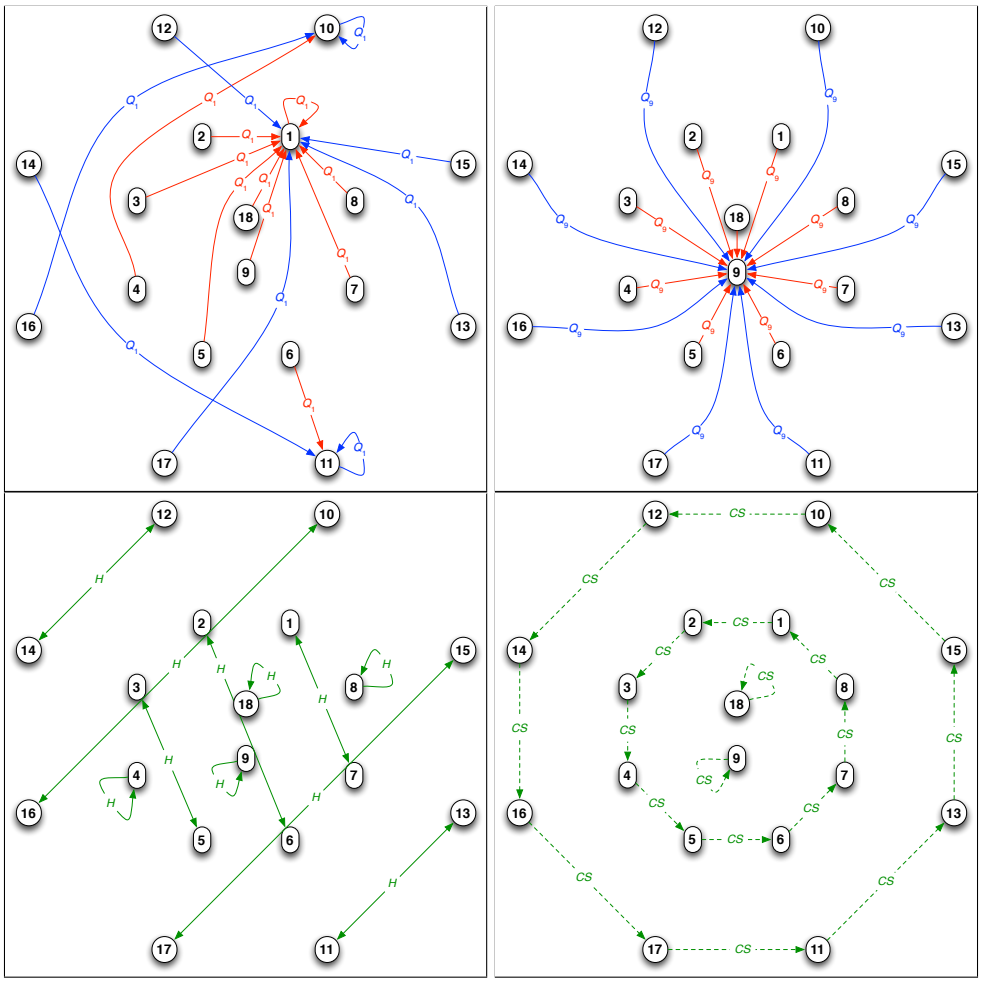


Fig. 6: **Generators of the Automaton of 3-queens puzzle for Case 2 (Rule 1 with Symmetries). Node numbers encode board configurations (states) according to the enumeration in Figure 3. The 4 different views correspond to the actions of a minimal set of 4 generators of the semigroup for this case, in order from top-left Q_1, Q_9, and on the second row H and CS. Arrows indicate state-transtions and are labelled by the generator (input symbol).**

Including symmetries in the semigroup means that non-trivial *permutation-reset groups* will be found through holonomy decomposition. In this case it will also to give rise to subgroups that permute certain proper subsets of the configuration space. We study the meaning of these permutation-reset groups in detail in Section 5.1.

3.3 Case 3: 3-Queens, R2, No Symmetries

As a variation of Case 1, the Q_i transformations of Case 3 follow rule R2 instead of rule R1. This means that we are now allowed the removal of an individual queen located at position i from the board by applying transformation Q_i. Naturally, repeating the process of applying the same Q_i over and over always changes the state of the board as it places or removes at least one queen. It is important to notice that the first time the transformation Q_i is applied, if there are any conflicting queens and they are removed, they are *not* placed back on the board if the transformation Q_i is applied again. This shows that Q_i can sometimes have the some result when applied to two different states. Thus Q_1 is non-invertible as one cannot uniquely recover the previous configuration. Thereafter, further repetitions of Q_i toggle back-and-forth by removing and placing one queen, and no other pieces are affected (any that would have been were removed at the first Q_i's occurrence).

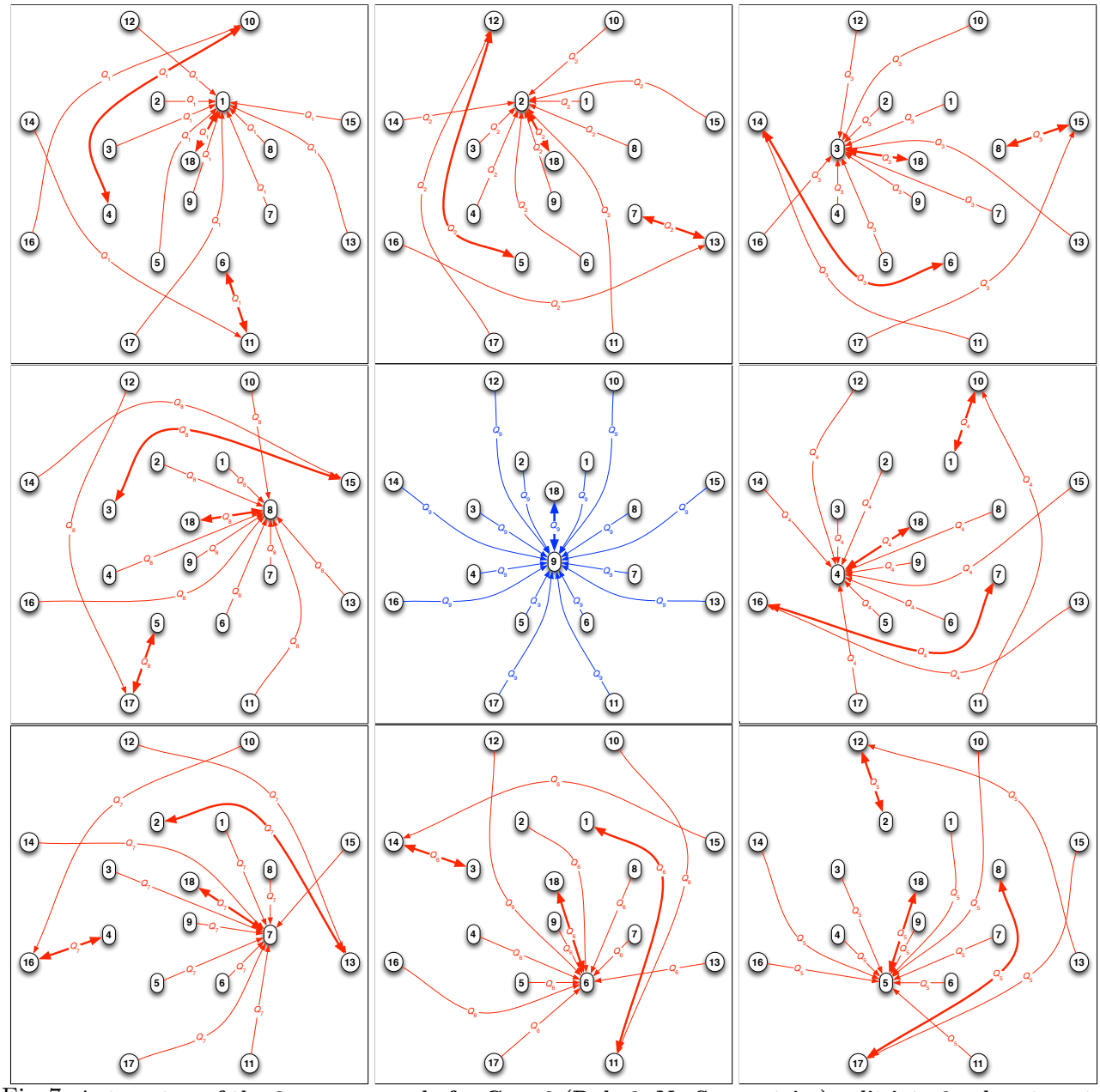


Fig. 7: **Automaton of the 3-queens puzzle for Case 3 (Rule 2, No Symmetries) split into 9 sub-automata corresponding to the 9 generators of the semigroup (9 symbols of the input alphabet Q_1 to Q_9 ordered right-to-left row-by-row starting from top-left). Node numbers encode board configurations (states) according to the enumeration in Figure 3. Arrows show state transtions labelled by the generator (input symbol). The thicker bi-directional arrows are meant to highlight the toggle behaviour caused by R2, which gives rise to cyclic 2-element permutation groups $C_2 = \langle Q_i^2, Q_i^3 \rangle$, each acting on a subset with 6 states (in the case of Q_1 through Q_8) or 2 states (for Q_9), respectively, by "toggling" (transposing pairs of states).**

Here Q_i is clearly not idempotent, but instead we can conclude that $Q_i = Q_i^3$ (and hence $Q_i^2 = Q_i^4$, etc.), and that Q_i does permutes the proper subset of configurations reached when the most recent move was an application of Q_1. This causes a very particular behaviour that is observed in the "skeleton" of the holonomy decomposition:[18] Each of the Q_i transformations gives rise to different permutation groups on proper subsets of the configuration space, but not over the whole set of configurations. These group actions are represented in Figure 7 by means of thicker double-headed arrows, and each Q_i a group acts on at least two states. Because the thicker arrows map to and from 1-queen and 2-queen states, in this figure the same colour was used for all the states in each sub-figure. After holonomy decomposition, these groups will be represented by 9 non-trivial permutation-reset groups that act on proper subsets of the state set (see Section 5.1).

[18] The skeleton is discussed in detail along with several examples in Section 5.

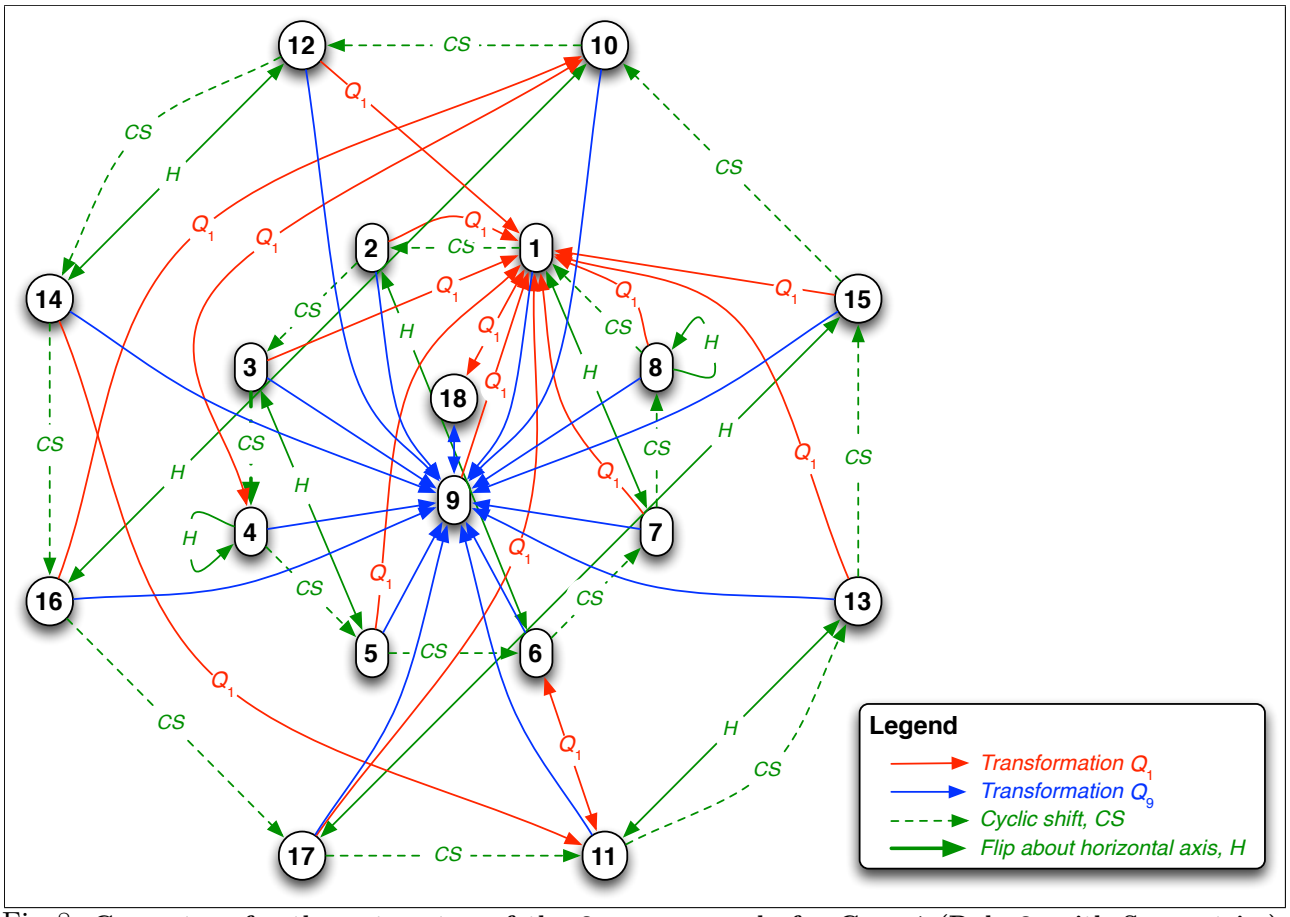


Fig. 8: **Generators for the automaton of the 3-queens puzzle for Case 4 (Rule 2, with Symmetries). Only one of the 8 possible minimal sets of semigroup generators is shown. Node numbers encode board configurations (states) according to the enumeration in Figure 2. Arrows show state transtions labelled by the generator (input symbol). H and CS leave States 9 and 18 unchanged (not drawn); the actions of Q_1 and Q_9 are identical to those in Figure 7; and the actions of H and CS are identical to those in Figure 6.**

3.4 Case 4: 3-Queens, R2, with Symmetries

Case 4 introduces symmetries to Case 3 in a similar way Case 2 introduced them to Case 1. Similarly to Case 2, just 4 generators are sufficient. Again, a pattern can be observed among some Q_i transformations (see Figure 8), meaning that the Q_i transformations can be described in terms of Q_1 and the symmetry CS. Intuitively, by expressing Q_i transformations in terms of other Q_i transformations via symmetries e.g. $Q_2 = CS^{-1}Q_1CS$, the structures associated with Q_1 are also preserved by the symmetries H and CS. Therefore, we expect to see 1-to-1 mappings between the subgroups that were induced by the Q_i transformations of Case 3, and this will be reflected in the skeleton diagram. We provide more details for this case in Section 5.1.

3.5 Further Visualization

Figure 9 shows a summary of the action of all the transformations on all the states of the 3-queens puzzle, for the 4 cases. Figures 10 and 11 show similar information as directed and undirected graphs. The solid undirected links between states shown in the graphs indicate that there are transformations – which can be read off from Figure 9 – that map the two states to each other. The dotted, directed links, on the other hand, indicate that only 1-directional transformations exist between the states they join. The regular patterns shown highlight the internal symmetry of this problem.

Cases 1 & 2	Transformations											
States	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	H	CS
18	18	1	2	3	4	5	6	7	8	9	18	18
1	1	1	2	3	**10**	5	**11**	7	8	9	7	2
2	2	1	2	3	4	**12**	6	**13**	8	9	6	3
3	3	1	2	3	4	5	**14**	7	**15**	9	5	4
4	4	**10**	2	3	4	5	6	**16**	8	9	4	5
5	5	1	**12**	3	4	5	6	7	**17**	9	3	6
6	6	**11**	2	**14**	4	5	6	7	8	9	2	7
7	7	1	**13**	3	**16**	5	6	7	8	9	1	8
8	8	1	2	**15**	4	**17**	6	7	8	9	8	1
9	9	1	2	3	4	5	6	7	8	9	9	9
10	10	10	2	3	10	5	11	16	8	9	16	12
11	11	11	2	14	10	5	11	7	8	9	13	13
12	12	1	12	3	4	12	6	13	17	9	14	14
13	13	1	13	3	16	12	6	13	8	9	11	15
14	14	11	2	14	4	5	14	7	15	9	12	16
15	15	1	2	15	4	17	14	7	15	9	17	10
16	16	10	13	3	16	5	6	16	8	9	10	17
17	17	1	12	15	4	17	6	7	17	9	15	11

Cases 3 & 4	Transformations											
States	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	H	CS
18	18	1	2	3	4	5	6	7	8	9	18	18
1	1	18	2	3	**10**	5	**11**	7	8	9	7	2
2	2	1	18	3	4	**12**	6	**13**	8	9	6	3
3	3	1	2	18	4	5	**14**	7	**15**	9	5	4
4	4	**10**	2	3	18	5	6	**16**	8	9	4	5
5	5	1	**12**	3	4	18	6	7	**17**	9	3	6
6	6	**11**	2	**14**	4	5	18	7	8	9	2	7
7	7	1	**13**	3	**16**	5	6	18	8	9	1	8
8	8	1	2	**15**	4	**17**	6	7	18	9	8	1
9	9	1	2	3	4	5	6	7	8	18	9	9
10	10	4	2	3	1	5	11	16	8	9	16	12
11	11	6	2	14	10	5	1	7	8	9	13	13
12	12	1	5	3	4	2	6	13	17	9	14	14
13	13	1	7	3	16	12	6	2	8	9	11	15
14	14	11	2	6	4	5	3	7	15	9	12	16
15	15	1	2	8	4	17	14	7	3	9	17	10
16	16	10	13	3	7	5	6	4	8	9	10	17
17	17	1	12	15	4	8	6	7	5	9	15	11

Legend

H = Flip about horizontal axis
CS = Cyclic shift
Q_i = Queen placed on square i, i = {1..9}
Q_0 = No action (identity)

Fig. 9: **Transformations corresponding to all 4 cases for the 3-queens puzzle. The cyclic shift moves the outer 8 squares and any queens they may contain according to the cyclic permutation (1 2 3 4 5 6 7 8).**

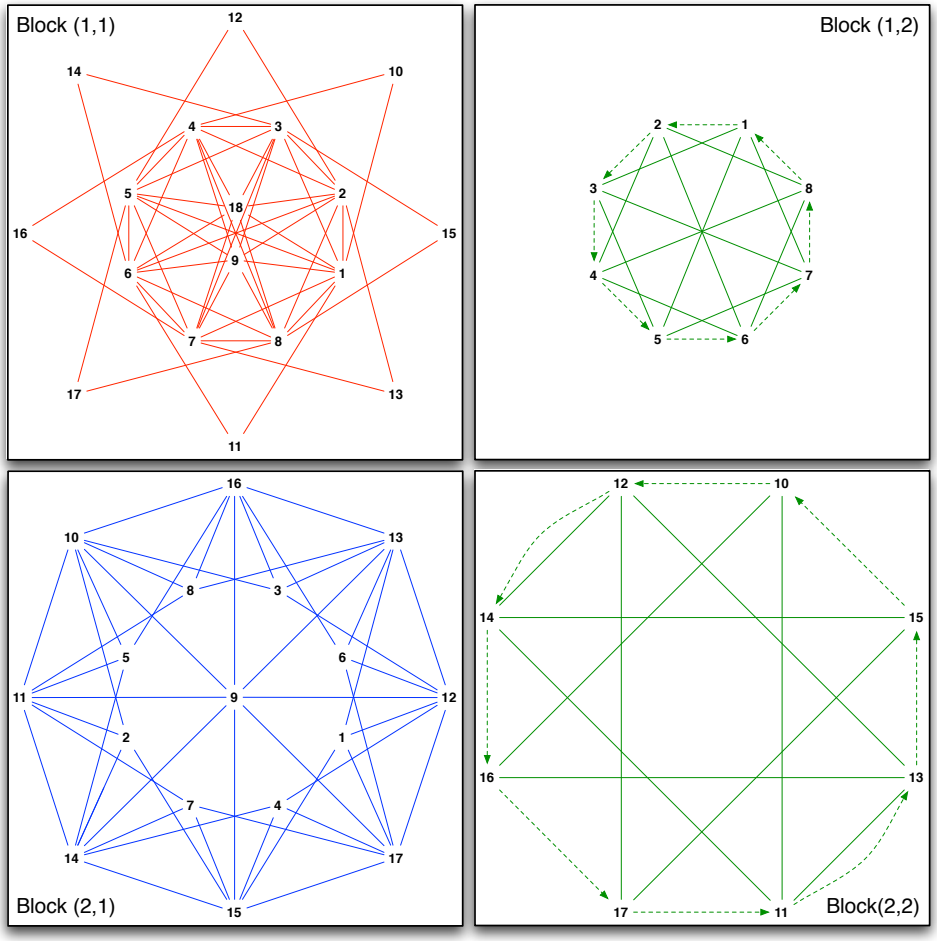


Fig. 10: **Graphical visualization of the Q_i and symmetry transformations shown in Figure 9 for Cases 1 and 2. Unlike that table, Blocks (1,2) and (2,2) show all the symmetries.**

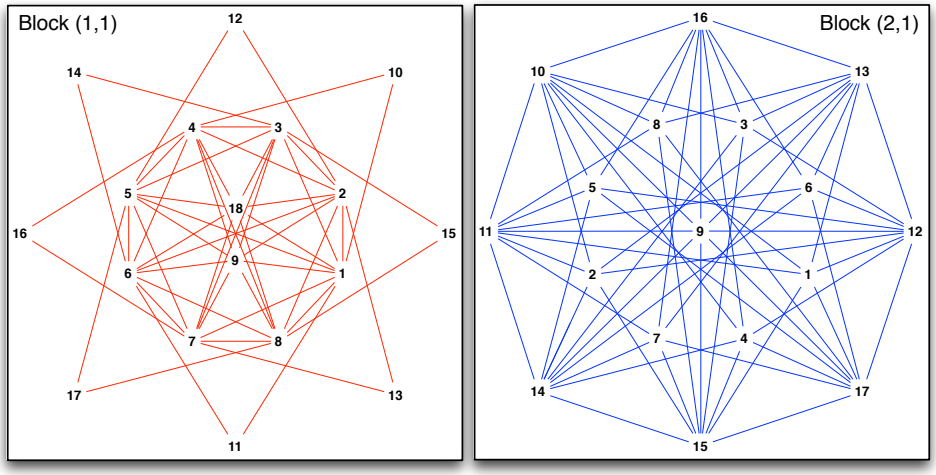


Fig. 11: **Graphical visualization of the Q_i transformations shown in Figure 9 for Cases 3 and 4. The symmetries are the same as shown in Figure 10.**

4 Structural Features of Holonomy Decomposition

Krohn-Rhodes theory states that there is always at least one cascade product of building-block components that emulates the automaton that is decomposed (henceforth referred to as 'original automaton'). In the case of holonomy decomposition, these building-block components are of the permutation-reset or identity-reset kinds.[19] It is important to remember that non-trivial permutation groups are present in the decomposition if and only if there is a non-trivial group of transformations acting on a subset of states of the original automaton. Otherwise, only resets will be observed.

4.1 Skeletons, Lifted States and Transformation Trees

In the rest of the paper we discuss the algebraic structure of the four automata we have set up for the four cases just presented through two different kinds of diagrams: the *skeleton*, giving information needed in the global decomposition of the original automaton, and, for each input symbol (or word) of the original automaton, a *transformation tree*, describing how the corresponding semigroup element acts in coordinate form. These are visualizations which are generated by SgpDec. States in the original automaton correspond to one or more *lifted states* in the coordinate system, which are given by a vector of coordinate values. Similarly, input symbols or generators (and hence more general transformations) in the original automaton are also lifted to the cascade product of building blocks. *Lifted transformations* are given by a vector of dependency functions,[20] as we have seen, and these are visualized as transformation trees. These are the *coordinatized form* of the original generators (and transformations), but they have a tree form.[21] Both kinds of diagram were developed over the course of many years, and to some extent are still in development. Each kind of diagram is an attempt at capturing the cascaded structure or 'positional number system expansion' coming from the holonomy decomposition. The skeleton allows one to see what building blocks to be cascaded are needed, and also what coordinate values are permitted in lifted states and transformations, whereas a transformation

[19] As mentioned above, Krohn-Rhodes Theory can be used to further decompose these into irreducible building blocks, namely, *primes* (whose action semigoups are finite simple groups) and *units* (where the semigroup acting is a subsemigroup of the *flip-flop*, which consists of two constant maps and an identity map. However, for purposes of improving understanding, it is already very productive to decompose to the permutation-reset level.

[20] As explained in Section 2.2, such a vector of dependency functions is called a *cascade* in SgpDec.

[21] Note that determining lifts for generators (corresponding to input symbols) also yields lifts for all words, so it is sufficient to lift every input symbol s to a coordinatized cascade $s^{\uparrow}$ (i.e., a vector of dependency functions, which can be visualized as a transformation tree), so that for every lift $x^{\uparrow} = (x_1, \ldots, x_n)$ of a given state x, i.e., x in coordinate form (with x_i a state of the ith building-block component), one has that $x^{\uparrow} \cdot s^{\uparrow}$ is a lift of $x \cdot s$ (possibly among many). With the requirement that distinct states do not share any common lifts, it can be shown that lifting input symbols determines lifts of all semigroup elements of the original automaton.

tree gives a complete description of how to lift a particular transformation in coordinate form. These diagrams require care to understand as they not only present different views of structures needed to pick out the holonomy decomposition of the automaton being analysed (as a generally much smaller substructure of a particular iterated wreath product or Full Cascade Product), but they also largely show *different information*: one a global layout useful in seeing what building-blocks are needed, and the other the coordinatized form for particular generators (of which there will generally be several).

Before we can list the properties of these two diagrams, we need to introduce one more level of conceptual (and formal) complexity that we have so far avoided: the states of the permutation-reset automata at the various levels of the holonomy decomposition that we have been talking about are not the singleton states of the original automaton, except for the very lowest level. At all the higher levels the states are actually *subsets* of the states of the automaton. In computer science these are often called *macrostates*, whereas we tend to call them *tiles*.[22] A *subset chain* will be a chain of image sets under inclusion $X = T_0 \supset T_1 \supset \cdots \supset T_{n-1} \supset T_n = \{x\}$, where x is a (singleton) state and each T_i, for $0 < i < n$, is an 'image macrostate' (i.e., a subset of the full state set X with $T_i = X^s$ for some semigroup element s). Clearly the semigroup of the automaton acts on these subset chains, and this action is enough to reconstruct the original automaton (just look at the singletons within the chains). If we additionally require that each T_i is a tile of T_{i-1} (for $1 \leq i < n$) and the chains are maximal, then this almost works as an action too, except for some collapsing. Holonomy can recover how the semigroup maps these tile chains much more efficiently. Speaking intuitively and loosely, the clever idea behind the holonomy decomposition is an efficient encoding of this "action" of the semigroup on tile chains into a wreath product whose component automata can be selected using representatives of what is going on at each level.

The structural implications are significant, since this means that the sub-automata at each level are greatly constrained in their constituent parts. In addition, this also means that the sub-automata at the different levels – as one moves down the hierarchy – can also be interpreted as increasingly "higher-resolution" versions of the original automaton. In the positional number system analogy, we might be tempted to identify this peculiar structure with the increasing resolution provided by the hundreds, tens, and units in a number like 789. As a matter of fact, the opposite direction applies, as shown in Figure 12 and explained in detail in the next section.[23]

4.2 Motivating Example: Decimal Expansion as a Holonomy Decomposition

In the discussion that follows we will give several examples of how this decomposition plays out in the 3-queens puzzle, yielding a coordinate system. This is analogous to how we put coordinates on the integers, using the famous decimal expansion. For example, the decimal expansion can be seen as holonomy coordinates, as follows – Figure 12 shows the skeleton.

> This construction is based on the method of [6] yielding the base b expansions. Taking base $b = 10$, let our state set be Z, the numbers 0 to 999 (modulo 1000). Take two input symbols as generators: $+1 : n \mapsto n + 1$ and $\times 10 : n \mapsto 10n$ (modulo 1000). These generate the transformation semigroup acting on Z. Consider the image sets that can be obtained by applying words in these generators, i.e. sets of the form Z^s for some s in the semigroup, depicted in Figure 12. Image sets are shown in white ovals or white rectangles. Arrows show inclusion relations (and also image-of relations). The image sets containing 0 are of the form $10^k Z$ (actually these are subgroups of the integers modulo 1000 under addition). These subgroups are on the leftmost path in the figure, shown in white rectangles to indicate that we chose them as representatives at their level. All other image sets are shown inside ovals. Then the 'cosets' $10^k Z + i$ of the representatives happen to be disjoint image sets at each level, and are all 'holonomy-equivalent' at each level, i.e., they can be mutually reached by elements

[22] Formally, we only speak of tiles with respect to a larger subset Y of the state space. A tile T of Y will be a proper subset of the particular subset Y such that if $T \subseteq Z \subseteq Y$ for any *image set* Z (i.e. a set appearing in the skeleton), then Z is either T or Y. We then say that T is a *tile* of Y.

[23] The algebraic notion of successive refinement discussed here is a result of the containment relations between equivalence classes (cosets) of integers (or real numbers): Although the representative elements of the equivalence classes are getting larger as we move to the left (10s, 100s, 1000s, etc), each equivalence class is a proper subset of the subgroup corresponding to the position on its right, e.g. for the number 789 its decimal form corresponds to $1000\mathbb{Z} + 700 \subset 100\mathbb{Z}$, $100\mathbb{Z} + 80 \subset 10\mathbb{Z}$, $10\mathbb{Z} + 9 \subset \mathbb{Z}$, etc.

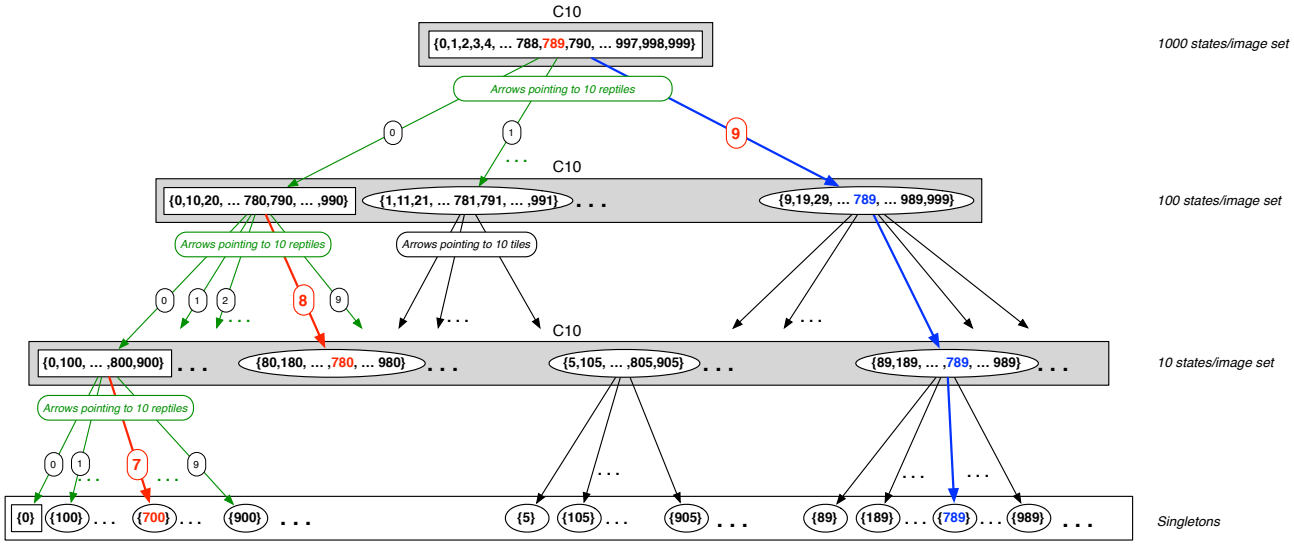


Fig. 12: **Decimal Expansion as Holonomy: Visualization of a possible holonomy decomposition skeleton for the integers (mod 1000), showing how "789", arises as the coordinate form of the number 789. Note numbers have no inherent base, neither base 10 nor base 2. An automaton leading to base 10 coordinatization has states $\{0, \ldots, 999\}$ and two generating inputs: corresponding to addition of $+1$, and multiplication by 10 (modulo 1000). Image sets form inclusion chains, and chains up from singletons correspond to numbers. These chains are then encoded in coordinate form as representative tile chains (realized as a vector of digits – see text). (This example is constructed following the method of Egri-Nagy and Nehaniv [6].)**

of the semigroup (namely, repeated addition of $+1$), so they form an equivalence class. An image set contained immediately in a representative is called a 'tile'. The tiles of a representative are called '*rep-tiles*', or **reptiles** for short. So we are taking the following representatives at the three levels above the singleton level:

- 1000-state image set, the full state set: $\{0, 1, 2, \ldots, 999\}$
- 100-state image set: $\{0, 10, ..., 990\}$ (the leftmost one)
- 10-state image set : $\{0, 100, .., 900\}$

Each image set (except singletons) has ten tiles of which it is the union. In particular, each of these 3 representatives has 10 such tiles, pointed to by green arrows in Figure 12. These reptiles are the permitted coordinate values for the corresponding levels in the decimal expansion. Thus, the holonomy coordinate form of the number 789 will be the string of digits 789. The inclusion tile chain $\{789\} \subset \{89, ..., 989\} \subset \{9, 19, ..., 999\}$ is encoded as the 'subduction' reptile chain $\{700\} \subset_S \{80, 180, ..., 980\} \subset_S \{9, 19, ..., 999\}$, which is encoded compactly as the digits 7, 8, 9 by assigning to each reptile a digit between 0 and 9 in the natural way. To see how to get the subduction chain from the inclusion tile-chain for 789, we go top-down. In the holomony coordinatization of states, the ith coordinate value is always given by a (rep)tile of a level i equivalence class representative. It is important to note when learning to understand the holonomy decomposition that the possible coordinate values at the ith position correspond to the reptiles of a representative at level i. In general, these reptiles might occur at various different levels below the representative (although in this example they are all at the level immediately below the representative); keeping this in mind helps understand the following construction:

- To make things work out in a familiar form in this example, we order the 10 reptiles at each level from zeroth to ninth, in order of their least element. Thus we count the reptile containing zero as the *zeroth* tile as illustrated in Figure 12.
- The top reptile is the same as tile $\{9, 19, ..., 999\}$ of the full state set. The full state set is unique and contains all image sets, so nothing else is equivalent to it. It is therefore its own representative; therefore, the tiles of Level 1 (which live on Level 2) are necessarily and always all reptiles. Thus, the top reptile of the decimal expansion of 789 by holonomy decomposition is the 9th tile, so the first coordinate is 9.
- To get the correct reptile for the second coordinate, first we need to move our chain to the equivalence class representative at Level 2, i.e., $\{0, 10, 20, .., 990\}$, so subtract 9 (i.e., add 991, applying $+1$ that many times). This moves tile $\{89, 189, ..., 989\}$ to $\{80, 180, ., 980\}$, the 8th tile of the representative, labelled 8. So the second coordinate value is 8.
- Now we are at $\{780\}$ at the bottom of our chain (having also subtracted 9 all they way down our chain), but we need to go to a tile of the Level 3 representative $\{0, 100, .., 900\}$, so subtract 80 (or add 920); this gives $\{700\}$, the 7th tile. Thus, the last coordinate value is 7.

So writing holonomy coordinates top-down yields (9,8,7) or, putting top coordinate rightmost, (7,8,9), yielding the decimal expansion. This works to give the decimal-expansion, coordinate form for any number $z \in Z$. Similar constructions yield the base b expansions of the integers.

Thus, loosely speaking, the skeleton diagram can be used to read off the structure of the expanded representation of a given state. Soon we will see that a transformation tree diagram can be used in the same way to read off the expanded version of a given transformation – which we called a transformation cascade in Section 2.2. In each case, such expanded representation is called a 'lift'. Note, therefore, that the skeleton gives indications of the collection of all possible state lifts, whereas, by contrast, a transformation tree specifies a lift for just *one* generator. Further, the skeleton and the transformation tree emphasize different kinds of maps. The skeleton emphasizes vertical maps *between levels*, i.e. downwards maps from subsets of states (macrostates, tiles) at higher levels to their image sets at lower levels, often shown by annotating an arrow with a word in the generators that maps its source onto its target. The maps acting *at any one level* in the skeleton are not shown explicitly, except for an indication of the presence of equivalence classes to which different macrostates (tiles) may belong. However, a group of permutations acting on the tiles of a representative is shown in the skeleton as a symbol for the group labelling the equivalence class – e.g. the cyclic group on ten elements C_{10} as indicated in Figure 12 – unless the group is trivial (i.e. composed of just the identity element).

Transformation trees (not shown for the two generators in this example, but see other examples below), on the other hand, say nothing at all about vertical maps, they only display the dependency functions and the horizontal maps (permutations, resets, and identities) that act on the reptiles at each level. The action on tiles is always encoded by a member of a group permuting the reptiles (like C_{10} in the decimal example) or by resets. A special case of such a permutation is of course the identity map. Thus the holomony decomposition is into a cascade product of permutation-reset and identity-reset transformation semigroups. At a given level in the transformation tree, the number of downward branches that open out from a given node is equal to the number of tiles of representatives present at that same level, with each node decorated by the group element or reset that acts on the tiles of the representative corresponding to that node. Non-identity permutations are usually given inside the node in the standard orbit-cycle notation of group theory. Identity transformations are shown as small empty circles. The node for a reset transformation gives the constant value taken by the reset, indicating a particular fixed reptile (usually by its numerical index).

Although this makes it easier to display structural information graphically, the shorthand nature of these two kinds of diagrams requires care in their interpretation. For example, the numbers displayed by SgpDec in a transformation tree diagram to indicate states do not generally indicate the original singleton states. Instead, they represent an additional encoding, where each number at a given level of the tree refers to a particular tile at that level.[24] Consequently, *the encoding actually changes from level to level*, and this must be borne in mind. Given the combinatorially large algebraic structure of automata and the high level of abstraction of the holonomy theorem, however, the fact that anything at all can be visualized is a remarkable feat in itself. The discussion that follows complements a similar discussion found in [3].

4.3 Properties of the Skeleton and Transformation Tree Diagrams

The skeleton diagram shows a number of important structural features of a holonomy decomposition that are meant to make it easier to understand and visualize the cascade product (sub-wreath product) that emulates the original automaton:

1. The skeleton shows the number of levels (or of sub-automata, or of coordinate positions if the decomposition is interpreted as the expansion in an abstract positional number system) needed in the decomposition. From the

[24] Currently, the particular enumeration of tiles is chosen somewhat arbitrarily and not shown in the skeleton, although the encoding can be recovered from SgpDec via explicit queries.

diagram, one can read off what the component building-block permutation-reset automata used for the decomposition are. These automata correspond to the places, and their states (tiles of representatives, i.e. reptiles) correspond to values of place-value numerical expansions.[25]

2. At each level, the skeleton shows the macrostates, which are image sets, and which are encoded by reptiles; it does not show transformations explicitly.
3. The equivalence classes of image sets are shown as long rectangles that contain all equivalent image sets. These image sets are drawn as ovals, except that SgpDec chooses one of the tiles as a class representative[26] and draws it as a rectangle. As we mentioned above, its tiles are 'reptiles' (tiles of the representative).
4. For each equivalence class, SgpDec draws the arrows from the representative down to the tiles it is mapped to, which could be on different (lower) levels. Such tiles are always proper subsets of the representative, but they are not necessarily disjoint. (Except for singletons, each image-set is the union of its tiles). In those cases where a tile is not the image of the representative, the tile the arrow points to is called a *Garden of Eden* (GoE) state.[27]
5. There are analogous arrows issuing from the other image sets within an equivalence class. However, these are not drawn because the action on their tiles isomorphic to the action on the reptiles. Not only would indicating them clutter the diagram without providing any new essential information, but the holonomy decomposition exploits this isomorphism structure to eliminate the need for representing them directly in the coordinate form at all. This vastly reduces the size and number of coordinate values that could occur at a given level. Instead, inclusion chains passing through these other image sets are encoded so that only tiles of their representatives appear in the coordinatized form of states.
6. There are many maps between image sets that are present in the semigroup but that are not drawn explicitly in the skeleton diagram (the skeleton is *not* meant to draw the sub-automata at each level). For example, for two image sets to be in an equivalence class means that there exist semigroup elements (words or strings of generators or input symbols concatenated together) that map the tiles onto each other, but no arrows are drawn to depict such maps.[28]
7. When an equivalence class is drawn as a filled dark-gray rectangle, this indicates that a group is acting at that level. The symbol of such a group is drawn inside (or above) the dark rectangle, and it should be understood as acting on the tiles *below* that level. The number of states it acts on is the number of tiles it permutes. These are indicated by the arrows issuing to these tiles from the representative of that class. In addition, each of these permutations of tiles is given by some word w in the generators, and the input sequence w also necessarily also permutes the states *within* the representative.
8. When an equivalence class is drawn as a white rectangle, on the other hand, this indicates that an identity-reset component is acting at that level. No symbols are drawn and the action within the corresponding component automaton is to map the tiles *below* the rectangle to each other using either the identity mapping or constant maps (resets).
9. The skeleton makes it possible to begin to infer what the lift of a given automaton state might be. It is important to remember that for a given state multiple lifts are not only possible but quite common. The lift is constructed by picking a path from the state as singleton to the full state set, which gives an inclusion chain of image sets. Encoding replaces this by a chain in which only reptiles appear, which comprise the "digits" in the coordinatized form of the path (which determines the state of course). To determine the lifts for the state fully, maps from image sets to their representatives and back are used. (See the example in the discussion of the decimal expansion as holonomy above.) The way the lifting of states is done is not always easy or obvious, so we will revisit this below with specific examples from the 3-queens puzzle.
10. Because of the use of representatives in the holonomy decomposition that exploits the isomorphism between the permutation groups acting on the tiles of image sets within the same equivalence class, representatives (and hence reptiles) must be picked by SgpDec at each level that generally do not contain the same singleton state in the diminishing chain of subsets as one travels down the hierarchy of levels. This can cause confusion when interpreting lifts, but as long as one remembers the isomorphism it is not too difficult to "find" the state one is trying to lift in the appropriate tile at a given level. It is necessary to do this kind of scouting when interpreting the output of the SgpDec command for the lift of a given state since the encoding will point to tiles at each level that may or may not contain the original (singleton) state one is trying to lift.

We provide a similar list of properties of transformation trees in order to make it easier to interpret the results that follow.

1. Each generator (input symbol) can be lifted to a set of one or more vectors of dependency functions. Each such vector is a lift for the generator, and there may well be many lifts of a given generator in general. Each lifted transformation

[25] As we saw above, in the case of the decimal expansion, the automata were modulo-10 counters with resets (permutation-reset automata). Their states were the 10 tiles of representatives corresponding to the digits 0, 1, ..., 9.

[26] This is analogous to a coset representative in group theory. However, unlike the situtation for group theory, equivalent image sets may have non-trivial overlap.

[27] In SgpDec the arrows for GoE states used are drawn as dotted lines.

[28] However, SgpDec does keep track of them in its data on the skeleton, and also uses them in the encoding and decoding processes.

can be visualized as a transformation tree for the given generator. A transformation tree completely determines one particular lifted transformation via a level-by-level description of how it acts in coordinate form on lifted states. (Similarly one can lift transformations corresponding to longer input words in the same way, to transformation trees. But lifting generators suffices for the decomposition.)

2. If the cascade product has component transformation semigroups (X_i, S_i), $1 \leq i \leq n$, a transformation tree is a rooted tree that branches $|X_i|$ times from every node at level i, so that a state $(x_1, \ldots, x_n)$ $(x_i \in X_i)$ of the cascade product is identified with a path down the tree. Its nodes are labelled so as to completely determine a semigroup element of the cascade product which, as described above, can be given by a vector $d = (d_1, \ldots, d_n)$ of dependency functions. There is a node in the tree for every proper prefix of such a state. That is, the empty prefix is the root; the tree branches from the root for each state of X_1, etc., so prefix $(x_1, \ldots, x_k)$ of the state gives a path down the tree along branches labelled $x_1, ..., x_k$, in that order, to a particular node corresponding exactly to this prefix. This node is labelled by an element of S_{k+1}, namely by $d_{k+1}(x_1, ..., x_k)$, the value of the $k+1$st dependency function $(0 \leq k < n)$. In the holonomy decomposition this value will always be either a permutation (possibly the identity) or a reset. Identity transformations are shown as small empty circles. Thus, the root is labelled by the element $d_1(\emptyset) \in S_1$, and the node reached from the root along path $(x_1, ..., x_k)$ is labelled by $d_{k+1}(x_1, ..., x_k)$. NB: Branches for states in X_n and hence nodes at the end of a path n are not shown, since dependency functions depend only on the state coordinates up to but not including the level acted upon.
3. The states referred to at each level in the transformation tree are numbered from $1, \cdots, n_k$ (where n_k depends on the level). However, each number that is shown as a label on a branch of the transformation tree or inside a node does not usually refer to one of the original states of the automaton, but rather to a fixed enumeration of the reptiles from level k. Each number refers to a particular *tile*, and therefore at each level and for each node this number can change. Note that this enumeration of tiles as numbers might be different in different runs of SgpDec. Appropriate SgpDec commands exist that can display this encoding explicitly as textual command-line output.
4. In accordance with the definition of the wreath product and/or the dependency functions, to apply a particular transformation lift one proceeds top-down. If the state has coordinate $(x_1, \ldots, x_n)$, and a lift with a particular transformation tree T is applied, the semigroup element that applies to each coordinate depends on the state in which the system finds itself at a given moment. This determination is done in a downward and loop-free manner. So after applying the lifted transformation the ith coordinate of the new state is $x_i^{d_i(x_1,\ldots,x_{i-1})}$ (for $i = 1$, this is $x_1^{d(\emptyset)}$). Thus, if we run down the tree from the root along the path $(x_1, \ldots, x_{i-1})$, we come to a node labelled with the transformation $d_i(x_1, \ldots, x_{i-i})$, a permutation or reset, to be applied to x_i. In other words, suppose we have a state of the original automaton and find a lift of that state in coordinate form. Then given a generator (or word) in the automaton, a lift of that generator (or more general transformation) is given by a particular transformation tree. We can coordinatewise deduce the next lifted state as just described by using the transformation tree, and therefore the next state.
5. The top component of the tree is a node and it shows the action of the given generator s on the set of tiles of the full state set X of the automaton. This is either a permutation of the tiles or a reset to some tile containing the image set X^s.
6. The action at the second coordinate level (indicated by labelled nodes one step away from the root in the the tree) is determined by which of the tiles of the full state set X the system is in. Therefore, in general multiple edges issue from the top node of the tree, each edge is connected to a node at the level below that shows the action at that level, and the edge is labelled with (the encoded value of) the state (tile) that determines that action. In other words, the labels of the edges issuing from Level 1 of the tree are the possible input parameters of the dependency function for Level 2.
7. The structure just described is recursive at each of the lower nodes of the tree (as described in Points 2 and 4 above).
8. Identity transformations are drawn as small empty circles or with label () for the identity permutation
9. The *value* of a dependency function is a semigroup element given by Eq. (3). Each node indicates the *effect* of such value as a transformation of tiles, i.e. a semigroup element acting on the possible state coordinates at the given level. In other words, each node shows the coordinatewise action on (the encoded values for) the source states to the (encoded value for) target states, as formalized by Eq. (9). Due to the encoding using tiles of representatives in the holonomy decomposition, this coordinatewise action will always be a permutation or a constant function (reset).
10. The coordinatewise effect of the lift of a given semigroup element (or generator) at each level i is dependent on which state $(x_1, \ldots, x_n)$ the cascade product is in at any given moment (through the dependency functions), and is given by applying the dependency functions d_i of the lift to the prefix $(x_1, \ldots, x_{i-1})$ of this state – this can be visualized in its transformation tree as the sequence of semigroup elements $s_i = d_i(x_1, \ldots, x_{i-1}) \in S_i$ that label the single strand of nodes issuing from the top node and sequentially connected by edges, i.e. without forks, along the path determined by coordinates of the current state.

In the following we describe the skeleton diagrams for the 4 cases examined and then we focus on the transformation trees for the 4 generators of Case 2.

5 Results

5.1 Skeleton Diagrams for the 4 Cases

Case 1: 3-Queens, R1, No Symmetries As stated in Section 3.1, no permutation groups should be observed in the holonomy decomposition of Case 1 because all transformations are idempotent. However, some kind of patterns can be observed by following the arrows Q_1 to Q_8 from the top level of the skeleton (see Figure 13); prominently, the image of the whole state set $\{1, \ldots, 18\}$ under each of those Q_i transformations is a set of 3 states, and each of those image sets has 2 tiles in the level below (each belonging to a different equivalence class) that cover it. These similarities in the skeleton suggest that these transformations affect the whole state set in a similar way, and they also suggest that there may be a way to relate partially (or even completely) the Q_i transformations to each other. We explore this in more detail in Section 5.1. The transition denoted by the dotted arrow indicates that the subset $\{18\}$ is not an image set of the whole state set under any sequence of moves. This reflects the fact that state 18, which represents the empty board, is a GoE state. This is because the board will always have *at least one queen* after any non-empty sequence of Q_i transformations since queens can only be removed by placing a conflicting queen on a different square; therefore no non-empty sequence of transformations can map any valid state to the empty board.

Another particular behaviour is described by transformation Q_9, which corresponds to placing a queen in the centre of the board. It is always possible to go to State 9 from any of the other 17 states by means of transformation Q_9, meaning that transformation Q_9 is a reset for all states of the automaton. Therefore, the effect of transformation Q_9 over the whole state set is very simple to describe, and can be observed by looking at its lift, which is the same also for Case 2 (see Figure 19).

Case 2: 3-Queens, R1, with Symmetries The purpose of Case 2 is to illustrate the effect of adding symmetries to Case 1. By definition, symmetries preserve the structural properties by acting as renamings of states, so symmetries themselves do not add levels nor nodes to the skeleton as they do not produce new image sets. Figure 13 illustrates[29] how symmetries induce an equivalence relation among the images of transformations Q_1 to Q_8. Specifically, the long rectangles that contain ovals indicate equivalence classes of (macro)states. The smaller rectangle that can be found within each larger rectangle is the representative of the equivalence class (analogous to a coset representative in group theory). As a consequence of the existence of these equivalence classes it is not necessary to consider 8 different component building blocks as we have to do in Case 1, but it is enough just to consider one. There is no change in the number of tiles of the full state set (which are the Level 1 coordinate values), but the introduction of symmetries has substantially reduced the number of components needed at each of Levels 2 and 3 in the decomposition (from 8 copies to 1 in each case).

The minimal generating set for Case 1 involves 9 transformations: Q_1-Q_9. By contrast, in Section 3.2 we stated that a minimal set of generators of the semigroup for this case has only 4: H, CS, Q_9, and any one from Q_1-Q_8. However, in addition to H, CS, and Q_9 labels on the arrows (showing words yielding image sets), Figure 13 displays Q_1 to Q_8 transformations instead of equivalent strings in terms of a chosen set of 4 generators.[30] As we mentioned previously, the generators of the semigroup of a transformation semigroup are an algebraic way to describe the input symbols of the input alphabet of the corresponding automaton. Thus, the automaton for Case 1 has a 9-symbol alphabet whereas the automaton for Case 2 is able to do with a 4-symbol alphabet thanks to the introduction of symmetries. Symmetries also make the skeleton much simpler by making many macrostates equivalent. This translates to a holonomy decomposition using far fewer building block components and having far fewer states.

[29] Not explicitly, but only in comparison to the skeleton of Case 1.

[30] Q_i representations are shorter as words than $CS^{i-1}Q_1$ representations. Furthermore, there is not a unique representation for transformation Q_i in terms of the 4 generators.

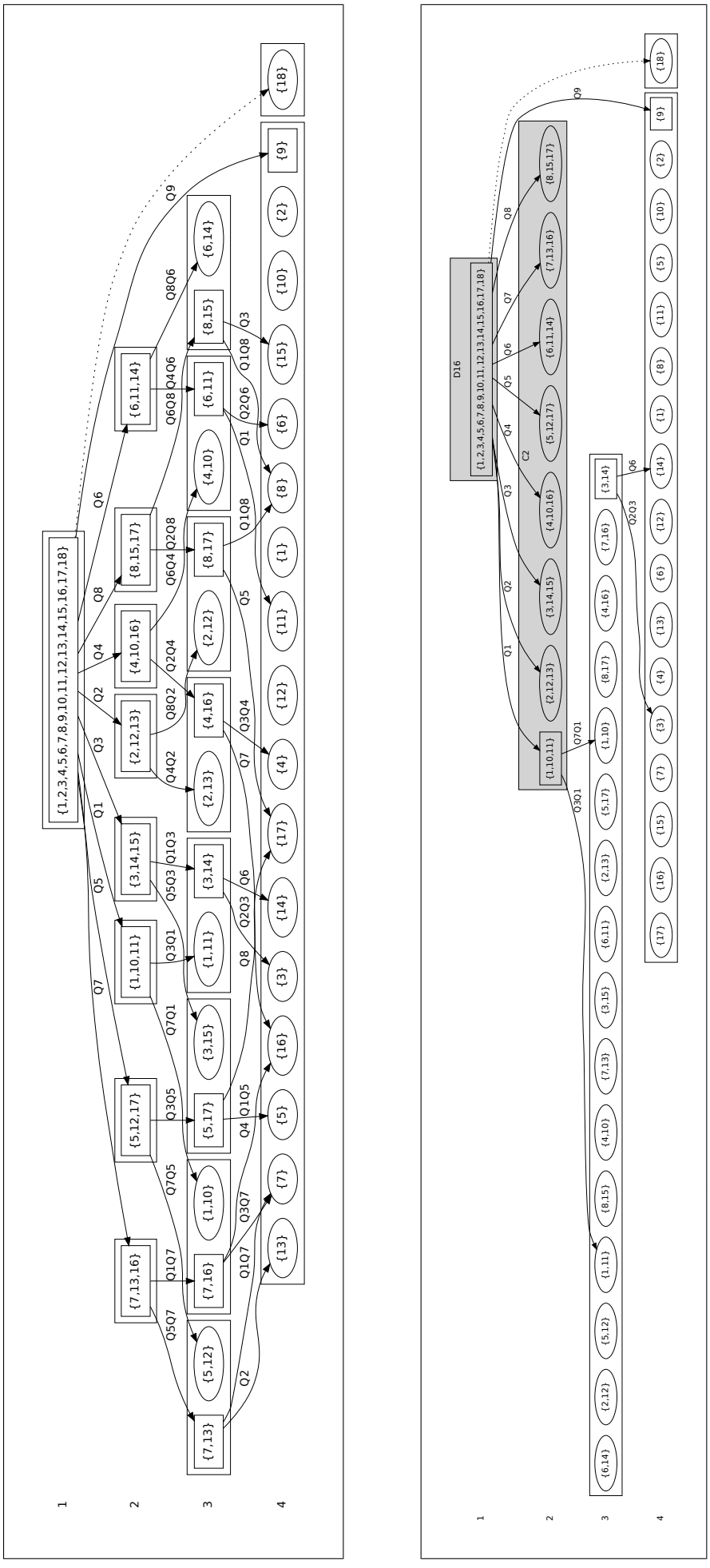


Fig. 13: **Skeletons of automata with non-reversible transformations (Cases 1 and 2)**

Case 3: 3-Queens, R2, No Symmetries For Case 3, we highlight the consequences of applying *locally reversible* transformations instead of the non-reversible ones of Case 1. The skeleton for this case is shown in Figure 14. The term *locally reversible* refers to transformations that act on a particular proper subset of states and permute it.[31] Not only are the elements of such a subset permuted by the permutation group, but that permutation group also acts on the tiles of this subset, which are the ovals and small rectangles in the levels below and are pointed to by arrows. In this case, the permutation groups that are observed are the two-element cyclic group C_2 and $C_2 \times C_2$. The C_2 permutation groups act on two elements (say 0 and 1) and their action (say $flip$) permutes the elements non-trivially, so 0 $flips$ to 1 and 1 $flips$ to 0. The $C_2 \times C_2$ permutation groups act on four elements with the first component C_2 flipping between two macrostates that are not affected by the second component C_2, and similarly the second C_2 flips two other macrostates unaffected by the first.

Consider, for example, the permutation group generated by Q_1 and Q_6 acting on the proper subset $\{1, 6, 11, 18\}$. In the skeleton, it can also be observed that this permutation group acts as a group $C_2 \times C_2$ would act on the subsets $\{6, 11\}$, $\{1, 18\}$, $\{1, 11\}$ and $\{6, 18\}$, which are the tiles of representative image set $\{1, 6, 11, 18\}$:

- $\{6, 11\}^{Q_1} = \{11, 6\} = \{6, 11\}$
- $\{6, 11\}^{Q_6} = \{18, 1\} = \{1, 18\}$
- $\{1, 18\}^{Q_1} = \{18, 1\} = \{1, 18\}$
- $\{1, 18\}^{Q_6} = \{11, 6\} = \{6, 11\}$

is the action of one C_2 group on the two elements $\{6, 11\}$ and $\{1, 18\}$, while

- $\{1, 11\}^{Q_1} = \{18, 6\} = \{6, 18\}$
- $\{1, 11\}^{Q_6} = \{11, 1\} = \{1, 11\}$
- $\{6, 18\}^{Q_1} = \{11, 1\} = \{1, 11\}$
- $\{6, 18\}^{Q_6} = \{18, 6\} = \{6, 18\}$

is the action of the other C_2 group on the elements $\{1, 11\}$ and $\{6, 18\}$. It follows that the action Q_1Q_6 is the same as that of Q_6Q_1 on this set of tiles, swapping macrostates within both pairs, so the permutation group is commutative. Moreover, the square of any of these permutations is the identity on this set of reptiles. As both C_2 groups act independently, we observe a $C_2 \times C_2$ group. It should be noticed that this is not a transitive action, but has two disjoint orbits.

In Cases 3 and 4, State 18 is no longer a GoE state. The local reversibility of the Q_i has a profound impact on the structure of the skeleton as it adds a level to it (see Figure 14).

Case 4: 3-Queens, R2, with Symmetries In Case 4 we illustrate the impact of symmetries when they are added to Case 3. The skeleton for this case is shown in Figure 14 as well. Similarly to Case 2, these added symmetries induce an equivalence relation among transformations Q_1 to Q_8, from which we can conclude that we need not lift all of Q_1 to Q_8 in the modelling of the 3-queens puzzle, since lifting one will generate lifts of the others when combined with the lift of CS. The skeleton displays what seems to be a highly organized, hierarchical structure, similar to the one seen in Case 2 in Figure 13, but also displaying the permutation groups induced by the locally reversible transformations of Case 3 and the additional level that results. Similar to the effect of introducing symmetries to Case 1, introducing them here to Case 3, we observe (1) the number and identity of tiles of the full state set, i.e. at the top level, are not affected (although these tiles now have more elements due to local reversibility), and (2) again the number of equivalence classes is greatly reduced at the intermediate

[31] Remember that a group has an identity transformation and an inverse transformation for every element.

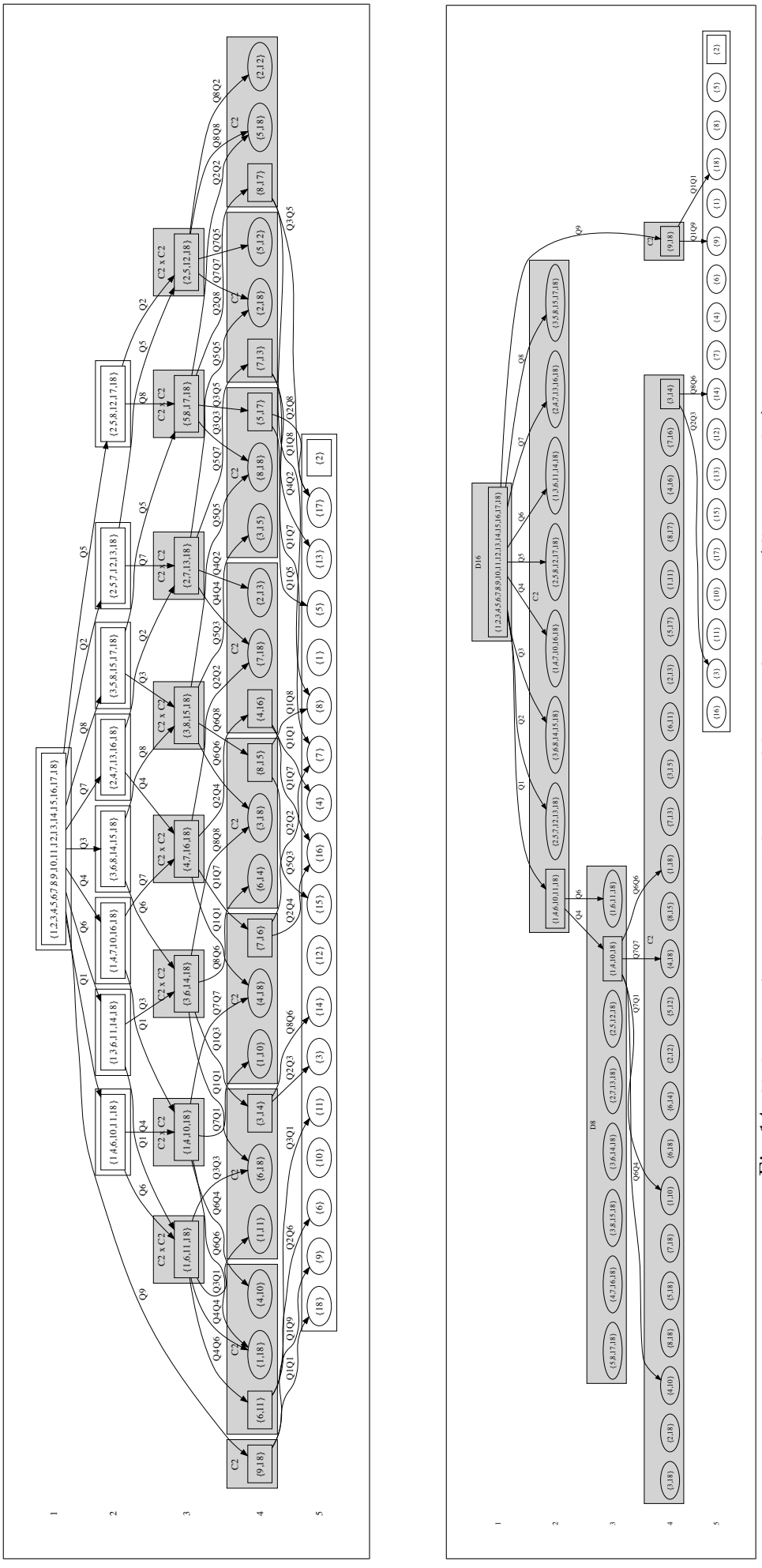


Fig. 14: **Skeletons of automata with reversible transformations (Cases 3 and 4)**

levels (levels 2, 3 and 4), resulting in far fewer components and states for the holonomy decomposition for Case 4 relative to Case 3.

The hierarchical structure observed in the skeleton yields one of the possible cascade products that emulate the automaton presented in Figure 8 (though Figure 8 only shows one of the possible minimal generator representations for the sake of clarity). The holonomy decomposition has one building-block component for each representative macrostate above the singleton level, and representatives at the same level in the skeleton result in components running in parallel at the same level in the cascade product. To complete the structural analysis of the 3-queens puzzle, in the following section we study the lifts of transformations for Case 2, which follows R1 and includes symmetries. We focus on this one since, as we have just seen, its holonomy decomposition is the simplest of the 4 cases considered with the least number of states, levels, and fewest building-block components needed to construct the emulating cascade product.

5.2 Image Sets and Transformation Trees for Case 2

In this Section we describe how to interpret the dependency functions that SgpDec creates during the holonomy decomposition of Case 2. As mentioned, SgpDec creates an internal numbering of image sets (see Figure 19) that we already discussed as a level-dependent encoding of macrostates and that the reader must not confuse with the singleton states of the original automaton (presented in Figure 3) as we will be addressing both in this section. Please refer to Table 2 to see how the two numbering systems are related. Since the encoding of macrostates or tiles results in an additional level of abstraction than we have so far presented, we make a few additional observations that should help the reader follow the discussion of the examples in the rest of the paper.

The easiest way to understand a transformation tree is to remember that the labels on the different parts of the tree are associated with the different *roles* that states can play relative to a given transformation. On the one hand, when discussing singleton states of the original automaton or lifted states, the only roles one needs to keep track of are **source states** on which transition functions or semigroup generators act to produce **target states**. When discussing the transitions acting on any one level i and for a given generator s, on the other hand, one also needs to keep track of the state coordinates from the levels above that act as arguments to the pertinent dependency function at level i. Macrostates (or reptiles) playing this additional role will be referred to as **input coordinates** to the dependency function. According to Eq. (3), the dependency function then produces semigroup element s_i acting at level i to map any source state (or tile) to a target state (or tile) at that particular level. In the agreed shorthand notation for the dependency functions (see Eq. (9)), the only states that are shown explicitly in the expression of a transformation at a given level are the input coordinates as arguments of the dependency function d_i acting on the source state x_i at level i. In the transformation tree, thus, the input coordinate states for a given level i label the edges between levels 1-2, 2-3, $\cdots$, $(i-1)$-i of the tree *above* nodes at level i; the source states x_i for that level i are represented as the edge labels *below* this node between levels i and $(i+1)$ (if drawn[32]); and the target state at the same level i can be deduced by applying the transformation shown within the node *at* the given level i of the tree.

We remind the reader that for any one automaton there is a single skeleton diagram and as many transformation trees as there are symbols in the input alphabet (or semigroup generators).[33] With that in mind, the final general point about transformation trees we need to establish before we drill down into specific examples is how the information they contain should be interpreted relative to the skeleton. The transformation trees of a given automaton are all related to its skeleton in the same way: namely, the component action encoded by a transformation tree at each of its labelled nodes at

[32] The edges and corresponding labels below the singletons at the last level are omitted from transformation trees and, as already mentioned, so are nodes and edges where the cascade acts trivially (i.e., by identity permutations only) below them.

[33] There will generally be more transformation trees corresponding to words in the generators, but only a collection of such trees corresponding to generators must be given to carry out the decomposition and these comprise a more minimal (smaller) set.

Encoding	Macrostate
$(\cdot,\cdot,\cdot)$	$\{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18\}$
$(1,\cdot,\cdot)$	$\{1,10,11\}$
$(2,\cdot,\cdot)$	$\{2,12,13\}$
$(3,\cdot,\cdot)$	$\{3,14,15\}$
$(4,\cdot,\cdot)$	$\{4,10,16\}$
$(5,\cdot,\cdot)$	$\{5,12,17\}$
$(6,\cdot,\cdot)$	$\{6,11,14\}$
$(7,\cdot,\cdot)$	$\{7,13,16\}$
$(8,\cdot,\cdot)$	$\{8,15,17\}$
$(9,\cdot,\cdot)$	$\{9\}$
$(10,\cdot,\cdot)$	$\{18\}$
$(1,1,\cdot)$	$\{1,10\}$
$(1,2,\cdot)$	$\{1,11\}$
$(2,1,\cdot)$	$\{2,12\}$
$(2,2,\cdot)$	$\{2,13\}$
$(1,1,1)$	$\{1\}$
$(1,2,1)$	$\{1\}$
$(1,1,2)$	$\{10\}$

Table 2: **Some encodings of macrostates provided by SgpDec. The $\cdot$ is a placeholder. Observe how resolution increases as the encodings become more precise. For example, the macrostate $\{1\}$ has an encodings $(1,1,1)$ and $(1,2,1)$, both fitting the encodings $(1,1,\cdot)$ and $(1,2,\cdot)$ of macrostates $\{1,10\}$ and $\{1,11\}$ respectively; and at the same time these satisfy the encoding $(1,\cdot,\cdot)$ of macrostate $\{1,10,11\}$.**

a given level i is a **mapping from the set of tiles of Level i representative image sets (i.e. from reptiles of Level i equivalence class representatives in the skeleton) to this same set of reptiles.** It is *not* a mapping defined on the set of image sets at Level i, but rather on the set of reptiles of image sets at Level i. NB: These reptiles could generally be at various different levels, but must each occur strictly below Level i in the skeleton diagram. In other words, everything we have said in the previous paragraph about source states, target states, and input coordinates applies exclusively to reptiles rather than to tiles, at any one level. This is a demonstration of the efficiency of holonomy decomposition, which is able to emulate the behaviour of the original automaton by cascading only the actions on reptiles, by encoding all their isomorphic copies at each level. However, as we will see in the next section, it makes the tracing of a particular algorithm visiting a particular set of singleton states of the original automaton that much more subtle. In fact, in addition to lifting each state into a hierarchy of nested components (inclusion tile chains) one also needs to keep track of how each such chain is mapped to and from its corresponding subduction reptile chain.

These general properties of transformation trees are now demonstrated through a detailed discussion of the lifted states and transformations for Case 2 that is based on the abstract coordinatization visualized in Figure 15. The structural property of this puzzle that drives its coordinatization is the spatial relationship between the square on which a queen is located and the squares on the board that are not attacked by that queen. This property gives rise to the 'V' patterns highlighted in the figure, and provides an intuitive, visual explanation for the fact that the same singleton state can have more than one lift.

Not obvious when we started out is that we obtain a 3-level coordinate system for this puzzle. The top level coordinate encodes the most recent move, the second level encodes a branch of a 'V' pattern where another queen might be, and the third level encodes the number of queens on the board (unless the top level coordinate already captures this – as is the case with the empty board and the configuration with a queen in the centre of the board). These coordinates can be updated in a top-down manner, using the generalized "carry rules" given by the transition trees for the lifts of generators. Thus, given a move, computing the new number of queens, for example, depends on an identity or reset that is given as a function of the previous move and on which branch of the V-pattern indicates the position of a possible second queen (whether or not there is one). The coordinates are hierarchically organized,

so, for instance, updates to the first two coordinates do not depend at all on the number of queens currently on the board.

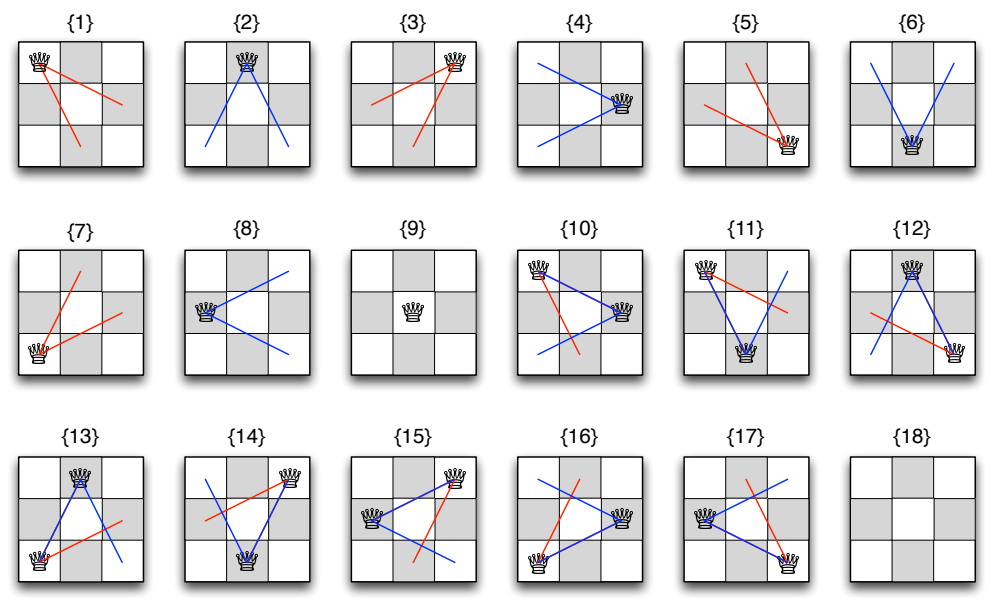


Fig. 15: **Visualization of the abstract coordinatization of the 3-queens puzzle**

In the rest of this section we provide a more formal discussion of these points through two different results: one is the plain output given by SgpDec after holonomy decomposition, and the other is a new modelling of the 3-queens puzzle based on the output given by SgpDec, but not created by SgpDec. First, we show how to find semantics from the modelling provided by SgpDec (henceforth known as modelling α), as it is carried out purely symbolically using computer algebra with no readily interpretable semantics provided by the tool.[34] Then, we show how based on modelling α we came up with a second modelling β that we consider easier to understand. We refer the reader to the respective sections so that the different modellings α and β can be read in parallel.

5.3 Coordinatization of the 3-Queens Puzzle: The 3-Tuple State Space

The original automaton of the 3-queens puzzle has 18 valid configurations (states). Let us call this set of states $\mathbb{Q}_3$. In Figure 3 we presented an encoding for states in $\mathbb{Q}_3$ by using 18 numbers, so that we could refer to valid configurations (states) with numbers in the set $\mathbf{18} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$. Let us call that encoding $X : \mathbf{18} \to \mathbb{Q}_3$, so $X(i)$ $(1 \leq i \leq 18)$ unambiguously refers to *one* valid configuration.

For the holonomy decomposition of Case 2, from the skeleton diagram (lower part of Figure 13), we read off that we have three components to cascade. The first two are permutation-reset automata with groups D_{16} (the dihedral group of the octagon) and C_2 (a modulo-2 counter), while the third building block is an identity-reset automaton (the flip-flop). The number of states in these components is the number of tiles of representatives (reptiles) at levels 1, 2 and 3 of the skeleton, i.e., 10, 2, and 2, respectively. Let us use the bold numbers **10** to refer to the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and **2** for the set $\{1, 2\}$. Consider the state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \times \{1, 2\} \times \{1, 2\}$ as the Cartesian product of three sets. In order to refer to states in $\mathbb{Q}_3$, SgpDec provides a somewhat arbitrary encoding (seen in Figure 16) and which we could call $X_\alpha^\uparrow : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \to \mathbb{Q}_3$. We remind

[34] However, semantics of the coordinate form is present in the mapping constructed by SgpDec that decodes lifted, coordinatized states back to states of the original automaton, and the mapping from the cascade lifts of generators to generators of the automaton.

the reader that skeletons and lifts provided by SgpDec provide a *blind suggestion* for a new abstract positional number system based on the structural properties of the automaton, so the $X_\alpha^\uparrow$ may appear to be a rather arbitrary encoding for some states. Section 5.4 explains the encoding $X_\beta^\uparrow$, which was created after the analysis of the $X_\alpha^\uparrow$ encoding.

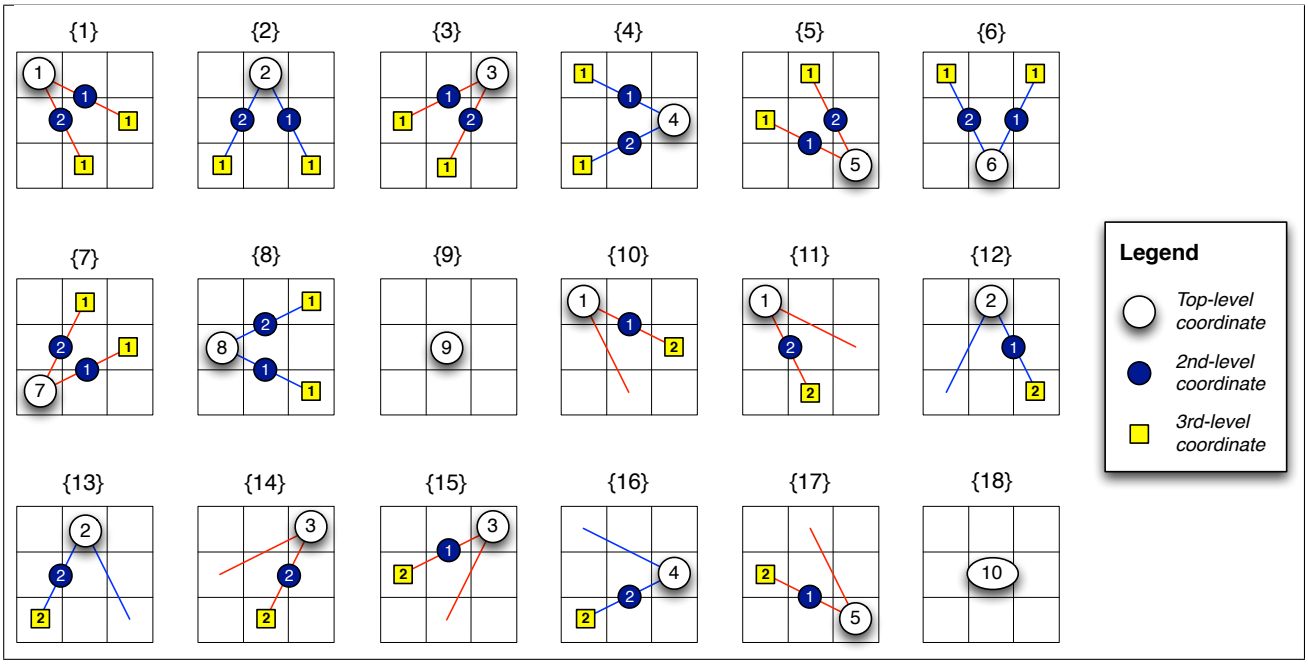


Fig. 16: **Visualization of a subset of the possible lifts of the 18 states of the 3-queens puzzle. The lift encodings chosen by SgpDec are consistent with the transformation lifts shown in Figure 19.**

Transformation Tree of Q_1 Figure 19 displays the transformation tree of Q_1, which is an output of SgpDec after holonomy decomposition. We proceed with the interpretation by levels of the transformation tree of Q_1 provided by SgpDec. For the following discussion, it helps to refer to Figures 15, 16, 17, and 18. It may also help the reader to refer to Section 5.4, as we provide a reference point in the form of another transformation that emulates Q_1 in the new 3-tuple coordinate system. The explanation is structured as an example transformation between the lifts of three pairs of (singleton) states: $\{3\} \xrightarrow{Q_1} \{1\}$, $\{14\} \xrightarrow{Q_1} \{11\}$, and $\{15\} \xrightarrow{Q_1} \{1\}$, which we can also express in exponential notation as

$$3^{Q_1} = 1, \qquad 14^{Q_1} = 11, \qquad 15^{Q_1} = 1. \tag{15}$$

- The top node in the transformation tree of Q_1 in Figure 19 indicates 'Reset to 1'. Consistently with the empty set $\emptyset$ in the dependency function $d_1(\emptyset)$ that appears in Eq. (10), this action 'Reset to 1' at the top level does not depend on any particular **input coordinates**, it applies regardless of which of the 18 states the board is in. Thus, the **source** state at this level is no one specific state, but any one of the 18 possible states. Because we are at the top level the **target** state, indicated as '1', is specified with minimum "resolution": as shown in Table 2, it is specified as the tile $\{1, 10, 11\}$. In other words, if the automaton is in any of the 18 possible states, when we apply Q_1 the result is that the board will end up in one of the three states 1, 10, or 11. No other possibilities apply according to the rules of Case 2. For the example we are discussing here, as shown in Figure 17 the top-level source state for state 3 or 14 or 15 of the original automaton is the reptile $\{3, 14, 15\}$ encoded as 3; and the target state, the reptile $\{1, 10, 11\}$, is encoded as 1. Since the tiles of the top level are all reptiles, at this level Figure 18 looks the same.
- To explain the next level of the transformation lift we now observe from Figure 19 that if the input coordinate[35] is the subset or tile $\{3, 14, 15\}$, the corresponding node on Level 2 is 'Reset to 2', since 2 encodes the tile $\{1, 11\}$. The reason is summarized in Table 4 and visualized in Figure 17: we can see that the result of applying Q_1 to $\{3, 14\}$ or $\{3, 15\}$ is always (a subset of) $\{1, 11\}$. Thus, 'Reset to 2' means a reset to $\{1, 11\}$. This establishes the target state for Level 2. What about the source state for Level 2? Well, the transformation necessarily acts on all the tiles of

[35] The **source** state for Level 1 is also the **input coordinate** to the dependency function for Level 2, and *one* of the input coordinate states for Levels 3 and beyond (below, on the page), if any.

$\{3, 14, 15\}$ for this particular branch, so the source state can be either $\{3, 14\}$ or $\{3, 15\}$. Figure 18 shows what this action translates into for the reptiles of the Level 2 representative shown in Figure 13 for Case 2.

- For the encoding of the final level of the tree, now we have to observe whether the Level 2 macrostate below $\{3, 4, 15\}$ in the inclusion chain is its tile $\{3, 15\}$ or its tile $\{3, 14\}$. If it is $\{3, 15\}$ (which is encoded as 1 in the second coordinate position), then the source singleton can be either $\{3\}$ or $\{15\}$; applying Q_1 to either gives the singleton $\{1\}$ (which is encoded as 1 in the third coordinate position), hence the 'Reset to 1' in the third node of the Q_1 transformation lift along the 3-1 path. If instead the macrostate is $\{3, 14\}$ (which is encoded as 2 in the second coordinate position), Q_1 will map $\{3\}$ or $\{14\}$ to $\{1\}$ or $\{11\}$, respectively. But $\{1\}$ or $\{11\}$ are encoded by reptiles $\{3\}$ and $\{14\}$, so the map is the identity map and a small empty circle is drawn in the Q_1 transformation tree shown in Figure 19. Since – as shown in Figure 13 – SGPDEC chose the Level 3 reptiles as $\{3\}$ and $\{14\}$, the holonomy action shown in Figure 18 requires some unravelling, but is still valid as it is isomorphic to what Figure 17 shows, which is exactly what it encodes.

Table 4 shows how the lifts of 3, 14, and 15 we set out to analyse map to the lifts of 1 and 11 under the action of Q_1. The table also shows how a given singleton state may have more than one lift. The justification for this fact can be seen in Figure 15: for cases where there is only one queen the two edges shown are both valid, meaning that the value of the second coordinate can be either 1 or 2. The value of the third coordinate corresponds to the number of queens present (except for States 9 and 18 for which it does not matter), so that limits the possibilities to 1's for States 1 to 8 and to 2's for States 10 to 17.

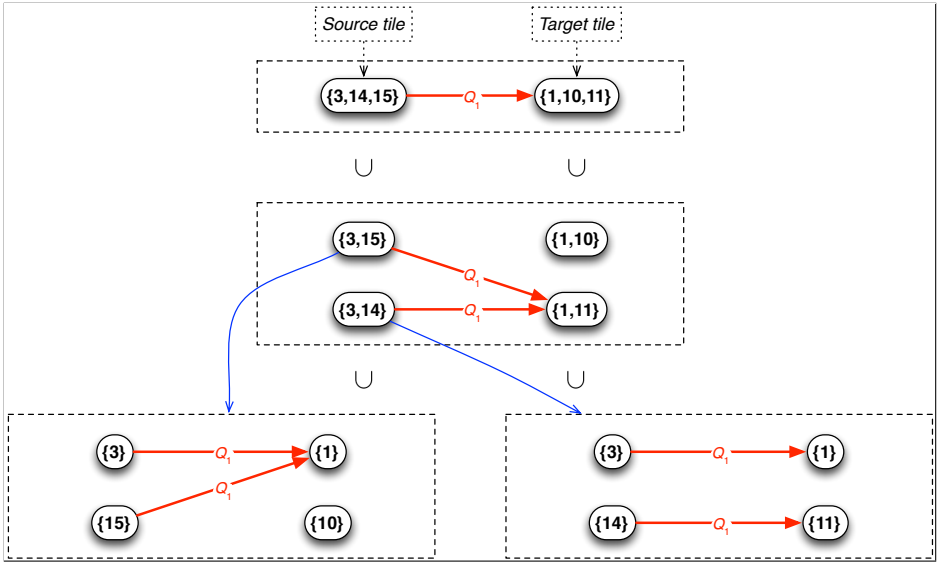


Fig. 17: **Visualization of the Q_1 map acting on the unencoded inclusion tile chains $\{3\} \subset \{3, 14\} \subset \{3, 14, 15\}$, $\{3\} \subset \{3, 15\} \subset \{3, 14, 15\}$, $\{14\} \subset \{3, 14\} \subset \{3, 14, 15\}$, and $\{15\} \subset \{3, 15\} \subset \{3, 14, 15\}$, for singleton states $\{3\}$, $\{14\}$, and $\{15\}$, for Case 2**

Top-Level Source Macrostate	Singleton Source State	Singleton Target State	Lifted Source State(s)	Lifted Target State(s)
3	3	1	(3,1,1) (3,2,1)	(1,2,1)
3	14	11	(3,2,2)	(1,2,2)
3	15	1	(3,1,2)	(1,2,1)

Table 3: **Action of Q_1 on singleton states belonging to Macrostate 3 as well as on the corresponding lifts, Case 2**

Transformation Tree of Q_9As we observed before, the transformation Q_9 resets the whole set of 18 elements to the set $\{9\}$. Figure 19 displays the transformation tree of Q_9 when it is lifted to the 3-tuple space. The action denote "$\rightarrow 9$" represents reset on the first coordinate of the 3-tuple notation, and the action over the other coordinates is the identity. Therefore, the image of an element (A, B, C)

under the lifted transformation of Q_9 will be $(9, B, C)$, which is always a lift of the configuration 9 with just a queen at the centre.

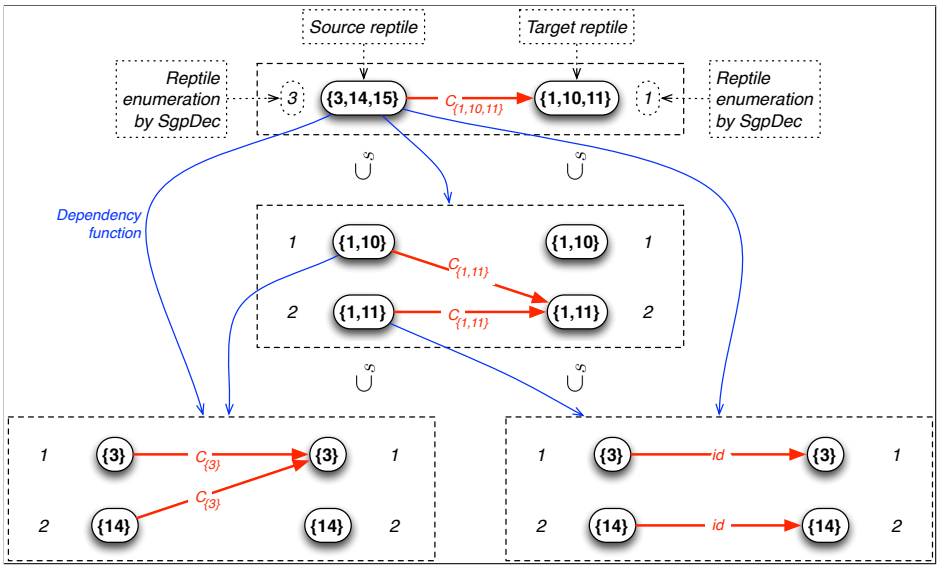


Fig. 18: **Componentwise visualization of the lift of the Q_1 map between the source and target encoded subduction reptile chains corresponding to Figure 17. $C_{\{\cdot\}}$ indicates a constant map and id the identity map. Compared to the previous figure, inclusion $\subset$ of tile chains is replaced by the subduction relation $\subset_S$ on reptiles.**

Transformation Tree of *CS* Figure 19 shows that the transformation tree of CS is more complex than the transformation tree of Q_9. When CS is applied, the macrostates that correspond to the tiles of the whole state set permute:

$$(\{1, 10, 11\}, \{2, 12, 13\}, \{3, 14, 15\}, \{4, 10, 16\}, \{5, 12, 17\}, \{6, 11, 14\}, \{7, 13, 16\}, \{8, 15, 17\})$$

It also leaves the other two tiles of the configuration space fixed: $\{9\}$ and $\{18\}$ corresponding to one queen at the centre and the empty board are unaffected by the cyclic shift. Since these image sets refer to values in the first component of the 3-tuple representation, it is natural to see the queens rotating through the board in a cyclic motion as the lifted CS transformation is applied. We expect the lifted CS transformation to permute the first component in the abstract positional number system as it refers to the location of a queen, and this can be observed in Figure 19. The second level of the permutation tree shows that some flips need to be performed in order to match the correct encoding of states e.g. applying the cyclic shift to the state $(4, 1, 2)$ yields state $(5, 2, 1)$ (see Figure 16). In Section 5.4 we provide another lifted version of CS matching encoding $X_\beta^\uparrow$, showing how this lifted transformation *only* needs to act on the first component of the 3-tuple, unlike the one provided by SgpDec. The action at the third level is by the identity map, as CS does not change the number of queens on the board.

Transformation Tree of *H* Figure 19 displays the transformation tree of H. Similar to the transformation tree of CS, when H is applied the tiles of the whole set of states *flip* over the horizontal axis as they are permuted

$$(\{1, 10, 11\}, \{7, 13, 16\})\,(\{2, 12, 13\}, \{6, 11, 14\})\,(\{5, 12, 17\}, \{3, 14, 15\})$$

Again the tiles corresponding to the empty board and to the queen at the centre are fixed, but so are the two tiles corresponding to having a queen in the middle of the right or the left edge of the board (those containing 4 and 8, respectively).

Similar to the action of CS, H should flip queens on the first component over the horizontal axis. The action of H should leave queens on squares 4 and 8 fixed, which is what is observed with the action on the first component of the 3-tuple coordinate system. The second level of the transformation tree displays how the V patterns flip for states with queens on squares 4 and 8 once this permutation is applied, which is only natural. Interestingly, no other flips need to be performed, contrary to the other lifted H transformation presented in Section 5.4, which always flips the V patterns. The action at the third level is by the identity map, as H does not change the number of queens on the board.

Source State	**Original Target States**				**Lifted Source State(s)**	**Encoded Lifted Target States**			
	Q_1	Q_9	H	CS		Q_1	Q_9	H	CS
1	1	9	7	2	(1,1,1) (1,2,1)	(1,1,1)	(9,1,1)	(7,1,1)	(2,1,1)
2	1	9	6	3	(2,1,1) (2,2,1)	(1,1,1)	(9,1,1)	(6,1,1)	(3,1,1)
3	1	9	5	4	(3,1,1) (3,2,1)	(1,1,1)	(9,1,1)	(5,1,1)	(4,1,1)
4	10	9	4	5	(4,1,1) (4,2,1)	(1,1,2)	(9,1,1)	(4,1,1)	(5,1,1)
5	1	9	3	6	(5,1,1) (5,2,1)	(1,1,1)	(9,1,1)	(3,1,1)	(6,1,1)
6	11	9	2	7	(6,1,1) (6,2,1)	(1,2,2)	(9,1,1)	(2,1,1)	(7,1,1)
7	1	9	1	8	(7,1,1) (7,2,1)	(1,1,1)	(9,1,1)	(1,1,1)	(8,1,1)
8	1	9	8	1	(8,1,1) (8,2,1)	(1,1,1)	(9,1,1)	(8,1,1)	(1,1,1)
9	1	9	9	9	(9,1,1) (9,1,2) (9,2,1) (9,2,2)	(1,1,1)	(9,1,1)	(9,1,1)	(9,1,1)
10	10	9	16	12	(1,1,2) (4,1,2)	(1,1,2)	(9,1,1)	(4,2,2)	(2,1,2)
11	11	9	13	13	(1,2,2) (6,2,2)	(1,2,2)	(9,1,1)	(2,2,2)	(2,2,2)
12	1	9	14	14	(2,1,2) (5,2,2)	(1,1,1)	(9,1,1)	(3,2,2)	(3,2,2)
13	1	9	11	15	(2,2,2) (7,2,2)	(1,1,1)	(9,1,1)	(1,2,2)	(3,1,2)
14	11	9	12	16	(3,2,2) (6,1,2)	(1,2,2)	(9,1,1)	(2,1,2)	(4,2,2)
15	1	9	17	10	(3,1,2) (8,2,2)	(1,1,1)	(9,1,1)	(5,1,2)	(1,2,2)
16	10	9	10	17	(4,2,2) (7,1,2)	(1,1,2)	(9,1,1)	(1,1,2)	(5,1,2)
17	1	9	15	11	(5,1,2) (8,1,2)	(1,1,1)	(9,1,1)	(3,1,2)	(1,2,2)
18	1	9	18	18	(10,1,1) (10,1,2) (10,2,1) (10,2,2)	(1,1,1)	(9,1,1)	(10,1,1)	(10,1,1)

Table 4: **Global table for the action of the 4 generators of Case 2 on singleton states and lifted states. In the four right-most sub-columns, only one of the possible lifted target states is shown for each target singleton; it is obtained by applying the lifted generator to one of the lifted source states. Other lifts of a source state could yield other lifts of the *same* target states than shown (but these appear in the column of the lifted source states in the same row). Always, applying the lifted generator to the lift of a state yields *some state lift* of that generator applied to the state.**

Summary of Coordinatization Suggested by SgpDec Figure 16 shows a visualization of the complete coordinatization given by SgpDec for Case 2. The state lifts shown were picked specifically because they are consistent with all the transformation lifts shown in Figure 19.

- The encoding of the top coordinate is the same for all transformation lifts and it is shown in Table 2. Aside from 10 which indicates the empty board, the other numbers indicate the square on which the first queen has been placed.
- The encoding for the second coordinate is always the same in the sense that it always refers to one of the two edges of the Vs shown in Figure 15. However, depending on the top coordinate the two edges refer to different subsets of two singleton states each. Furthermore, the numbering is not always the same, meaning that Edge 1 is on the left of State 1, but on the right of State 3. At first it seemed that the relationship was one of dihedral flips, but that does not work for all cases. Rather, the numbering shown enables the transformation lifts shown in Figure 19 to be valid for *all* singleton states, which are suitably represented at the top level by only the 10 encoded states shown in Table 2.
- The encoding of the third coordinate is also context-dependent, but in any case it always indicates whether there are 1 or 2 queens present.

It may help the reader if we interpret a few of the state and transformation lifts shown in Figure 19 in light of Figure 16. For example, the lift of H says that if we are in State 4 at the top level

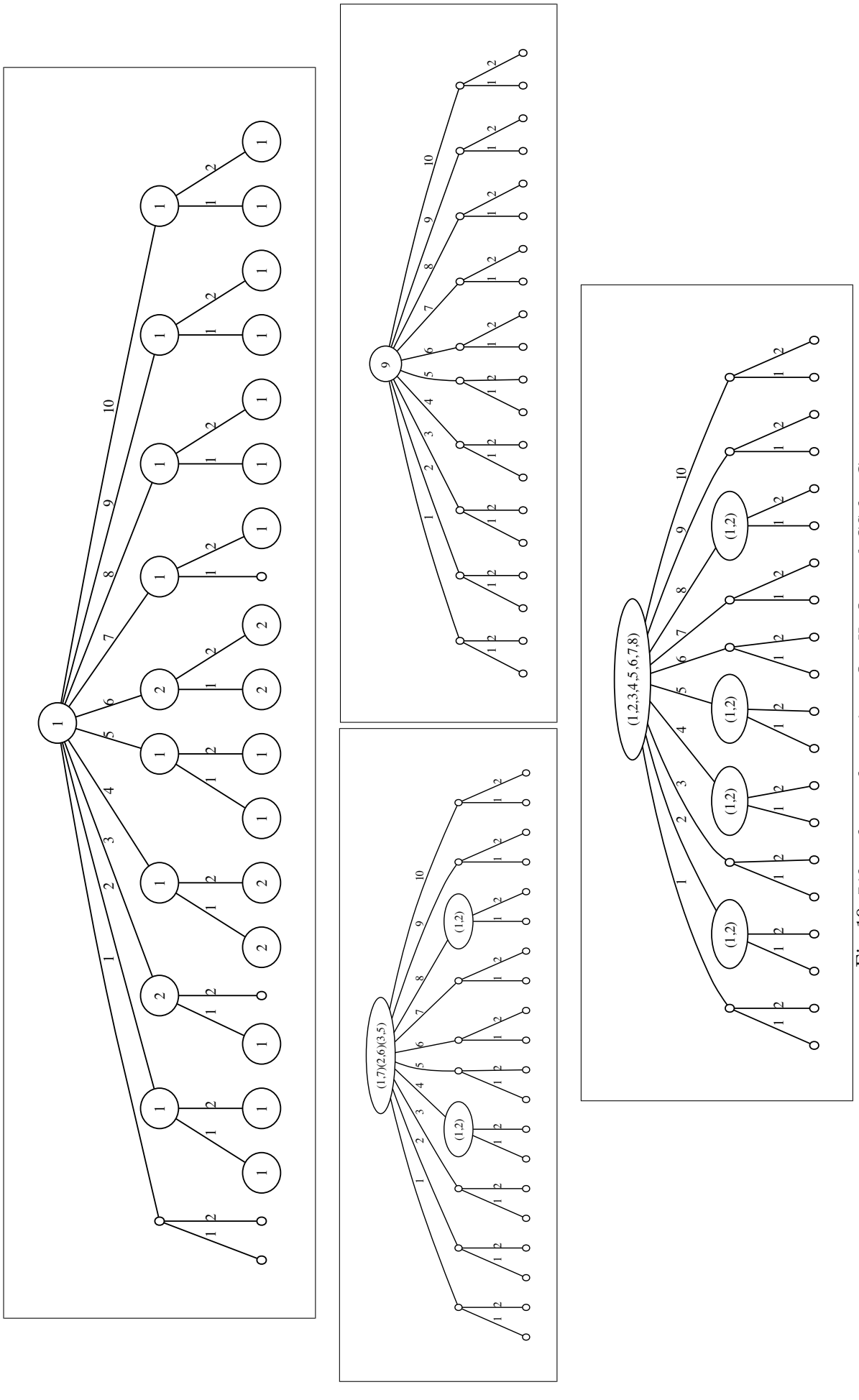


Fig. 19: **Lifts of transformations Q_1, H, Q_9, and CS for Case 2**

then the action of H is to permute the second coordinates. In fact, State 16 is obtained by flipping 4 horizontally and then switching Edge 1 with Edge 2. The reason it had to switch is that it would have interfered with the second coordinate of State 10. In other words, whereas $(4, 2, 2) = 16$, $(4, 1, 2) = 10$. State 8 behaves in the same way.

Moving to the lift for CS, we can see the cyclic permutation at the top. Now, if we are in State 2 at the top level and we apply CS, then this figure tells us that the edges have to switch. This is not to accommodate the singleton state 2 but, rather, 12 and 13. In fact, $12 = (2, 1, 2)$ moves to $14 = (3, 2, 2)$, for which we can see the second coordinate is a 2, while $13 = (2, 2, 2)$ moves to $15 = (3, 1, 2)$, for which the second coordinate is a 1. The same identical pattern is repeated for all the other state labels for this transformation lift.

5.4 β-Modelling: New Encoding of States

As an alternative to SgpDec's encoding, we propose a surjective encoding of states $X_\beta^\uparrow : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \to \mathbf{18}$ in hierarhcial coordinate form. This is shown graphically in Figure 20, where the image of 3-tuple (A, B, C) with $A \in \mathbf{10}$, $B \in \mathbf{2}$, $C \in \mathbf{2}$ under $X_\beta^\uparrow$ is a valid configuration (state) that satisfies:

- If $A = 10$, then (A, B, C) refers to the empty board; otherwise (A, B, C) refers to a state where there is *at least one* queen, located on square A.
- If A is between 1 and 8 and $C = 1$, then (A, B, C) refers to a state where there is only one queen on the board, i.e. located on square A. However, if $C = 2$, (A, B, C) refers to a state where there are two queens on the board. The location of the second queen is given by coordinate B. Given that in general there are two possible places to put the second queen, if $B = 1$ then the second queen is located at the tip of the left edge of the V; otherwise the second queen is located at the tip of the right edge of the V.

The definition above favours an easy description of encoding semantics. The encoding $X_\beta^\uparrow$, which can be observed in Figure 20, is based on consistently assigning the value 1 of the second component to the left edge of the V. We consider encoding $X_\beta^\uparrow$ to be more helpful for understanding the abstract positional number system, but it is a representation that was obtained *after* reflecting over the output given by SgpDec– as the software in no way can realize which encodings might make it easier for a human being to interpret a particular coordinatization of the 3-queens puzzle. In the following sections we present the transformations that emulate Q_1, CS and H following the new encoding $X_\beta^\uparrow$. The definitions of these new transformations can be considered transformation trees.

The Transformation $Q_{1\beta}^\uparrow$Consider the following definition of transformation $Q_{1\beta}^\uparrow$:

Definition 9. $Q_{1\beta}^\uparrow$ is a transformation in the state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, so $Q_{1\beta}^\uparrow : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \longrightarrow \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, such that for any 3-tuple $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, the image (D, E, F) of (A, B, C) under $Q_{1\beta}^\uparrow$ satisfies:

- D is always 1 (action by a reset)
- The value of E depends on the value of A:
  - If $A = 1$ then $E = B$ (action by identity)
  - If $A = 4$ or $A = 7$ then $E = 1$. (reset)
  - If $A = 3$ or $A = 6$ then $E = 2$. (reset)
  - Otherwise $E = B$. (identity)
- The value of F depends on the value of A and B:
  - If $A = 1$ then $F = C$ (identity)
  - If $A = 4$ or $A = 6$ then $F = 2$ (action by reset)
  - If $A = 3$ and $B = 1$ then $F = C$ (identity)
  - If $A = 7$ and $B = 2$ then $F = C$ (identity)
  - Otherwise $F = 1$ (reset).

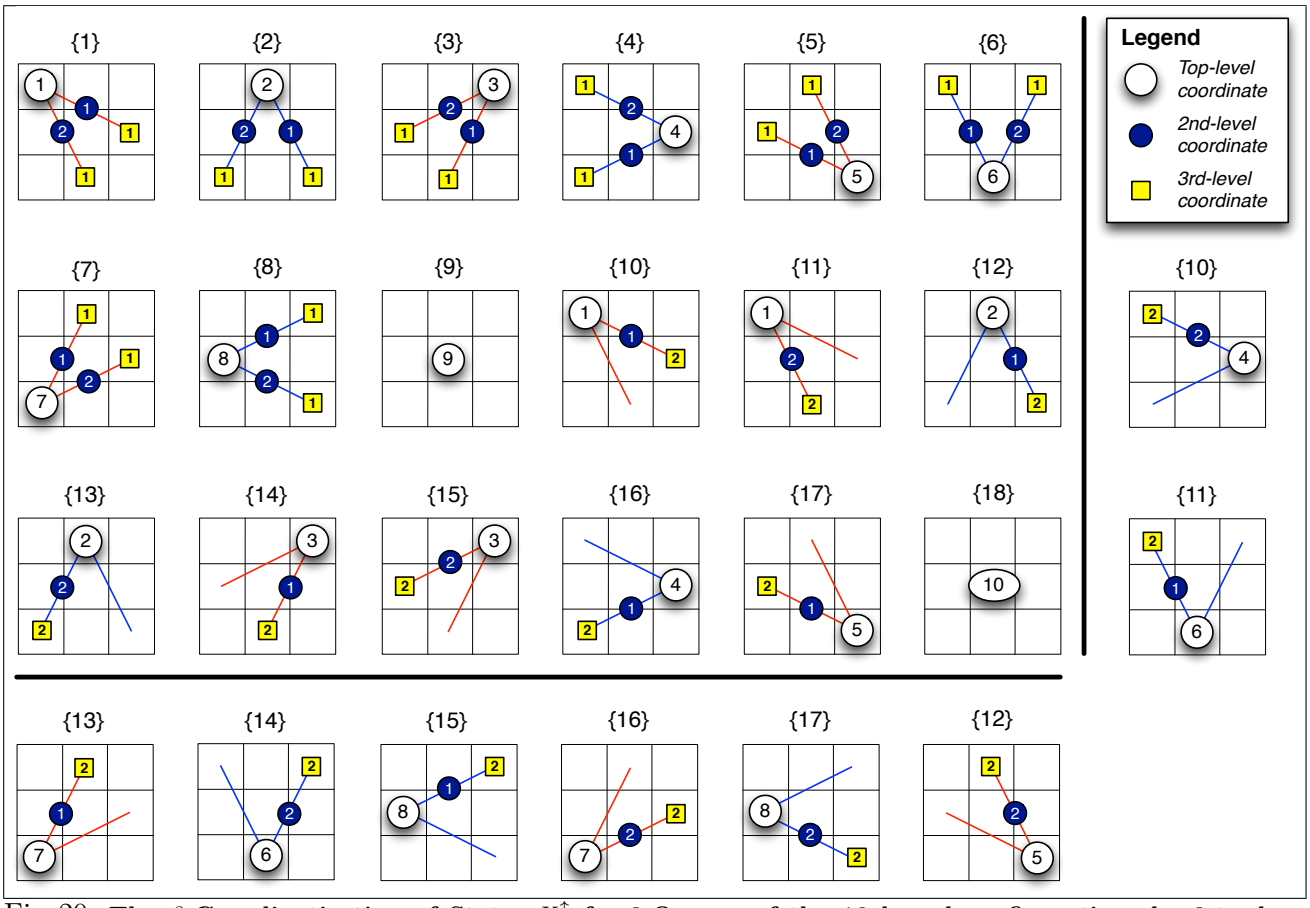


Fig. 20: **The β-Coordinatization of States $X_\beta^\uparrow$ for 3-Queens of the 18 board configurations by 3-tuples (A, B, C) in $10 \times 2 \times 2$, proposed after analysis of the results of holonomy decomposition provided by SgpDec. Note: as before, configurations can have more than one lift (coordinatized encoding), but the encoding determines the configuration uniquely.**

Note that the definition of $Q_{1\beta}^\uparrow$ shows a clear hierarchical dependence, as the values of E depend on A and the values of F depend on A and B. But, even more strongly, $Q_{1\beta}^\uparrow$ is actually a transformation cascade: $Q_{1\beta}^\uparrow$ can be equivalently specified directly as the transformation cascade defined by dependency functions as visualized by its transformation tree in Figure 21. Nodes are labelled with the transformation giving the coordinatewise action as a function of higher-level states (given by the labels on edges leading from the root to the node), if any. (This component transformation was given in paratheses in the case-by-case definition of $Q_{1\beta}^\uparrow$ and is seen in Figure 21 as a node label.)

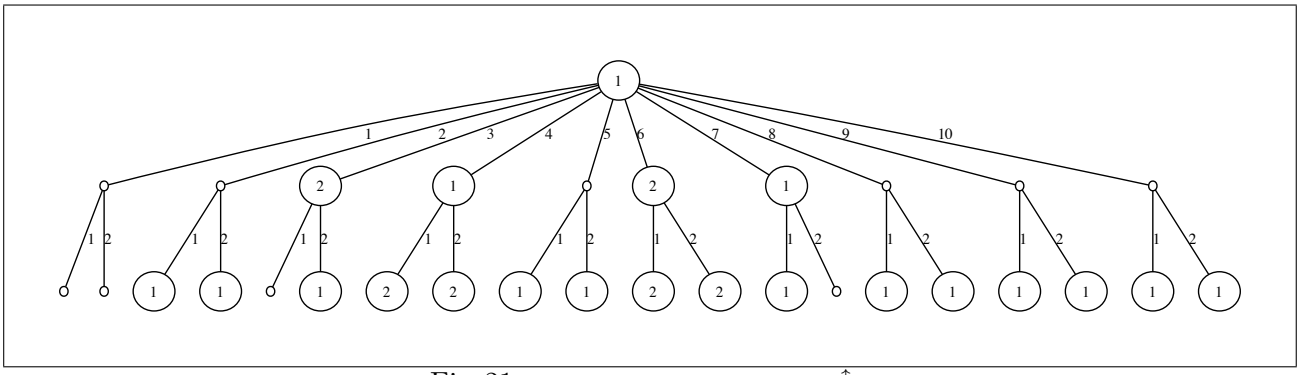


Fig. 21: **Transformation tree of $Q_{1\beta}^\uparrow$**

The transformation $Q^{\uparrow}_{1\beta}$ is a lift of the transformation Q_1 in $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, given that Q_1 was defined for the 18-element encoding. We also say that $Q^{\uparrow}_{1\beta}$ *emulates* Q_1 with respect to the state coordinatzation $X^{\uparrow}_{\beta}$. Formally:

Lemma 1. $Q^{\uparrow}_{1\beta}$ emulates Q_1 *in* $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ *with respect to the* β*-coordinatization of states.*

Proof. To show that $Q^{\uparrow}_{1\beta}$ emulates Q_1, it must be shown that the action of $Q^{\uparrow}_{1\beta}$ over all encoded states $(A, B, C) \in \mathbf{10}\times\mathbf{2}\times\mathbf{2}$ is the same as the action of Q_1 over the corresponding encoded state $d \in \mathbf{18}$; that is, $X^{\uparrow}_{\beta}((A, B, C))\cdot Q_1 = X^{\uparrow}_{\beta}((A, B, C) \cdot Q^{\uparrow}_{1\beta})$. This is shown in Table 5. □

(A, B, C)	$X^{\uparrow}_{\beta}((A, B, C))$	$X^{\uparrow}_{\beta}((A, B, C)) \cdot Q_1$	$(A, B, C) \cdot Q^{\uparrow}_{1\beta}$	$X^{\uparrow}_{\beta}((A, B, C) \cdot Q^{\uparrow}_{1\beta})$	$X^{\uparrow}_{\beta}((A, B, C)) \cdot Q_1 \stackrel{?}{=} X^{\uparrow}_{\beta}((A, B, C) \cdot Q^{\uparrow}_{1\beta})$
$(1, 1, 1)$	1	1	$(1, 1, 1)$	1	true
$(1, 1, 2)$	10	10	$(1, 1, 2)$	10	true
$(1, 2, 1)$	1	1	$(1, 2, 1)$	1	true
$(1, 2, 2)$	11	11	$(1, 2, 2)$	11	true
$(2, 1, 1)$	2	1	$(1, 1, 1)$	1	true
$(2, 1, 2)$	12	1	$(1, 1, 1)$	1	true
$(2, 2, 1)$	2	1	$(1, 2, 1)$	1	true
$(2, 2, 2)$	13	1	$(1, 2, 1)$	1	true
$(3, 1, 1)$	3	1	$(1, 2, 1)$	1	true
$(3, 1, 2)$	14	11	$(1, 2, 2)$	11	true
$(3, 2, 1)$	3	1	$(1, 2, 1)$	1	true
$(3, 2, 2)$	15	1	$(1, 2, 1)$	1	true
$(4, 1, 1)$	4	10	$(1, 1, 2)$	10	true
$(4, 1, 2)$	16	10	$(1, 1, 2)$	10	true
$(4, 2, 1)$	4	10	$(1, 1, 2)$	10	true
$(4, 2, 2)$	10	10	$(1, 1, 2)$	10	true
$(5, 1, 1)$	5	1	$(1, 1, 1)$	1	true
$(5, 1, 2)$	17	1	$(1, 1, 1)$	1	true
$(5, 2, 1)$	5	1	$(1, 2, 1)$	1	true
$(5, 2, 2)$	12	1	$(1, 2, 1)$	1	true
$(6, 1, 1)$	6	11	$(1, 2, 2)$	11	true
$(6, 1, 2)$	11	11	$(1, 2, 2)$	11	true
$(6, 2, 1)$	6	11	$(1, 2, 2)$	11	true
$(6, 2, 2)$	14	11	$(1, 2, 2)$	11	true
$(7, 1, 1)$	7	1	$(1, 1, 1)$	1	true
$(7, 1, 2)$	13	1	$(1, 1, 1)$	1	true
$(7, 2, 1)$	7	1	$(1, 1, 1)$	1	true
$(7, 2, 2)$	16	10	$(1, 1, 2)$	10	true
$(8, 1, 1)$	8	1	$(1, 1, 1)$	1	true
$(8, 1, 2)$	15	1	$(1, 1, 1)$	1	true
$(8, 2, 1)$	8	1	$(1, 2, 1)$	1	true
$(8, 2, 2)$	17	1	$(1, 2, 1)$	1	true
$(9, 1, 1)$	9	1	$(1, 1, 1)$	1	true
$(9, 1, 2)$	9	1	$(1, 1, 1)$	1	true
$(9, 2, 1)$	9	1	$(1, 2, 1)$	1	true
$(9, 2, 2)$	9	1	$(1, 2, 1)$	1	true
$(10, 1, 1)$	18	1	$(1, 1, 1)$	1	true
$(10, 1, 2)$	18	1	$(1, 1, 1)$	1	true
$(10, 2, 1)$	18	1	$(1, 2, 1)$	1	true
$(10, 2, 2)$	18	1	$(1, 2, 1)$	1	true

Table 5: **Action of both $Q^{\uparrow}_{1\beta}$ and Q_1 on their corresponding coordinate spaces.**

The Permutation $Q^{\uparrow}_{9\beta}$We now provide a transformation $Q^{\uparrow}_{9\beta}$ that emulates Q_9 in the state space $X^{\uparrow}_{\beta}$:

Definition 10. $Q^{\uparrow}_{9\beta}$ is a transformation in the state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, so $Q^{\uparrow}_{9\beta} : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \longrightarrow \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, such that for a 3-tuple $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, the image (D, E, F) of (A, B, C) under $Q^{\uparrow}_{9\beta}$ satisfies:

- $D = 9$ (reset)
- $E = B$ (identity)
- $F = C$ (identity)

Observe that $Q^{\uparrow}_{9\beta}$ is a cascade with transformation tree shown in Figure 22.

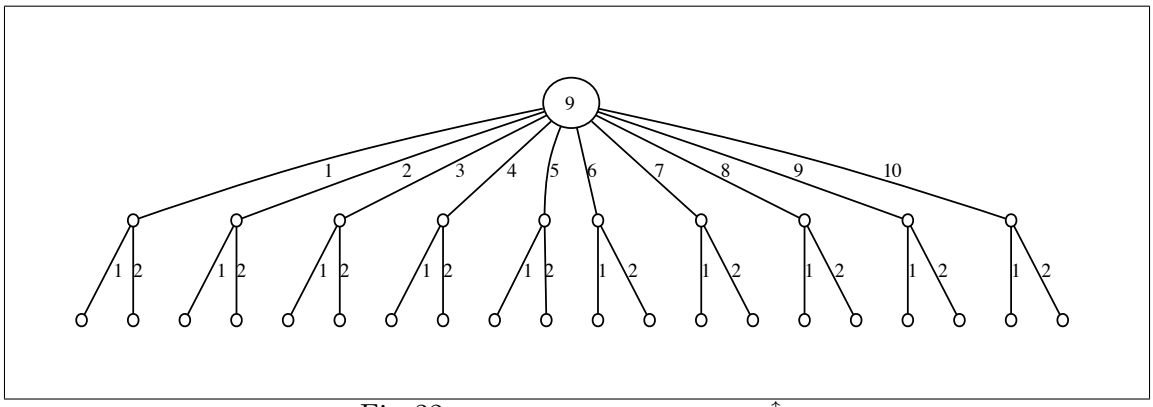


Fig. 22: **Transformation tree of** $Q^{\uparrow}_{9\beta}$

The constant nature of transformation Q_9 is preserved by the definition of $Q^{\uparrow}_{9\beta}$ acting as a reset on the first component of the tuples. The choice of an identity action on the second and third component is rather arbitrary, as it is also possible to act on the second and third component, e.g., via resets to 1 and still satisfy that $Q^{\uparrow}_{9\beta}$ emulates Q_9.

Lemma 2. $Q^{\uparrow}_{9\beta}$ emulates Q_9 *in* $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ *with respect to the* β*-coordinatization of states.*

Proof. To show that $Q^{\uparrow}_{9\beta}$ emulates Q_9, it must be shown that for every encoded state $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$ it holds that $X^{\uparrow}_{\beta}((A, B, C)) \cdot Q_9 = X^{\uparrow}_{\beta}((A, B, C) \cdot Q^{\uparrow}_{9\beta})$. This is shown in Table 6. □

The Permutation $CS^{\uparrow}_{\beta}$We now provide a transformation $CS^{\uparrow}_{\beta}$ that emulates CS in the state space coordinatization given by $X^{\uparrow}_{\beta} : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \to \mathbf{18}$.

Definition 11. $CS^{\uparrow}_{\beta}$ is a transformation in the state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, so $CS^{\uparrow}_{\beta} : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \longrightarrow \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, such that for any 3-tuple $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, the image (D, E, F) of (A, B, C) under $CS^{\uparrow}_{\beta}$ satisfies:

- If $A = 8$, then $D = 1$. If $A > 8$ then $D = A$. Otherwise, $D = A + 1$. (cyclic permutation, but leaving two points fixed)
- $E = B$ (identity)
- $F = C$ (identity)

(A,B,C)	$X_\beta^\uparrow((A,B,C))$	$X_\beta^\uparrow((A,B,C)) \cdot Q_9$	$(A,B,C) \cdot Q_{9\beta}^\uparrow$	$X_\beta^\uparrow((A,B,C) \cdot Q_{9\beta}^\uparrow)$	$X_\beta^\uparrow((A,B,C)) \cdot Q_9 \stackrel{?}{=} X_\beta^\uparrow((A,B,C) \cdot Q_{1\beta}^\uparrow)$
$(1,1,1)$	1	9	$(9,1,1)$	9	true
$(1,1,2)$	10	9	$(9,1,2)$	9	true
$(1,2,1)$	1	9	$(9,2,1)$	9	true
$(1,2,2)$	11	9	$(9,2,2)$	9	true
$(2,1,1)$	2	9	$(9,1,1)$	9	true
$(2,1,2)$	12	9	$(9,1,2)$	9	true
$(2,2,1)$	2	9	$(9,2,1)$	9	true
$(2,2,2)$	13	9	$(9,2,2)$	9	true
$(3,1,1)$	3	9	$(9,1,1)$	9	true
$(3,1,2)$	14	9	$(9,1,2)$	9	true
$(3,2,1)$	3	9	$(9,2,1)$	9	true
$(3,2,2)$	15	9	$(9,2,2)$	9	true
$(4,1,1)$	4	9	$(9,1,1)$	9	true
$(4,1,2)$	16	9	$(9,1,2)$	9	true
$(4,2,1)$	4	9	$(9,2,1)$	9	true
$(4,2,2)$	10	9	$(9,2,2)$	9	true
$(5,1,1)$	5	9	$(9,1,1)$	9	true
$(5,1,2)$	17	9	$(9,1,2)$	9	true
$(5,2,1)$	5	9	$(9,2,1)$	9	true
$(5,2,2)$	12	9	$(9,2,2)$	9	true
$(6,1,1)$	6	9	$(9,1,1)$	9	true
$(6,1,2)$	11	9	$(9,1,2)$	9	true
$(6,2,1)$	6	9	$(9,2,1)$	9	true
$(6,2,2)$	14	9	$(9,2,2)$	9	true
$(7,1,1)$	7	9	$(9,1,1)$	9	true
$(7,1,2)$	13	9	$(9,1,2)$	9	true
$(7,2,1)$	7	9	$(9,2,1)$	9	true
$(7,2,2)$	16	9	$(9,2,2)$	9	true
$(8,1,1)$	8	9	$(9,1,1)$	9	true
$(8,1,2)$	15	9	$(9,1,2)$	9	true
$(8,2,1)$	8	9	$(9,2,1)$	9	true
$(8,2,2)$	17	9	$(9,2,2)$	9	true
$(9,1,1)$	9	9	$(9,1,1)$	9	true
$(9,1,2)$	9	9	$(9,1,2)$	9	true
$(9,2,1)$	9	9	$(9,2,1)$	9	true
$(9,2,2)$	9	9	$(9,2,2)$	9	true
$(10,1,1)$	18	9	$(9,1,1)$	9	true
$(10,1,2)$	18	9	$(9,1,2)$	9	true
$(10,2,1)$	18	9	$(9,2,1)$	9	true
$(10,2,2)$	18	9	$(9,2,2)$	9	true

Table 6: **Action of both $Q_{9\beta}^\uparrow$ and Q_9 on their corresponding coordinate spaces.**

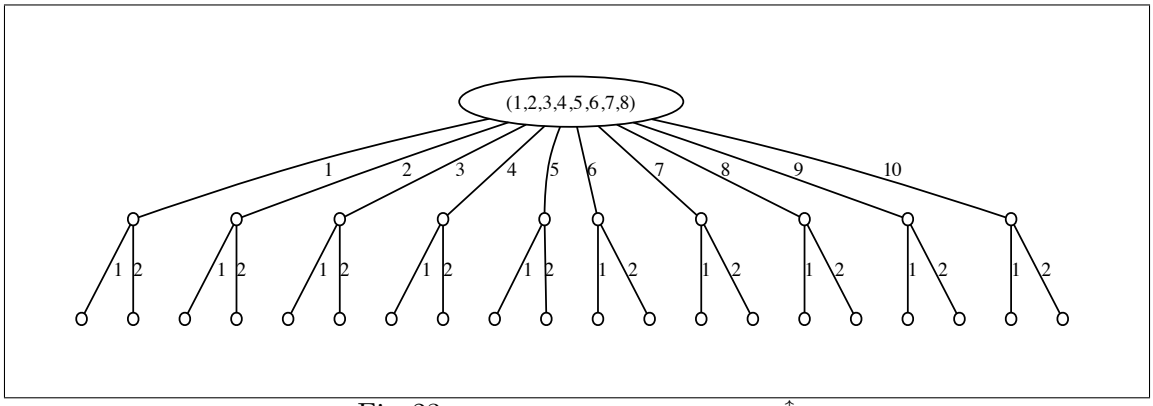


Fig. 23: **Transformation tree of $CS_\beta^\uparrow$**

$CS^{\uparrow}_{\beta}$ is a cascade with transformation tree shown in Figure 23. It is in fact a permutation of $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, since all the component actions are permutations.

We believe transformation $CS^{\uparrow}_{\beta}$ to have more natural semantics than the transformation tree provided by SgpDec for transformation CS, in the sense that $CS^{\uparrow}_{\beta}$ affects only one of the coordinates of the new 3-tuple representation, i.e. the position of the main queen, while the lift of CS provided by SgpDec indicates that two coordinates must be changed when the value of the first coordinate is either 2, 4, 5 or 8 (see Figure 19).

Lemma 3. *$CS^{\uparrow}_{\beta}$ emulates CS in $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ with respect to the β-coordinatization of states.*

Proof. To show that $CS^{\uparrow}_{\beta}$ emulates CS, it must be shown that for every encoded state $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$ it holds that $X^{\uparrow}_{\beta}((A, B, C)) \cdot CS = X^{\uparrow}_{\beta}((A, B, C) \cdot CS^{\uparrow}_{\beta})$. This is shown in Table 7. □

(A, B, C)	$X^{\uparrow}_{\beta}((A, B, C))$	$X^{\uparrow}_{\beta}((A, B, C)) \cdot CS$	$(A, B, C) \cdot CS^{\uparrow}_{\beta}$	$X^{\uparrow}_{\beta}((A, B, C) \cdot CS^{\uparrow}_{\beta})$	$X^{\uparrow}_{\beta}((A, B, C)) \cdot CS \stackrel{?}{=} X^{\uparrow}_{\beta}((A, B, C) \cdot CS^{\uparrow}_{\beta})$
$(1, 1, 1)$	1	2	$(2, 1, 1)$	2	true
$(1, 1, 2)$	10	12	$(2, 1, 2)$	12	true
$(1, 2, 1)$	1	2	$(2, 2, 1)$	2	true
$(1, 2, 2)$	11	13	$(2, 2, 2)$	13	true
$(2, 1, 1)$	2	3	$(3, 1, 1)$	3	true
$(2, 1, 2)$	12	14	$(3, 1, 2)$	14	true
$(2, 2, 1)$	2	3	$(3, 2, 1)$	3	true
$(2, 2, 2)$	13	15	$(3, 2, 2)$	15	true
$(3, 1, 1)$	3	4	$(4, 1, 1)$	4	true
$(3, 1, 2)$	14	16	$(4, 1, 2)$	16	true
$(3, 2, 1)$	3	4	$(4, 2, 1)$	4	true
$(3, 2, 2)$	15	10	$(4, 2, 2)$	10	true
$(4, 1, 1)$	4	5	$(5, 1, 1)$	5	true
$(4, 1, 2)$	16	17	$(5, 1, 2)$	17	true
$(4, 2, 1)$	4	5	$(5, 2, 1)$	5	true
$(4, 2, 2)$	10	12	$(5, 2, 2)$	12	true
$(5, 1, 1)$	5	6	$(6, 1, 1)$	6	true
$(5, 1, 2)$	17	11	$(6, 1, 2)$	11	true
$(5, 2, 1)$	5	6	$(6, 2, 1)$	6	true
$(5, 2, 2)$	12	14	$(6, 2, 2)$	14	true
$(6, 1, 1)$	6	7	$(7, 1, 1)$	7	true
$(6, 1, 2)$	11	13	$(7, 1, 2)$	13	true
$(6, 2, 1)$	6	7	$(7, 2, 1)$	7	true
$(6, 2, 2)$	14	16	$(7, 2, 2)$	16	true
$(7, 1, 1)$	7	8	$(8, 1, 1)$	8	true
$(7, 1, 2)$	13	15	$(8, 1, 2)$	15	true
$(7, 2, 1)$	7	8	$(8, 2, 1)$	8	true
$(7, 2, 2)$	16	17	$(8, 2, 2)$	17	true
$(8, 1, 1)$	8	1	$(1, 1, 1)$	1	true
$(8, 1, 2)$	15	10	$(1, 1, 2)$	10	true
$(8, 2, 1)$	8	1	$(1, 2, 1)$	1	true
$(8, 2, 2)$	17	11	$(1, 2, 2)$	11	true
$(9, 1, 1)$	9	9	$(9, 1, 1)$	9	true
$(9, 1, 2)$	9	9	$(9, 1, 2)$	9	true
$(9, 2, 1)$	9	9	$(9, 2, 1)$	9	true
$(9, 2, 2)$	9	9	$(9, 2, 2)$	9	true
$(10, 1, 1)$	18	18	$(10, 1, 1)$	18	true
$(10, 1, 2)$	18	18	$(10, 1, 2)$	18	true
$(10, 2, 1)$	18	18	$(10, 2, 1)$	18	true
$(10, 2, 2)$	18	18	$(10, 2, 2)$	18	true

Table 7: **Action of both $CS^{\uparrow}_{\beta}$ and CS on their corresponding coordinate spaces.**

The Permutation $H_\beta^\uparrow$Consider the following definition of a transformation $H_\beta^\uparrow$:

Definition 12. $H_\beta^\uparrow$ is a transformation in the state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, so $H_\beta^\uparrow : \mathbf{10} \times \mathbf{2} \times \mathbf{2} \longrightarrow \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, such that for any 3-tuple $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$, the image (D, E, F) of (A, B, C) under $H_\beta^\uparrow$ satisfies:

- If $A = 8$, then $D = 8$. If $A > 8$ then $D = A$, Otherwise, $D = 8 - A$ (permutation)
- If $B = 1$ then $E = 2$. However, if $B = 2$, then $E = 1$ (permutation)
- $F = C$ (identity)

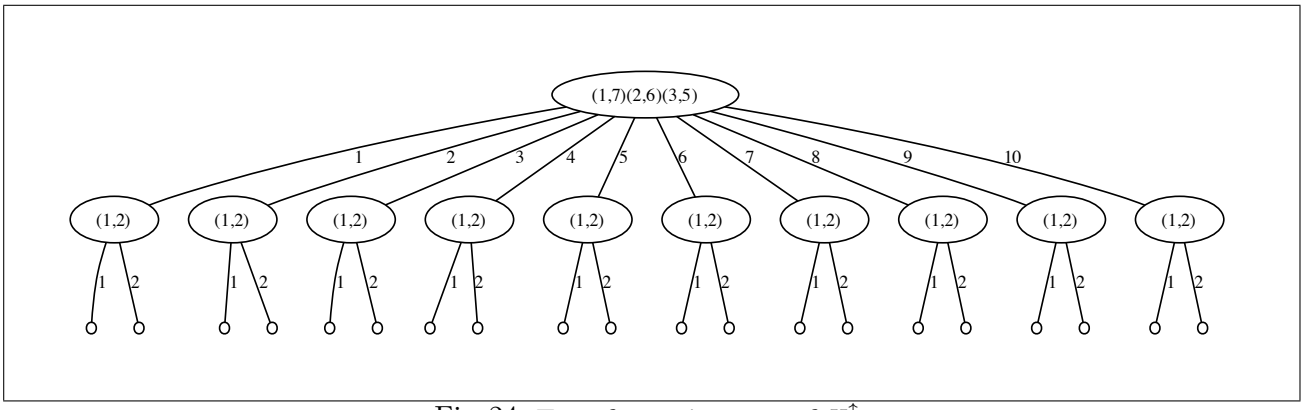


Fig. 24: **Transformation tree of $H_\beta^\uparrow$**

$H_\beta^\uparrow$ is a cascade with transformation tree shown in Figure 24. It is in fact a permutation of $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$, since all the component actions are permutations. Unlike the lift of H produced by SgpDec, its action on the second coordinate is always by the same mapping, the transposition $(1, 2)$, independent of the top-level coordinate. The third component mapping is the identity for both the transformation tree of H given by SgpDec and for $H_\beta^\uparrow$, as H is a permutation and does not change the number of queens. It is important to note that both $H_\beta^\uparrow$ and the transformation tree of H provided by SgpDec are valid lifted versions of H and are able to emulate H given their respective encoding *only*.

Lemma 4. $H_\beta^\uparrow$ emulates H *in* $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ *with respect to the* β*-coordinatization of states.*

Proof. Again, to show that $H_\beta^\uparrow$ emulates H, it must be shown that for every encoded state $(A, B, C) \in \mathbf{10} \times \mathbf{2} \times \mathbf{2}$ it holds that $X_\beta^\uparrow((A, B, C)) \cdot H = X_\beta^\uparrow((A, B, C) \cdot H_\beta^\uparrow)$. This is shown in Table 8. □

We can now formalize the relation between the β model and the 3-queens automaton for Case 2:

Theorem 1. *The* β *model emulates the* 3*-queens automaton for Case 2.*

Proof. As each of the four generators of the Case 2 model is emulated by one respective generator of the β model (as seen in the previous Lemmas 1, 2, 3 and 4), and since $X_\beta^\uparrow$ is a surjective mapping, i.e. for each element in $\mathbf{18}$ we can find at least one 3-tuple in $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ that represents said element in coordinate form, we can conclude that the β model emulates the Case 2 model. □

(A,B,C)	$X^{\uparrow}_{\beta}((A,B,C))$	$X^{\uparrow}_{\beta}((A,B,C)) \cdot H$	$(A,B,C) \cdot H^{\uparrow}_{\beta}$	$X^{\uparrow}_{\beta}((A,B,C) \cdot H^{\uparrow}_{\beta})$	$X^{\uparrow}_{\beta}((A,B,C)) \cdot H \stackrel{?}{=} X^{\uparrow}_{\beta}((A,B,C) \cdot H^{\uparrow}_{\beta})$
$(1,1,1)$	1	7	$(7,2,1)$	7	true
$(1,1,2)$	10	16	$(7,2,2)$	16	true
$(1,2,1)$	1	7	$(7,1,1)$	7	true
$(1,2,2)$	11	13	$(7,1,2)$	13	true
$(2,1,1)$	2	6	$(6,2,1)$	6	true
$(2,1,2)$	12	14	$(6,2,2)$	14	true
$(2,2,1)$	2	6	$(6,1,1)$	6	true
$(2,2,2)$	13	11	$(6,1,2)$	11	true
$(3,1,1)$	3	5	$(5,2,1)$	5	true
$(3,1,2)$	14	12	$(5,2,2)$	12	true
$(3,2,1)$	3	5	$(5,1,1)$	5	true
$(3,2,2)$	15	17	$(5,1,2)$	17	true
$(4,1,1)$	4	4	$(4,2,1)$	4	true
$(4,1,2)$	16	10	$(4,2,2)$	10	true
$(4,2,1)$	4	4	$(4,1,1)$	4	true
$(4,2,2)$	10	16	$(4,1,2)$	16	true
$(5,1,1)$	5	3	$(3,2,1)$	3	true
$(5,1,2)$	17	15	$(3,2,2)$	15	true
$(5,2,1)$	5	3	$(3,1,1)$	3	true
$(5,2,2)$	12	14	$(3,1,2)$	14	true
$(6,1,1)$	6	2	$(2,2,1)$	2	true
$(6,1,2)$	11	13	$(2,2,2)$	13	true
$(6,2,1)$	6	2	$(2,1,1)$	2	true
$(6,2,2)$	14	12	$(2,1,2)$	12	true
$(7,1,1)$	7	1	$(1,2,1)$	1	true
$(7,1,2)$	13	11	$(1,2,2)$	11	true
$(7,2,1)$	7	1	$(1,1,1)$	1	true
$(7,2,2)$	16	10	$(1,1,2)$	10	true
$(8,1,1)$	8	8	$(8,2,1)$	8	true
$(8,1,2)$	15	17	$(8,2,2)$	17	true
$(8,2,1)$	8	8	$(8,1,1)$	8	true
$(8,2,2)$	17	15	$(8,1,2)$	15	true
$(9,1,1)$	9	9	$(9,2,1)$	9	true
$(9,1,2)$	9	9	$(9,2,2)$	9	true
$(9,2,1)$	9	9	$(9,1,1)$	9	true
$(9,2,2)$	9	9	$(9,1,2)$	9	true
$(10,1,1)$	18	18	$(10,2,1)$	18	true
$(10,1,2)$	18	18	$(10,2,2)$	18	true
$(10,2,1)$	18	18	$(10,1,1)$	18	true
$(10,2,2)$	18	18	$(10,1,2)$	18	true

Table 8: **Action of both $H^{\uparrow}_{\beta}$ and H on their corresponding coordinate spaces.**

6 Discussion and Conclusion

During the process of applying holonomy decomposition to the 3-Queens puzzle, several representations of the problem were used and discovered. This suggests that holonomy decomposition helps in the search of newer and more natural representations for problems by focusing on the actions of transformations over the states of the automaton. This is interesting, because after a complete analysis of the actions of transformations it is very likely that there are enough tools and information to provide better and more meaningful representations, and this process can now be performed automatically through existing software, particularly SgpDec.

For example, here we found two holonomy decomposition coordinatizations for the 3-queens puzzle, one using SgpDec and another one modifying it. These both use the same three levels of building blocks. The α or β model is a cascade product of:

- Top level: permutation-reset group with dihedral group D_{16}, the symmetries of the octagon, acting on 10 states (2 states are fixed points under the action)
- Level 2: permutation-reset group acting on two states (with two-element group C_2 acting transitively)
- Level 3: the flip-flop: two-state, identity-reset automaton $(\mathbf{2}, \overline{\langle 1 \rangle})$.

These cascade products all have the hierarchical structure visualized in Figure 25, and emulate the original automaton of the puzzle (Case 2 variant: Rule 1 with symmetries) in coordinate form.[36]

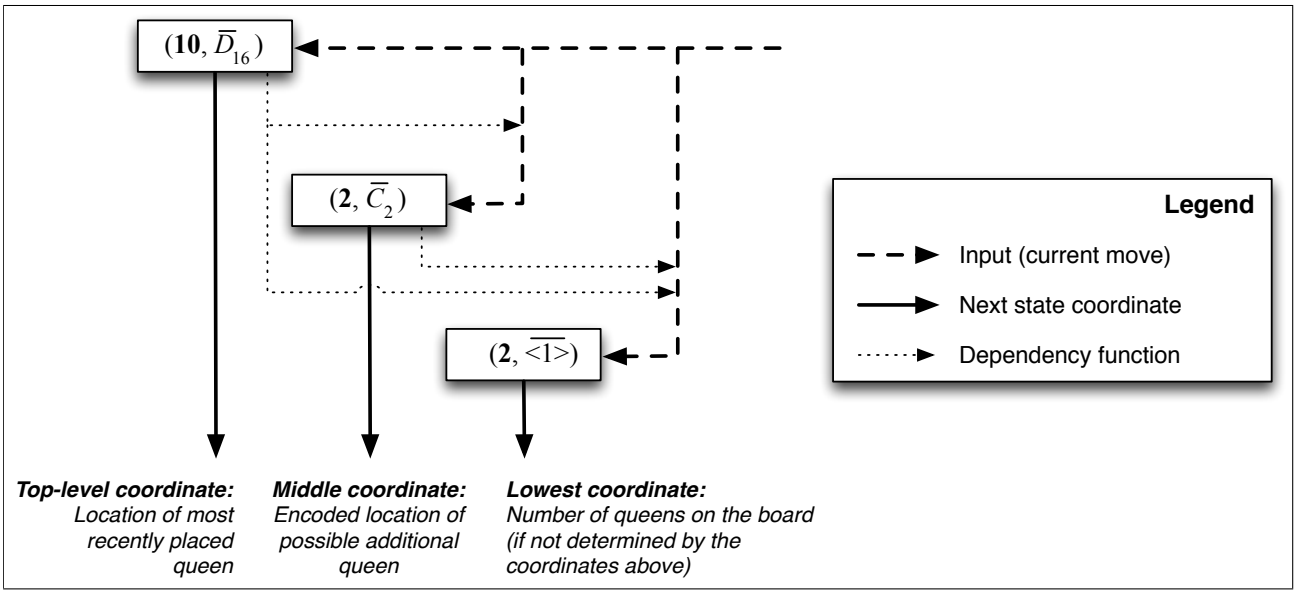


Fig. 25: **Cascade Product Structure for Holonomy Decomposition of the 3-Queens Puzzle. The three levels of building blocks are permutation-reset automata on 10, 2, and 2 states (corresponding to reptiles in the skeleton diagram), respectively, and on which the transformation acting on a given coordinate value is always either a constant map (reset) or an element respectively of the symmetry group of the octagon D_{16}, the order-2 cyclic groups C_2, or the trivial group $\langle 1 \rangle$. Bars over the group indicate that all constant maps on the state set are allowed as transformations in the components as well. Which element of the semigroup (i.e., which permutation or reset) acts is given by a dependency function, depending on the input symbol (current move) and the states of any components higher up in the hierarchy.**

From this we conclude that a main contribution of the process of holonomy decomposition of the 3-queens puzzle is the discovery of the new 3-level state space $\mathbf{10} \times \mathbf{2} \times \mathbf{2}$ with hierarchically related levels. Once the coordinates are given semantics via encodings (in this case $X_\alpha^\uparrow$ and $X_\beta^\uparrow$), the definition of the corresponding transformations in the new state space is not difficult and becomes more natural. At the start it was not at all obvious what the coordinate positions in the holonomy decomposition of the N-queens or 3-queens problem might signify. At the end of the investigation for $N = 3$, we observed that the top-level coordinate in the hierarchical description of board configurations encodes *the most recent move*, the middle coordinate encodes *the possible position of queens on the board given the history of moves* in relation to the first coordinate, and the lowest level coordinate encodes *the number of queens on the board*. At the outset, no one suspected that coordinates that can be interpreted in this particular manner would capture the full state and transformation information in a hierarchical feed-forward manner, e.g. one can make the picture coarser by projecting on just the first or first few coordinates without knowing the number of queens on the board, and transformations in coordinatized form (transformation trees) corresponding to moves can be used to give coarser or finer pictures of the dynamics of the N-queens puzzle at different levels of abstraction (tracking just the most recent move, or also transforming historical constraints on position of a possible second queen,

[36] We remark that while both the α- and β-models coordinatize the transformation semigroup of the 3-queens automaton using the same skeleton diagram and building-blocks to give a holonomy decomposition, they are not isomorphic to each other. This can be verified, e.g., by considering the action of the cascade lifts of move Q_1 in each model on the four state coordinatizations $(10, B, C)$ of the empty board.

or finally tracking also transformations on the actual number of queens depending on the first two coordinates to complete the picture). The finer descriptions never contradict the coarser ones, and there is a convergence on the dynamics of the puzzle given by projections and successive refinements.

One can easily further reduce the state space from 40 states to 34 states since one could choose a unique lift, say (9,1,1), for the state with the queen in the centre and a unique lift, say (10,1,1), for the empty board. One would have to modify the lifts of H and CS such that they left everything fixed on the branches starting at 9 and 10, and would have to modify the lift of Q_9 to reset to 1 at levels 2 and 3. Since each has 4 lifts this would remove the need for 6 states in the emulating cascade product. It is easy to check that these lifts, together with either the α- or β-version of the lift of Q_1, would result in a cascade product emulating the Case 2 automaton, with respect to the restriction $X_\alpha^\uparrow$ or $X_\beta^\uparrow$, respectively, to the domain of 34 states.

Nevertheless, given the state of the art, there are several issues arising when one seeks to look for better representations using holonomy decomposition. Among these is that current software implementations require the whole state set to be defined explicitly before the decomposition, which in the case of how we formalized the 3-Queens puzzle meant that all legal configurations including solutions had to be found before the structure of the problem was analysed, although abstractly a mathematical description of these states would not in principle require any such enumeration. This may put in question the usefulness of computational holonomy decomposition as a problem-solving technique; however, even in this case such a decomposition might prove useful at the moment of implementation (or even specification) of software, since it suggests what structures and representations should be used. Moreover, it is by no means necessarily the case that one would have to find the entire legal configuration space in advance, e.g. one could rely either on a machine enumeration or include all possible configurations, legal or not, to avoid this.

Another potentially problematic issue is related to the potentially exponential complexity of the image set calculation, as it is never advisable to increase state sets in such large proportion. The major computational cost in carrying out a holonomy decomposition in the absence of special restrictions is determining the image sets of the automaton (or transformation semigroup). Lazy evaluation of the skeleton (including image sets), of state spaces, and of lifts could be used to mitigate these issues in the future, with more advanced versions of the decomposition software.

Given this experience, it seems likely that for the N-queens problem (also with N beyond where we can feasibly compute using SgpDec or computer algebra in general) similar interpretations of the coordinate systems should hold. For example, one could expect that the top-level state coordinates will reflect which moves were executed in the game to lead to the current configuration, with the most recent move captured by the top coordinate and possible positions for other queens captured by the lower coordinates. We might also conjecture that the model of N-queens from Case 1 (following Rule 1 without symmetries) results for all N in an aperiodic semigroup, i.e., a semigroup with no nontrivial permutations on any subset of states given by any sequence of moves.

Given the models we analysed, it seems clear too that including symmetries of the board among the generators will result in more tractable coordinate systems, involving fewer building-block components (probably just one per level), as compared to not including them, for all N.

Given this experience we can proceed to the computational investigation of the coordinatizations of the N-queens problem for larger N and, with the insights gained from the $N = 3$ case, mathematical investigation should allow us to verify (or possibly refute) the conjectures just mentioned for arbitrarily large board sizes N. In sum, not only do we have a hierarchical coordinate system for $N = 3$, but we have clear ideas to work from on what to expect of the coordinate systems for an N-queens problem of any board size.

The significance for BIOMICS of these findings lies in having established a closer correspondence between the algebraic coordinatization of the automaton we analysed and different possible interpretations for the algorithmic and "abstract state" significance of its coordinates. An example of the former is the past history interpretation of the second coordinate, whereas an example of the latter is the number of queens present on the board given by the third coordinate (when not yet determined by higher-coordinate, "more significant" values). Thus, generalization of the 3-queens problem to N queens will be pursued with particular attention paid to the recognition and interpretation of general computational properties of automata.

References

1. M A Arbib, editor. *Algebraic Theory of Machines, Languages, and Semigroups*. Academic Press, 1968.
2. P Cameron. *Introduction to Algebra (2nd Ed.)*. Oxford University Press, Oxford, 2008.
3. P Dini, C L Nehaniv, A Egri-Nagy, and M J Schilstra. Exploring the Concept of Interaction Computing through the Discrete Algebraic Analysis of the Belousov-Zhabotinsky Reaction. *BioSystems*, 112(2):145–162, 2013.
4. P Dömösi and C L Nehaniv. *Algebraic Theory of Automata Networks*. SIAM, Philadelphia, 2005.
5. A Egri-Nagy, P Dini, C L Nehaniv, and M J Schilstra. Transformation Semigroups as Constructive Dynamical Spaces. In Fernando A B Colugnati, Lia C R Lopes, and Saulo F A Barretto, editors, *Digital Ecosystems: Proceedings of the 3rd International Conference, OPAALS 2010*, pages 245–265, Aracaju, Sergipe, Brazil, 22-23 March, 2010. Springer LNICST.
6. A. Egri-Nagy and C. L. Nehaniv. Finite residue class rings of integers modulo n from the viewpoint of global semigroup theory. In J. M. AndrŐ, V. H. Fernandes, M. J. J. Branco, J. Fountain, G. M. S. Gomes, and J. C. Meakin, editors, *Proceedings of International Conference on Semigroups and Languages 2005 In Honour of the 65th birthday of Donald B. McAlister*, pages 66–83. World Scientific Press, 2007.
7. A Egri-Nagy and C L Nehaniv. Cascade Product of Permutation Groups. 2013. http://arxiv.org/abs/1303.0091.
8. A Egri-Nagy, C L Nehaniv, and J D Mitchell. SgpDec– Hierarchical Decompositions and Coordinate Systems, Version 0.7.25+. `(http://sgpdec.sf.net)`, 2013.
9. Samuel Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.
10. The GAP Group. Gap– *Groups, Algorithms, and Programming, Version 4.7*, 2013. http://www.gap-system.org.
11. K Krohn and J Rhodes. Algebraic Theory of Machines. I. Prime Decomposition Theorem for Finite Semigroups and Machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.
12. Gérard Lallement. *Semigroups and Combinatorial Applications*. Wiley, New York, 1979.
13. O Maler. On the Krohn-Rhodes Cascaded Decomposition Theorem. In Z Manna and D Peled, editors, *Time for Verification: Essays in Memory of Amir Pnueli*, Lecture Notes in Computer Science, 6200. Springer, Berlin, 2010.
14. C L Nehaniv. Algebraic Engineering of Understanding: Global Hierarchical Coordinates on Computation for the Manipulation of Data, Knowledge, and Process. In *Proceedings of the 18th Annual International Computer Software and Applications Conference (COMPSAC94)*, pages 418–425, Taipei, Taiwan, 1994. IEEE Computer Society Press.
15. J Rhodes. *Applications of Automata Theory and Algebra via the Mathematical Theory of Complexity to Biology, Physics, Psychology, Philosophy, and Games*. World Scientific Press, 2010. Foreword by M W Hirsch, edited by C L Nehaniv (Original version: University of California at Berkeley, Mathematics Library, 1969).
16. H P Zeiger. Cascade synthesis of finite-state machines. *Information and Control*, 10(4):419–433, 1967.