Generalized Dynamical Systems Part I: Foundations

Zargham, Michael; Shorish, Jamsheed

Published: 14/07/2022

Document Version
Publisher's PDF, also known as Version of record

GENERALIZED DYNAMICAL SYSTEMS PART I: FOUNDATIONS

A PREPRINT

Michael Zargham zargham@block.science

Jamsheed Shorish jamsheed@shorishresearch.com

July 11, 2022

ABSTRACT

In the first of three works we consider a generalized dynamical system (GDS) extended from that initially proposed by [25, 24], where a data structure is mapped to itself and the space of such mappings is closed under composition. We argue that GDS is the natural environment to consider questions arising from the computational implementation of autonomous and semi-autonomous decision problems with one or more constraints, nesting into one framework well-studied models of optimal control, system dynamics, agent-based modeling, and networks, among others. Particular attention is paid to mathematical constructions which support applications in mechanism design. The contingent derivative approach is defined, along with an associated metric, for which a GDS admits the study of existence of state trajectories that satisfy system constraints. The system may also be interpreted as a discretized version of a differential inclusion, allowing the characterization of the reachable subspaces of the state space, and locally controllable trajectories. The second and third parts in the three-part series are briefly described and cover, respectively, applications and implementations, with the latter demonstrating explicitly how a GDS can be implemented as software using Complex Adaptive Dynamics Computer Aided Design (cadCAD) [30].

Contents

1 Introduction

A dynamical system is simply a way of describing how one state evolves into another over time, and they are ubiquitous across a wide spectrum of disciplines. *Continuous time differential equations* were constructed in order explain observed natural phenomena such as the laws of motion from mechanics and electrodynamics; the have provided the foundation for applications as diverse as ecology and missile targeting [13]. The study of *nonlinear dynamical systems* provides insights into systems where small perturbations can cause large changes in observed phenomena [16]. *Dynamical systems embedded on networks* demonstrate the extent to which network typologies influence the possible trajectories of distributed processes [22]. *Discrete event* and *hybrid systems* demonstrate how non-trivial the dynamics of a system become when finite state machines are introduced and otherwise continuous systems are equipped with mechanisms for switching between different laws of motion [6]. *Differential games* and the associated *calculus of variations* provide insight into the entanglements between mechanisms and strategies of agents [14]. *Compositional games* developed using applied category theory offer additional machinery for evaluating interactions between mechanisms [11].

Ever-increasing entanglements between algorithmic and human decision-making invite the application of dynamical systems to complex networked dynamical systems, including the behavior of physical, digital, social and economic actors [26]. In order to design, develop, test, monitor and govern these systems it is necessary to extend our formal theory of dynamical systems beyond smooth manifolds to arbitrary data structures. State spaces must be made up of "dimensions" which are abstract classes, and points in these state spaces are instances of those classes for each "dimension" in the space. The dynamics of these systems can be any transformation from the state space to itself. While declaring such a construction is not difficult, recovering the mathematical machinery that makes dynamical systems useful for basic science and engineering is non-trivial.

In basic *scientific* settings, the most important property of a dynamical system is its capacity to represent an observed phenomena [23]. When the observed phenomena is represented by real-valued data points changing over (continuous) time, a system of differential equations becomes an extremely powerful modeling tool. However, when the observed data is categorical, structural, or is comprised of arbitrary data payloads emitted by a software system, a system of differential equations cannot always effectively represent the dynamics [7]. This gives the researcher the choice between adapting the research question to be consistent with modeling as a system of differential equations (perhaps bringing the question into conflict with the data), or investigating alternative representations (one such generalized representation is the focus of this work).

In *engineering* settings, dynamical systems are crucial for computer aided design as part of a broader practice of Model-Based Systems Engineering (MBSE) [8]. A representation of a system's state and its dynamics, as well as interfaces to external actors and environment, allow an engineer explore a design space in search of mechanisms suitable for achieving objectives while preserving desired properties [5]. Among the most critical theoretical methods is proving the existence and uniqueness of equilibria within that system. A global asymptotically stable equilibrium defined by a real-valued objective function represents a tendency for a system to advance its goal despite a range of unknown

circumstances [18]. Additionally, it is critical to ensure that trajectories of dynamical systems avoid unsafe regions of the state space, particularly when humans and their decisions are part of the system [17]. Determining which subspaces of the state space are reachable from which other points in the space, given the dynamics, can be used to monitor system health and design fail-safe mechanisms to ensure these system remain in their stable regimes.

In this paper, *Generalized Dynamical Systems Part I: Foundations*, we characterize *generalized dynamical systems* (GDS[1]), as formalized by e.g. Roxin ([25], [24]). This is a rich enough framework that it can represent the dynamical evolution of both real-valued and general data structures, and can act in both the basic scientific and engineering settings as outlined above. In our characterization, emphasis is placed on the notation and definitions (Section 2), and on conditions for representing dynamical systems within the GDS formalism from two conceptually related viewpoints based upon *contingent derivatives* (Section 3) and *differential inclusions* (Section 4). We elaborate on our motivations for developing GDS to further the practice of MBSE, and proceed to outline the forthcoming Parts II & III of the GDS series (Section 5). The second part, *Generalized Dynamical Systems Part II: Applications*, will demonstrate the use of the GDS formalism for three example systems drawn from different domains: an insurance contract, an automated market, and a governance contract. The final part, *Generalized Dynamical Systems Part III: Implementations*, will provide the formal specification for implementing GDS as computational models for use in the design, analysis and operation of complex systems.

2 GDS Notation and Definitions

A generalized dynamical system (GDS) is simply a pair $\{h, X\}$, where $h : X \to X$ is a state transition map and X is a state space, that may be influenced by factors that can change h. We set out the formal definitions in what follows, and provide a summary table (Table 1, cf. Appendix, p. 15) that may be leveraged as a quick-reference guide of this Section.

The *State Space* of a dynamical system is the representation of the information the system is describing in its evolution.

Definition 2.1. The **State Space** is the collection of all objects X that are sufficient to define a dynamical system.

In general, a State Space may be thought of as a natural environment for *data structures*. A selection from the State Space, $x \in X$, represents the state of the system at a particular point in time. Examples include a point in the phase space for a swinging pendulum, a particular graph realization from a dynamic graph, or a particular database record prior to updating. Note that although we adopt set-theoretic structures throughout (for notational conciseness and reader familiarity), general data structures and their manipulation may more readily be expressible using a *typed theory*, such as the typed lambda calculus (see e.g. [3]). This expressivity is particularly important for the computer implementation of a GDS, which is the subject of the third work in this series.

Definition 2.2. A **State** is an object $x \in X$ that represents the current *configuration* of the system.

If one were to observe the state evolution of a dynamical system, one would observe a *trajectory* or sequence of states.

[1] In what follows we refer to GDS in the singular *System* or in the plural *Systems* as context defines.

Definition 2.3. A **(State) Trajectory** is a sequence of States $x_0, x_1, \ldots$, each $x_i \in X$ for some index set $\mathcal{I}$.

The factors that can influence the dynamical system are considered as *inputs* that can affect the resulting state trajectory. Inputs may be drawn from an *input space*, which (like the state space) may be thought of as a data structure.

Definition 2.4. An **Input Space** is a collection U of all objects that can influence the State Trajectory of a dynamical system. An **Input** is an object $u \in U$ that represents the current input into the system.

Note that while conventional examples of an input include an "agent" or a "controller", who is actively using State information to select an Input (cf. 2.8 below), an Input may also include stochastic influences that impact the system. Thus the evolution of the dynamical system under *uncertainty* may be treated within the same framework as a deterministic dynamical system.

In general there may be conditions under which the state of a system at one point in time restricts the feasibility, or *admissibility*, of an Input u from the Input Space U. This restriction may be captured by defining an *admissible input space* that specifies possible Inputs given a state x.

Definition 2.5. An **Admissible Input Space** is a collection of objects $U_x \subseteq U$ from which in Input may be selected when the current configuration of the system is x. The **Admissible Input Map** $\mathcal{U} : X \to \wp(U)$ is a map which takes the State x and returns the Admissible Input Space U_x (which must be a member of the power set $\wp(U)$ of U, the Input Space). A member $u \in U_x$ is also called an **Admissible Input** (for the given State $x \in X$).

Given a State $x \in X$ and an Admissible Input Space $U_x = \mathcal{U}(x)$, there exists some entity which selects an Input $u \in U_x$. This entity may be a decision-making "agent" (which may be autonomous) or an external influence, such as an exogenous stochastic process. Together, the pair (x, u) set the conditions for *what configuration happens next* according the transition or *state update* map of the dynamical system.

Definition 2.6. A **State Update Map** is a map $f : X \times U_x \to X$, such that Image$(f) \subseteq X$.

The collection Image(f) in X indicates the set of all possible future configurations the system may find itself in. Note that in the GDS framework the state update map is thought of as a discrete transition (which may be an approximation of a continuous change, depending upon the model) that depends upon discrete or continuous values of the State x and the Input u. This is similar to discrete event-driven dynamical systems models, such as Petri Nets [20], and to composable morphism models based upon Applied Category Theory (ACT) [10]. The distinction between GDS and these alternative dynamical systems models depends upon the complexity of the data structure representing the state, as well as the ability of the framework to be implemented computationally (as shall be described in further detail in the third Part of this series). For example, stochastic Petri nets (cf. e.g. [1]) are well-suited for modeling changes in the *numbers* of things, but perhaps less well-suited to model changes in the *structures* of things. By contrast, ACT derives strong results from *composability* (as does, to a lesser extent, GDS), but can be difficult to implement in a computational environment.[2]

[2]Recent work [2] leveraging ACT and Stock-Flow modeling to create a software implementation lowers the barrier of ACT implementation, in much the same way that cadCAD lowers the barrier of GDS implementation.

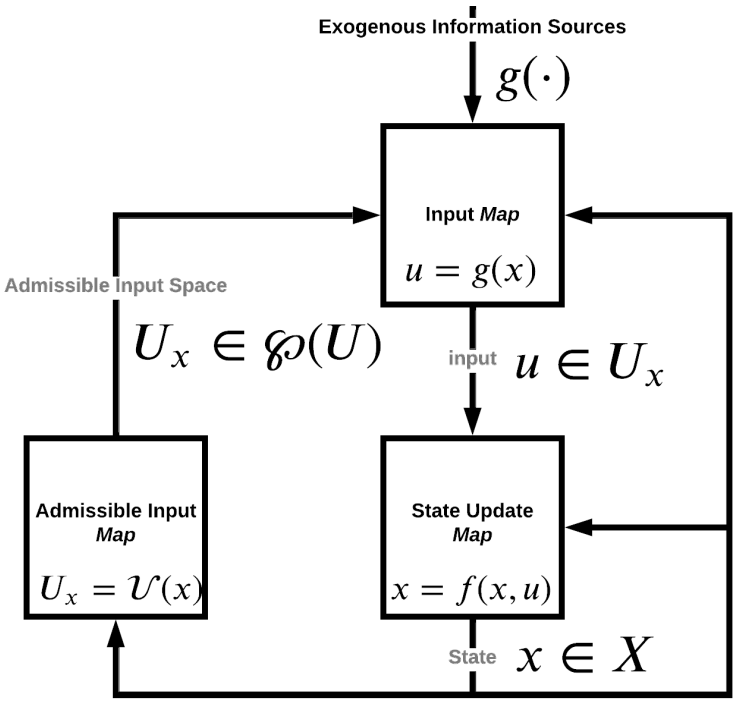

Figure 1: Block Diagram View of GDS Canonical Form

If the system is in a particular State $x \in X$, the State Update Map may be *restricted* to those future configurations that are consistent with the State x.

Definition 2.7. A **Restricted State Update Map** is a map $f|_x : U_x \to X$, such that Image$(f|_x) =$ Image$(f(x, \cdot))$,

where by Image$(f(x, \cdot))$ is meant the collection of values $\cup_{u \in U_x}\{f(x, u)\}$.

Restricting the state update map plays an important role in the design of mechanisms which have formally verifiable properties because it allows for declaration of differential (in)variants [21].

The method by which an Input $u \in U_x$, given a State $x \in X$, is selected is the *input map* that a decision-making agent or exogenous process possesses. This is conceptually distinct from the State Update Map, since the latter merely requires an Input *value* $u \in U_x$ and does not require a specification of how this value is selected.

Definition 2.8. An **Input Map** is a map $g : X \to U_x$ that selects an Input u from an Admissible Input Space U_x, given the current State $x \in X$.

When examining a State Trajectory over time, what is seen is the sequence of configurations $x_0, x_1, \ldots$ and what is hidden are the input selections generated from this sequence and from the associated admissible input spaces. The *state transition map* formalizes this observation of the sequence of configurations.

Definition 2.9. A **State Transition Map** is a map $h : X \to X$ given by $h(x) := f(x, g(x)) \equiv f|_x(g(x))$ such that $(x_0, x_1, x_2, \ldots) = (x_0, h(x_0), h \circ h(x_0), \ldots)$, with $x_t = \underbrace{h \circ \ldots \circ h}_{t \text{ times}}(x_0)$, is the State Trajectory.

Expanding this notional for clarity, the canonical form is given by:

$$u = g(x) \in U_x \tag{1}$$

$$x^+ = f(x, u) \in X \tag{2}$$

where $x \in X$ is the prior state, $u \in U_x$ is the realized input, and x^+ is the posterior state. A visual representation of this canonical form is provided in Figure 1. The GDS canonical form is chosen for its similarity to the canonical forms for linear and non-linear dynamical systems. Finally, we recap the start of this section by defining a Generalized Dynamical System, or GDS:

Definition 2.10. An Autonomous **Generalized Dynamical System** or **GDS** is a pair $\{h, X\}$, where h is a State Transition Map over a State Space X.

By collapsing the representation to $h(x) = f(x, g(x))$ the source of uncertainty g is endogenized to form an autonomous system. Although sources of uncertainty are expected to exist, inputs u are often unobservable, and in some cases entirely unknown. In many science and engineering applications the Input Map must be learned by observing trajectories.

For a quick reference guide on GDS notation and definitions, please see the Table in Appendix A (p. 15).

3 Contingent Representation of GDS

A GDS as defined in Section 2 is simply a pair $\{h, X\}$ that defines Trajectories of States within the State Space, but this obscures much of its formal representation. One such representation, the *contingent representation*, helps to answer the question, "Given a current State and a particular set of Admissible Inputs, what are all the possible States in the immediate future that the system can evolve to?" This is facilitated by the *attainability correspondence*, as defined in Roxin [24], the seminal work on GDS from the contingent representation. An attainability correspondence[3] F indicates, at any iteration (or "time") t, the possible States that are attainable for *any* sequence of admissible Inputs $u \in U$, for a given initial condition and an initial time. Formally,

Definition 3.1. Given initial State $x_0 \in X$ and initial time $t_0 \in \mathbb{R}_+$, the **attainability correspondence** $F : X \times \mathbb{R}_+ \times \mathbb{R}_+ \rightrightarrows X$ returns the subset of all States for which, at some time $t \geq t_0$, there exists a sequence $\{u_s\}_{s=t_0}^{t}$, each $u_s \in U_{x_s}$, such that

$$f(x_s, u_{s+1}) \in F(x_0, t_0, s+1),\ \forall t_0 \leq s \leq t.$$

[3]Note that this is a *set-valued* function—in the original work by Roxin this was called an 'attainability function' for this reason.

The attainability correspondence F rests upon both the State Update Map f and a sequence of Inputs $\{u_s\}$, providing a future State that can be reached from an initial State over time. This allows *reachability* to be introduced as criteria to help answer the question raised above: if a future State is in the attainability correspondence for some sequence of Inputs, then it is reachable. Reachability will be explored further in the differential inclusion representation of Section 4.

Recall from Table 1 that at each point in time s, if there is an Input Map g, then for a GDS

$$h(x_s) = f|_{x_s}(g(x_s)) = f(x_s, g(x_s)),$$
$$u_{s+1} = g(x_s),$$

and the attainability correspondence returns a *single* $x_{s+1} \in X$ from $h(x_s)$. A sequence of States $x_0, x_1, \ldots$ achieved in this way constitutes a *solution Trajectory under* g of the GDS. But in general the attainability correspondence encompasses *all* possible choices of Inputs, providing all possible Trajectories. This makes it difficult to infer, from a particular trajectory, whether or not a particular g function (or possibly *families* of g functions) gives rise to this Trajectory. More structure on the constraints imposed by the Input Map g is required.

3.1 The Contingent Derivative

To connect observed Trajectories with possible Input Maps requires formalizing how the State changes over time. It may be proven that, provided the attainability correspondence satisfies certain properties [cf. 24, 12], there is a natural definition of a *derivative* of a Trajectory that can help determine if the Trajectory is a solution Trajectory under an Input map g. This *contingent derivative* extends the idea of the derivative as a tangent line to a *set* of tangent lines, each satisfying the intuitive notion of a "rate of change" of a Trajectory, but for different convergent (sub)sequences of iterations ("times"). Defining the contingent derivative requires additional structure on X:

Assumption 3.2. The State Space X is a metric space, endowed with a metric $d_X : X \times X \to \mathbb{R}$.

Assumption 3.2 is needed for the expression of the contingent derivative (Definition 3.3 below), but is not especially restrictive when considering more general State and Input Spaces for a GDS. Indeed, it is possible in principle (but may be computationally delicate) to generalize to second-countable, completely regular Hausdorff topological spaces $(X, \mathcal{T})$ via the Urysohn Metrization Theorem (cf. [4], Theorem 9.10, p. 28) and using the metric that generates the topology $\mathcal{T}$ over X.

Definition 3.3 ([24, Defn. 6.1, p. 196])**.** A generalized contingent (or *Dini*) derivative, for initial conditions (x_0, t_0) is a vector v in a vector space $V(X)$ over X such that, given

1. a sequence of times $\{t_k\}_{k=1}^{\infty}$ converging to t_0, $t_0 < t_k \leq t,\ t_k \in \mathbb{R}\ \forall k$, and
2. a sequence $\{x_k\}_{k=1}^{\infty}$ converging to x_0, each $x_k \in F(x_0, t_0, t_k)$,

$$\lim_{k \to \infty} \frac{d_X(x_k, x_0)}{t_k - t_0} = v,$$

if the limit exists. The set of all contingent derivatives at (x_0, t_0) is denoted $D'F(x_0, t_0, t)$.

To ensure that the definition is not vacuous, structure is imposed on the sequence $\{x_k\}_{k=1}^{\infty}$ as a selection from F:

Assumption 3.4. The sequence $\{x_k\}_{k=1}^{\infty}$ satisfies a uniform Lipschitz property: there exists a constant $M < \infty$ such that $\forall k, m \in \mathbb{N}_0$,

$$d_X(x_k, x_m) \leq M d(t_k, t_m), \tag{3}$$

where $d(\cdot, \cdot)$ is the (Euclidean) metric over $\mathbb{R}$.

While the set of contingent derivatives provides the 'directions' under which, starting from (x_0, t_0), a solution Trajectory may be found, it does not *define* a solution Trajectory, as this requires the sequence of admissible Inputs $\{u_s\}_{s=1}^{t}$ as described in Definition 3.1. What is required is a restriction on the set of contingent derivatives $D'F(x_0, t_0, t)$ that the system obeys, in much the same way that, for functions, a differential equation's initial and boundary conditions set out the restrictions a solution function must obey.

The most straightforward restriction is to assume that the set of contingent derivatives $D'F(x_0, t_0, t)$ is a subset of a 'constraint set' in X (cf. [24], Equation 3.1, p. 191):

$$D'F(x_0, t_0, t) \subseteq C(x_0, t_0; g) \subset X, \tag{4}$$

where the set C depends upon the initial conditions (x_0, t_0) and is formed from the (possibly unknown) Input Map g. If the attainability correspondence F satisfies this constraint, then there exists at least one admissible input sequence $\{u_s\}_{s=1}^{t}$ for which the trajectory of States is a solution Trajectory under g.

The goal, then, is to derive the attainability correspondence $F(x_0, t_0, t)$ given the contingent equation 4, subject to the initial conditions (x_0, t_0). In Roxin's original work this goal recovers the GDS because the attainability correspondence was (under certain axioms) *defined* as the GDS. This is both more general and more restrictive than our needs. It is more general because building an axiomatic definition of a GDS as the attainability correspondence F abstracts away from the Admissible Input Map $\mathcal{U}$ and the Input Map g, and places instead conditions upon the constraint set $C(x, t)$ to guarantee existence (and possibly uniqueness) of F. (By contrast, the relationship between U, g and (by extension) h on the one hand, and the structure of $C(x, t)$ on the other, will restrict the possible attainability correspondences and hence those State Update Maps f that can obtain (in our definition) a GDS $\{h, X\}$.)

The axiomatic foundation of a GDS *as* the attainability correspondence is, however, also more restrictive, particularly when applied to problems in **optimal control**. This is because when applied to systems with an Input Space U and associated Input u, with the latter derived from a possibly unknown Input Map g, the contingent (or Dini) derivative places restrictions on the properties of the optimization problem that derives the contingent equation. It is particularly dependent upon *convexity* as a property of the optimization problem, whereas the general specification of a GDS as laid out in Table 1 has no such *a priori* requirement.

Thus, in what follows we briefly outline the conditions under which existence of a GDS as an attainability correspondence may be supported, bearing in mind that the underlying functional forms for e.g. the input map g must be stated in order to specify the structure of the constraint set C.

3.2 Existence of GDS Solutions

The structure of the constraint set C given in Assumption 3.5 below determines whether or not one can conclude that a sequence of admissible Inputs $\{u_s\}_{s=1}^{t}$ exists:

Assumption 3.5. The constraint set $C(x, t; g) \subset X$ is:

1. compact,
2. convex, and
3. continuous in (x, t).[4]

Theorem 3.6 (*Existence* [24, Theorem 5.1, p. 195 and Theorem 6.1, p. 198])**.** *If the constraint set satisfies Assumption 3.5 then the restriction*

$$D'F(x_0, t_0, t) = C(x_0, t_0)$$

is satisfied for some attainability correspondence F and some sequence of admissible Inputs $\{u_s\}_{s=1}^{t}$ for initial conditions (x_0, t_0).

4 Differential Inclusion Representation of GDS

The contingent derivative approach to GDS provides a way to understand, in very general terms, the conditions under which a solution Trajectory exists. To investigate stronger conclusions, such as which solution Trajectories may be *controlled*, requires stronger assumptions—but these assumptions also admit a useful second representation of GDS, the *differential inclusion* representation. This representation is a very natural generalization of differential equations. For example, instead of a State $x \in \mathbb{R}^n$ evolving in continuous time according to some function $f : \mathbb{R}^n \to \mathbb{R}^n$:

$$\dot{x}(t) = f(x(t)),$$

where as usual the notation $\dot{x}$ indicates the instantaneous rate of change of a variable x with respect to time, a differential inclusion relaxes this to the correspondence

$$\dot{x}(t) \in F(x(t))$$

for some *set* F that is dependent upon the State at time t, $x(t)$. Note the similarity between the inclusion set F and the constraint set $C(x_0, t_0; g)$ from the contingent derivative approach: the inclusion set F indicates which instantaneous changes to the State are 'admissible' given the constraints of the system, while the constraint set C indicates which

[4]The set $C(x, t; g)$ is continuous at a point (x_0, t_0) if, for any $\varepsilon > 0$, there exists a non-empty neighborhood η around (x_0, t_0) such that for all $(x, t) \in \eta$, $d_H(C(x, t; g), C(x_0, t_0; g)) < \varepsilon$, where d_H is the Hausdorff distance between sets.

contingent derivatives of the attainability correspondence are 'admissible'. Both approaches help address the question, "What may the system evolve to in the future?", but by virtue of its more restrictive structure on the State Space X (and, in particular, how a State changes over time) there are stronger conclusions that can be drawn than under the contingent derivative approach.

4.1 Reachability

We recast *attainability* in terms of the differential inclusion representation to provide a definition of *reachability* that is needed in what follows [28]. The State update map $f(x, u)$,[5] with $x \in X$ the current State and $u \in U_x$ an admissible input, is important here since any future State x^+ must be *reachable* from x:

Definition 4.1. The set of *immediately reachable States from* x, $R(x)$, is the set of all possible future States x^+ that may be achieved from an admissible input:

$$R(x) := \cup_{u \in U_x} \{f(x, u)\}.$$

Each immediately reachable State in $R(x)$ induces a 'rate of change' Δx from x via the metric associated with X:

$$\Delta x := d_X(x^+, x).$$

If we further define *event times* t and t^+ for x and x^+, respectively, then a time derivative analogue $\dot{x}(t)$ may be approximated by

$$\dot{x}(t) := \lim_{t^+ \to t} \frac{d_X(x^+(t^+), x(t))}{t^+ - t}, \tag{5}$$

if the limit exists, and the resulting differential inclusion is

$$\dot{x}(t) \in \partial R(x), \tag{6}$$

where by $\partial R(x)$ is meant the instantaneous feasible rates of change of the State as $d_X(x^+, x) \to 0$. If X is a normed space, this may also be associated with the (Clarke) tangent cone to $R(x)$ (cf. e.g. [27], Sec. 2.2).

4.2 The Configuration Space

In general we would like to know more than just whether or not a particular solution trajectory to a GDS exists (as discussed in Section 3.2) [9]. We would also like to know if, given particular point in the State Space $x \in X$, there exists a trajectory commencing from an initial condition x_0 and passing through x. Note that this depends upon the sequence of reachable State sets $\mathcal{R}_n(x_0) := (R(x_0), R(x_1), \ldots, R(x_{n-1})$, with $x = x_n \in R(x_{n-1})$ for some n.

Definition 4.2. The **Configuration Space** for a GDS is that set $X_C \subseteq X$ such that for each $x \in X_C$, there exists at least one initial condition $x_0 \in X$ and a sequence of some length n such that $x \in R(x_{n-1})$, where $R(x_{n-1})$ is generated from some $\mathcal{R}_n(x_0)$.

[5] Not to be confused with the 'generic' f function used in the expression of the differential equation $\dot{x}(t) = f(x(t))$ above.

Intuitively, the Configuration Space $X_C \subset X$ is a completely connected subspace of X, wherein it is possible for some sequence of admissible Inputs to take the system State from any point $x_0 \in X_C$ to any other point $x \in X_C$. Thus, all configurations $x \in X_C$ are mutually reachable.

4.3 Local Controllability

For a particular reachable State $x \in X$, the *x-controllability* set is the set of initial conditions $X_0 \subseteq X$ from which x is reachable (cf. [27], Ch 6), i.e. the set $X_0 := \{x_0 \in X \mid \exists n \in \mathbb{N} \ni \ x \in \mathcal{R}_n(x_0)\}$. Under the assumption that there exists a reachable point where (in the absence of e.g. exogenous uncertainty) a differential inclusion achieves an equilibrium, i.e. $\exists x \in X$ such that

$$0 \in \partial R(x),$$

sufficient conditions for x-controllability from a neighborhood η_x around x may be stated. That is, under such conditions there exists a trajectory starting from any $x_0 \in \eta_x$ that terminates at x. (Note that this implies that x is a fixed point of the differential inclusion defined by equation 6.) We require the following stronger assumption on the State Space X:

Assumption 4.3. The State Space is a subset of the n-dimensional Euclidean space $\mathbb{R}^n$.

Without loss of generality we translate the system so that $x = 0$, i.e. $0 \in \partial R(0)$. Then the following result holds:

Theorem 4.4 (*Local Controllability* [27, Theorem 6.3, p. 148])**.** *If the mapping $\partial R : \mathbb{R}^n \to \mathbb{R}^n$ is a Lipschitzian correspondence with closed, convex values, and there is a controllable closed, strictly convex process sufficiently reachable near $x = 0$, then the system is 0-controllable from a neighborhood η_0 around $x = 0$.*

4.4 Observability and Design

It should be noted that from a *design* point of view it may be that one or more derived "invariant" properties are desired along the GDS Trajectory [19]. That is, for a Trajectory $x_0, x_1, \ldots$, one would like the evaluation of a property `P` to be `TRUE` for one or more States:

$$\mathtt{P}(\mathtt{x_i}) = \mathtt{TRUE},$$

for some i.[6] If such properties are the driving force behind the design of the dynamical system, it may not be as important—or may not be possible—to observe the underlying States x_i.

Thus, controllability may be cast within the context of *observability*, in that rather than specify x-controllability, i.e. the initial conditions such that a State x is observed to be reachable, one instead attempts to design the system for `P`-controllability, i.e. such that the set of initial conditions 'guarantees' that observable property `P` is satisfied for 'almost all' realized Trajectories of the system.

[6] Note that `P` may be a set of properties, in which case a vector may be returned with each element the evaluation of each property, or a single value may be returned, if e.g. all properties are evaluated to `TRUE`.

5 Next Steps

The development and refinement of GDS is rooted in the authors' practice of Model-Based Systems Engineering (MBSE) applied to systems containing algorithmic policies or automated contracts, both of which involve automation coupled with human behavior. These types of systems are prone to unintended system properties and MBSE provides tools for designing and governing these systems. GDS in turn enables MBSE because MBSE requires data-driven models of existing systems, and computer computational models of to-be systems. MBSE is characterized by systems of systems, each represented by loosely-coupled models of subsystems and components.

Systems involving automated contracts (including but not limited to blockchain based *smart contracts*) are collective scale human-machine assemblages, comprised of interacting subsystems which vary significantly in even their most basic characteristics, including but not limited to physical plants, software systems and human behavior. Dynamical systems defined over real-valued fields cannot adequately represent all aspects of these systems.

In CyberPhysical Systems, hybrid systems models address much of the resulting 'representation gap'. But social and economic systems possess structural and functional elements which evolve in a state space too rich even for hybrid systems models. For example, a social or economic network is characterized by a directed graph, with node and edge sets that are subject to change. In many cases, moreover, each node has its own instance of an abstract class of behavioral models, with its own observable and controllable states as well as its own estimation and decision methods.

The GDS framework allows constructs such as these 'agent network' states to be composed with states representing a designed system plant and its environment within one state space. As high dimensional as that space is, the GDS also encodes the network formation process, the revision rules for agent strategies, plant dynamics and exogenous processes driving the system. Among the most interesting examples of GDS are those representing socio-technical systems, where the behaviors of agents are predicated on their observations of each other and the plant dynamics, which contains state-feedback mechanisms steering toward a design goal [29].

A common area of interest for modeling these socio-technical systems is *Reinforcement Learning* (RL) [15]. While RL agents perform well in environments where the field of action is relatively static, and the agents can perform fictitious play in order to learn viable strategies, this is rarely the case for open systems such as political and economic games. By focusing on admissible action sets and accounting directly for the system's laws of motion, GDS makes it easier to bring RL techniques into both policy making and agent-strategy automation contexts.

The application of GDS to the above contexts is facilitated by 1) demonstrating the formalisms in this work with the help of specific examples, and by 2) providing a computer implementation framework for GDS. These are the focus of the succeeding works in this sequence and are briefly described below.

5.1 Applications

The next paper, *Generalized Dynamical Systems Part II: Applications*, focuses on applying GDS notation to example systems which already exist, and proceeds to demonstrate what can be gained in design, operations and governance work from a GDS model.

The first example is an automated insurance contract. Its core policies include the triggering conditions under which an insuree is due payment, and the laws of motion describing the conservation flow of the monetary resource. Assuming the insurer and a class of insurees have different access to capital and time preferences of money, there exists a set of circumstances under which an insurance arrangement has a positive sum outcome.

The second example is an automated market, specifically, a Balancer Pool with exactly two assets and even weights.[7] Its two core mechanisms are (i) the ability to add or remove liquidity, and (ii) the ability to swap one asset for the other. It can be shown that the laws of motion for the market preserve a *differential invariant*, that constrains the set of possible trajectories in a manner that reveals price preference.

The third example is a governance structure. Its core policies include a *collection* of other policies, which are parameterized. The governance structure allows proposals to change the value of one or more parameters, and those parameter changes are enacted if and only if a voting mechanism passes the proposal. The GDS framework provides a clear-cut means for defining the set of all possible reachable states, by considering the reachable states for all possible parameterizations of the policies being governed.

5.2 Implementations

The final paper, *Generalized Dynamical Systems Part III: Implementations*, focuses on a framework specifically designed for the implementation of a GDS: the Complex Adaptive Dynamics Computer Aided Design (cadCAD) framework.

The legacy version of cadCAD has the semantics of a repeated stage game: at each stage there are actions taken in parallel followed by a state update, computed by applying the laws of motion for the active mechanism(s) during the stage. As part of the GDS research, a new *cadCAD reference implementation* is being to developed with semantics which precisely mirror those of GDS. This streamlines the transition from mathematical specification to software.

Implementations are characterized by an initial instance of the state object and a sequence of transformations, each of which contains an optional input stage for the Input Map $u = g(x)$ logic, and a mechanism stage for the State Update Map $x^+ = f(x, u)$. While many software frameworks support similar iterative processes, cadCAD is differentiated by its emphasis on scientific controlled experiments. Each simulation configuration contains parameters which can be scalars, but may also be methods, or even classes. The richness of the parameter space, combined with the richness of the state space and state update maps allows cadCAD users to follow computer aided design workflows for a broad class of GDS problems.

[7] This market structure is equivalent to Uniswap V1/V2, but not to V3, the latest version at the time of this writing.

A Appendix: Notation and Definitions

Basic Definitions			
Term	**Notation**	**Definition**	**Relations**
State	x	an object or point representing the current configuration of the system	
State Space	X	a data structure or space containing all possible values of a State x	$x \in X$
Input	u	an object representing an input to the system	
Input Space	U	a data structure or space containing all possible values of an Input u	$u \in U$
Admissible Input Space	U_x	the set containing all reachable Input values of u *given* the current State $x \in X$	$u \in U_x \subseteq U$
Admissible Input Map	$\mathcal{U}$	a map which takes the current State x and returns the Admissible Input Space U_x	$\mathcal{U} : X \to \wp(U)^\dagger$
State Update Map	f	a map which takes the current State $x \in X$ and an admissible Input $u \in U_x$ to return a new State in X.	$f : X \times U_x \to X$, Image$(f) \subseteq X$
Restricted State Update Map	$f\|_x$	a State Update Map f *fixed* at a State $x \in X$, taking an admissible Input $u \in U_x$ and returning a new State in X.	$f\|_x : U \to X$
Input Map	g	a map which takes the current State $x \in X$ and returns an admissible Input $u \in U_x$	$g : X \to U_x$
State Transition Map	h	a map which takes the current State $x \in X$ to a point in the same State Space X	$h = f\|_x \circ g$, $h : X \to X$
Posterior State	x^+	the State the system arrives at after applying the State Transition Map h to State $x \in X$	$x^+ = h(x)$
Trajectory	$x_0, x_1 \ldots$	A sequence of States in X achieved by recursively applying the State Transition Map h starting at initial State x_0	$x_t = \underbrace{h \circ \ldots \circ h}_{t \text{ times}}(x_0)$
Generalized Dynamical System	$\{h, X\}$	A State Transition Map h and an associated State Space X such that $h : X \to X$ generates Trajectories for any initial State x_0	$x_1 = h(x_0)$, $x_2 = h(x_1), \ldots$, $x_t = h(x_{t-1}), \ldots$

†The notation $\wp(U)$ denotes the power set of the set U.

Table 1: Quick Reference: Notation and Definitions

References

[1] John Baez and Jacob Biamonte. *Quantum Techniques for Stochastic Mechanics*. 2012. DOI: 10.48550/ARXIV.1209.3632.

[2] John Baez et al. *Compositional Modeling with Stock and Flow Diagrams*. 2022. DOI: 10.48550/ARXIV.2205.08373.

[3] H. Barendregt et al. *Lambda Calculus with Types (Perspectives in Logic)*. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139032636.

[4] Glen E. Bredon. *Topology and Geometry*. Springer New York, 1993.

[5] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[6] Christos G Cassandras, Stephane Lafortune, et al. *Introduction to discrete event systems*. Vol. 2. Springer, 2008.

[7] Sarah Day, Robertus C.A.M. Vandervorst, and Thomas Wanner. "Topology in Dynamics, Differential Equations, and Data". In: *Physica D: Nonlinear Phenomena* 334 (2016). Topology in Dynamics, Differential Equations, and Data, pp. 1–3. ISSN: 0167-2789. DOI: https://doi.org/10.1016/j.physd.2016.08.003.

[8] Jeff A Estefan et al. "Survey of model-based systems engineering (MBSE) methodologies". In: *Incose MBSE Focus Group* 25.8 (2007), pp. 1–12.

[9] Michael Farber and Mark Grant. "Topological complexity of configuration spaces". In: *Proceedings of the American Mathematical Society* 137.5 (2009), pp. 1841–1847.

[10] Brendan Fong and David I Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. 2018. DOI: 10.48550/ARXIV.1803.05316.

[11] Neil Ghani and Jules Hedges. "A compositional approach to economic game theory". In: *CoRR* abs/1603.04641 (2016). arXiv: 1603.04641. URL: http://arxiv.org/abs/1603.04641.

[12] S. Gutman. "Uncertain dynamical systems–A Lyapunov min-max approach". In: *IEEE Transactions on Automatic Control* 24.3 (1979), pp. 437–443.

[13] Philip Holmes. "A Short History of Dynamical Systems Theory: 1885". In: *History of Mathematics* (2010), p. 115.

[14] Rufus Isaacs. *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation, 1965.

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[16] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[17] Nancy G Leveson. *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[18] Aleksandr Mikhailovich Lyapunov. "The general problem of the stability of motion". In: *International journal of control* 55.3 (1992), pp. 531–534.

[19] Peter J Olver. *Equivalence, invariants and symmetry*. Cambridge University Press, 1995.

[20] C. Adam Petri and W. Reisig. "Petri net". In: *Scholarpedia* 3.4 (2008), p. 6477. DOI: 10.4249/scholarpedia.6477.

[21] André Platzer. "Differential Dynamic Logics: Automated Theorem Proving for Hybrid Systems". PhD thesis. Department of Computing Science, University of Oldenburg, 2008, p. 299.

[22] Mason A Porter and James P Gleeson. "Dynamical systems on networks". In: *Frontiers in Applied Dynamical Systems: Reviews and Tutorials* 4 (2016).

[23] Alfio Quarteroni. "Mathematical models in science and engineering". In: *Notices of the AMS* 56.1 (2009), pp. 10–19.

[24] Emilio Roxin. "On Generalized Dynamical Systems Defined by Contingent Equations". In: *Journal of Differential Equations* 1.2 (1965), pp. 188–205.

[25] Emilio Roxin. "Stability in general control systems". In: *Journal of Differential Equations* 1.2 (1965), pp. 115–150.

[26] Hiroki Sayama. *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks, 2015.

[27] Georgi V. Smirnov. *Introduction to the Theory of Differential Inclusions*. American Mathematical Society, 2002.

[28] Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Vol. 6. Springer Science & Business Media, 2013.

[29] Shermin Voshmgir and Michael Zargham. "Foundations of cryptoeconomic systems". In: *WU Vienna University of Economics and Business:https://epub.wu.ac.at/7782* (2020).

[30] M. Zargham and E Lima. *cadCAD Formal Spec*. https://github.com/cadCAD-org/cadCAD/blob/master/documentation/spec.pdf. May 2021.