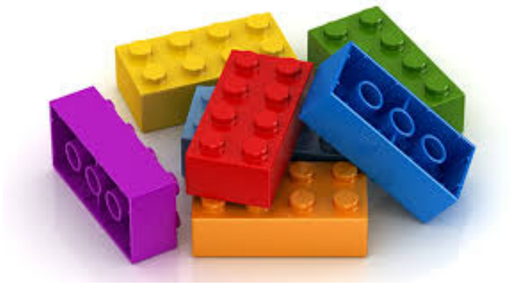
Lecture 2: Functors

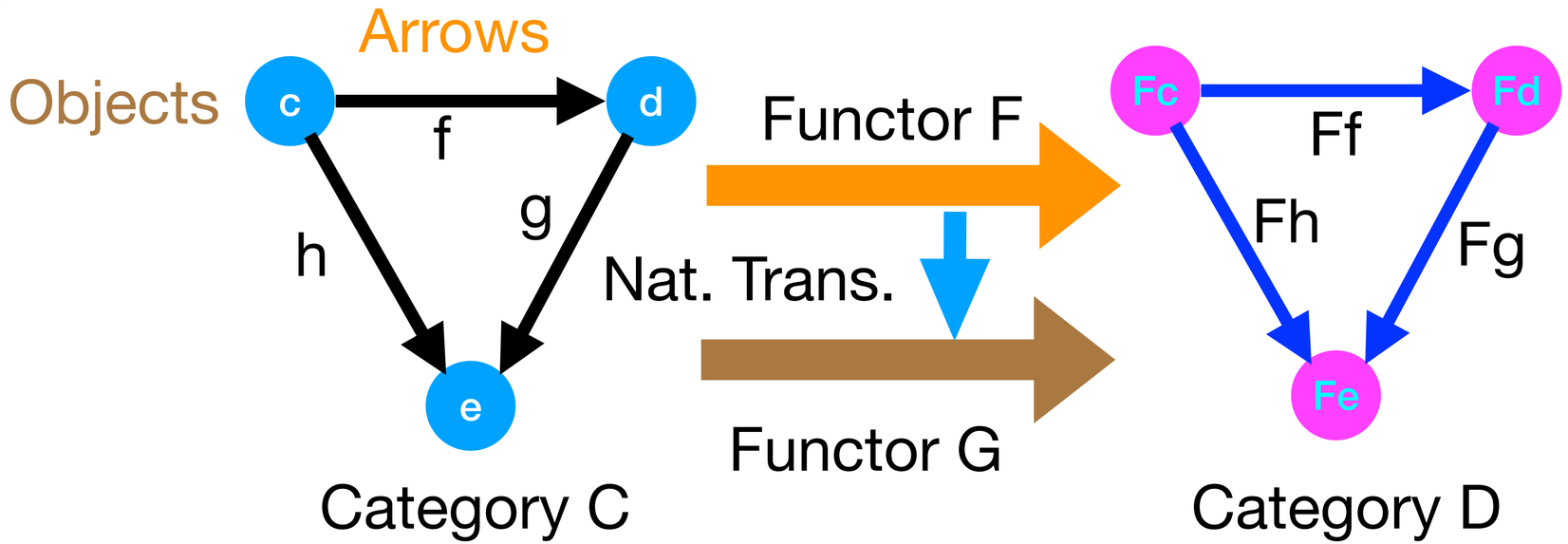
Sridhar Mahadevan, Adobe Research and U.Mass, Amherst

High level structure of the course

- Weeks 1 through 7:
  - The Yoneda view of AGI: behavior is all that matters
  - Representable functors C(-, x) captures everything about object x
  - LLMs personify the Yoneda viewpoint: attention is all you need!
- Weeks 8 through 14:
  - The topos view of AGI: consciousness, logic and reasoning is AGI
  - Agents cannot just act, they need to be self-aware of their actions
  - From chatGPT to MUMBLE-GPT: a new framework for AGI



The Building Blocks



Functor Types

Directed graphs specified as functors

Contravariant

Any particular graph is a functor F: C^{op} —> Set

Covariant

Any particular graph is a functor F: C —> Set

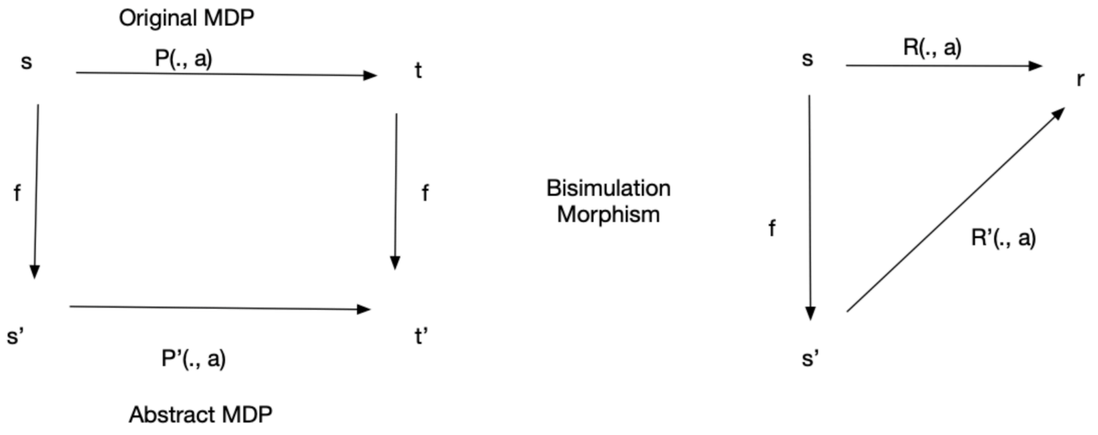
RL as a Category

Objects: Markov decision processes

Arrows: MDP homomorphisms

Objects are MDPs: $\langle S, A, \Psi, P, R \rangle$

- S is a discrete set of states
- A is the discrete set of actions
- $\Psi \subset S \times A$ is the set of admissible state-action pairs
- $P : \Psi \times S \to [0, 1]$ is the transition probability function specifying the one-step dynamics of the model
- $R : \Psi \to \mathbb{R}$ is the expected reward function



MDP homomorphism

Homomorphisms are studied across all of math (groups, graphs, rings, topological spaces)

An MDP homomorphism from MDP $M = \langle S, A, \Psi, P, R \rangle$ to $M' = \langle S', A', \Psi', P', R' \rangle$, denoted $h : M \twoheadrightarrow M'$, is defined by

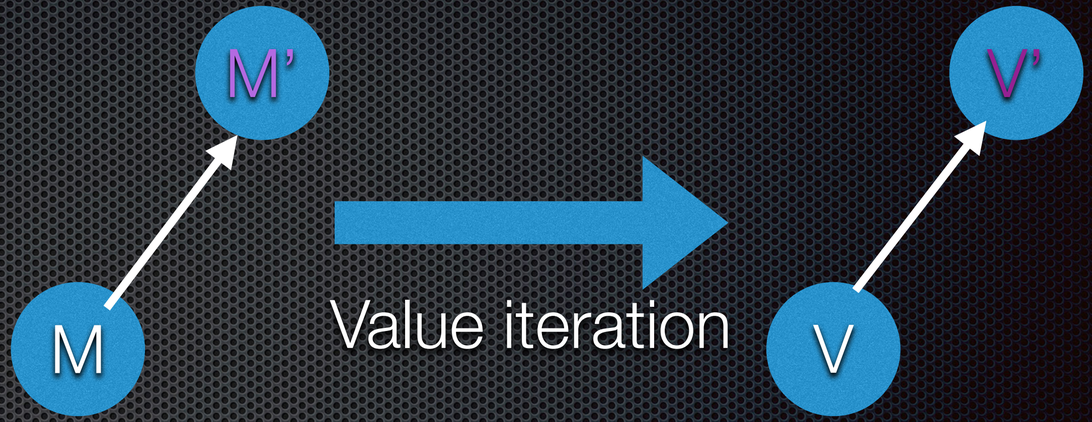
- A tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$
- where $f : S \twoheadrightarrow S'$, $g_s : A_s \twoheadrightarrow A'_{f(s)}$
- $h((s, a)) = \langle f(s), g_s(a) \rangle$, for $s \in S$
- Stochastic substitution property and reward respecting properties below are respected:

$$P'(f(s), g_s(a), f(s')) = \sum_{s'' \in [s']_f} P(s, a, s'') \quad (1)$$

$$R'(f(s), g_s(a)) = R(s, a) \quad (2)$$

RL algorithms are functors

They map a category of MDPs into a category of value functions



PSR Category

Objects: Predictive state representations

Arrows: PSR homomorphisms

PSR (and earlier models, like multiplicity automata, observer operator models etc.) form categories:

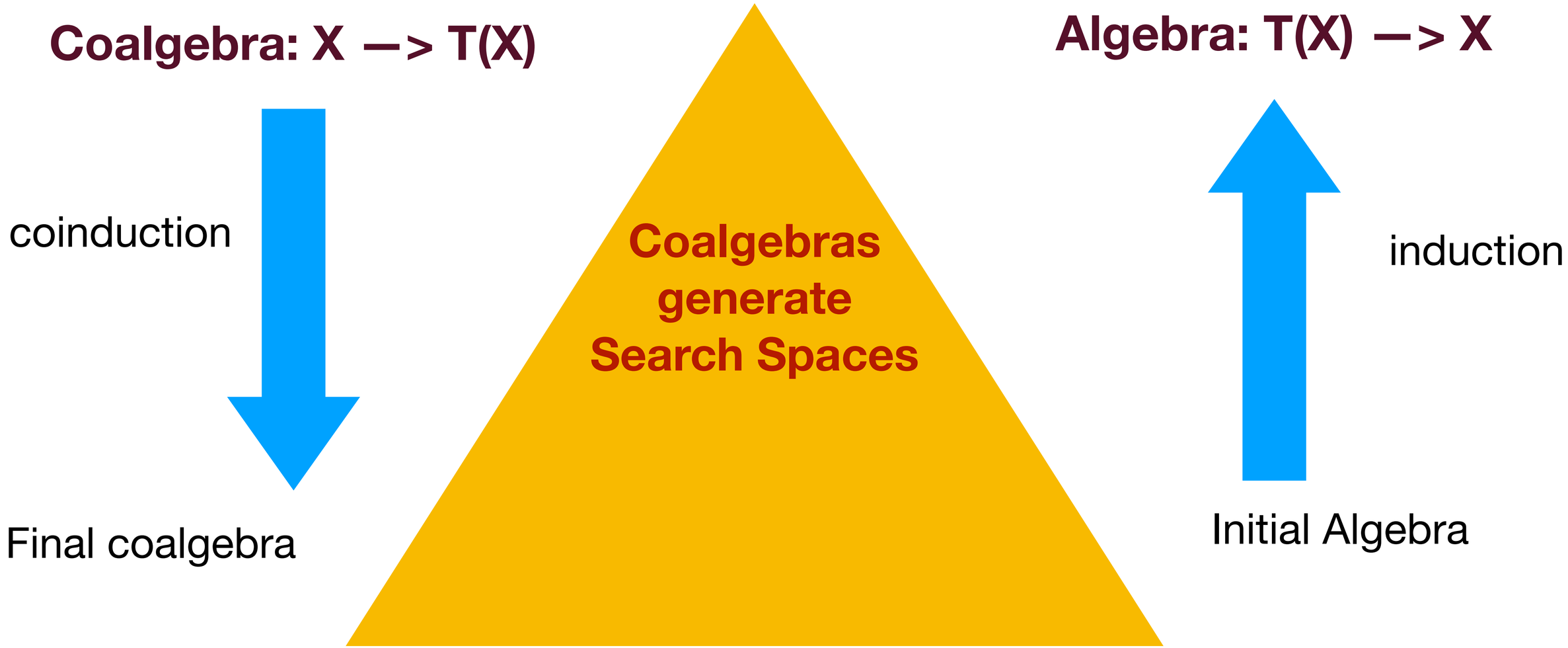
- Finite set of actions A and observations O.
- A *history*: sequence of actions and observations $h = a_1 o_1 \dots a_k o_k$.
- A *test*: possible sequence of future actions and observations $t = a_1 o_1 \dots a_n o_n$.
- $P(t|h)$ is a prediction test t will succeed from history h.
- State ψ: a vector of predictions of *core tests* $\{q_1, \dots, q_k\}$.
- The prediction vector $\psi_h = \langle P(q_1|h) \dots P(q_k|h) \rangle$ is a sufficient statistic. The entire predictive state of a PSR can be denoted Ψ.

A **PSR homomorphism** from a PSR Ψ to another PSR Ψ' is defined as:

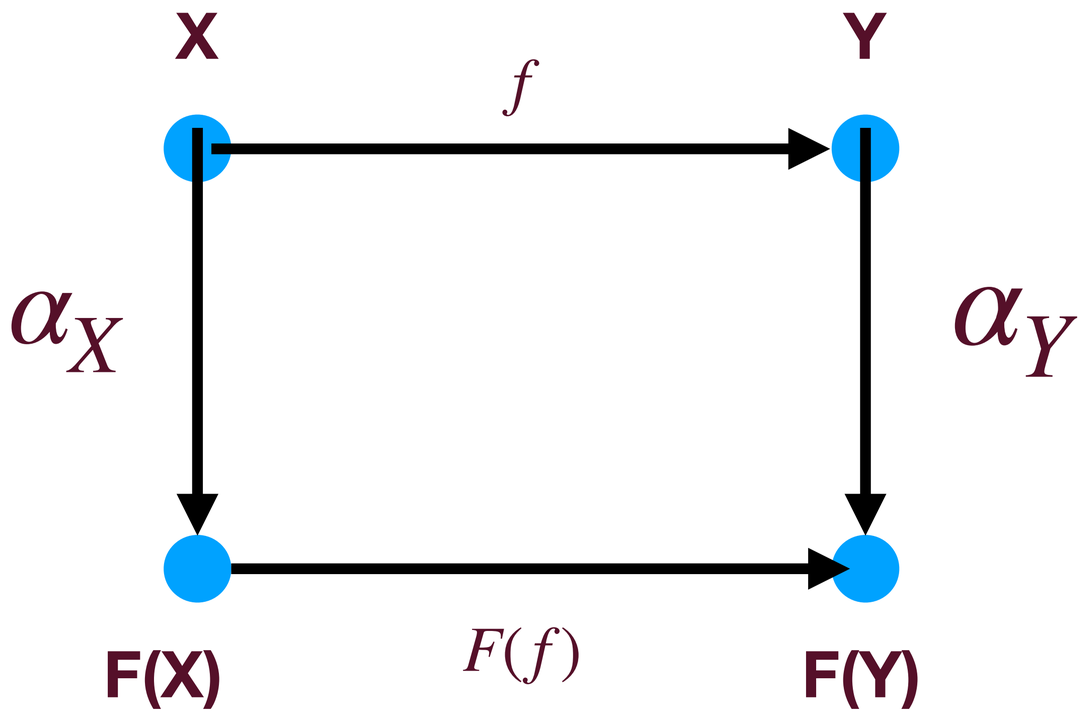
- A tuple of surjections $\langle f, v_\psi(a) \rangle$
- where $f : \Psi \to \Psi'$ and $v_\psi : A \to A'$ for all prediction vectors $\psi \in \Psi$
- such that

$$P(\psi'|f(\psi), v_\psi(a)) = P(f^{-1}(\psi')|\psi, a) \tag{3}$$

for all $\psi' \in \Psi, \psi \in \Psi, a \in A$.


Coalgebra: X —> T(X)
coinduction
Final coalgebra
Coalgebras
generate
Search Spaces
Algebra: T(X) —> X
induction
Initial Algebra

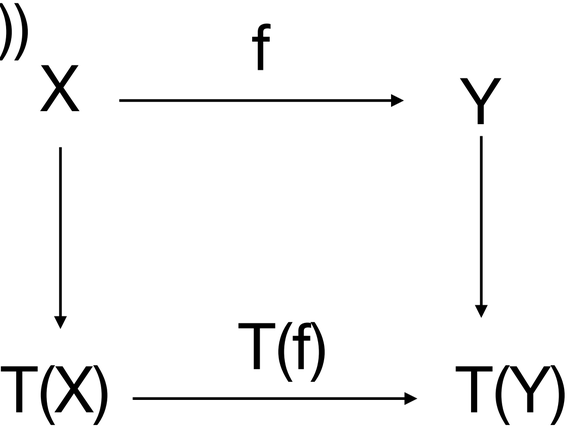
Arrows between Coalgebras

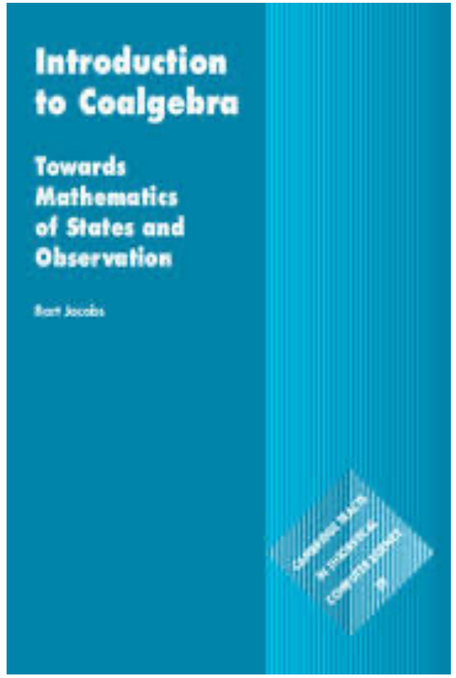


MDP/PSR homomorphisms are a special case of this framework

Category of Coalgebras

- Category of Coalgebras:
  - Objects: Coalgebras (X, a: X —> T(X))
  - Arrows: Coalgebra homomorphisms
- Initial and Final Coalgebras
  - Initial object: unique morphism into other objects
  - Final object: unique morphism from other objects
- Generalizes the concept of fixed points!





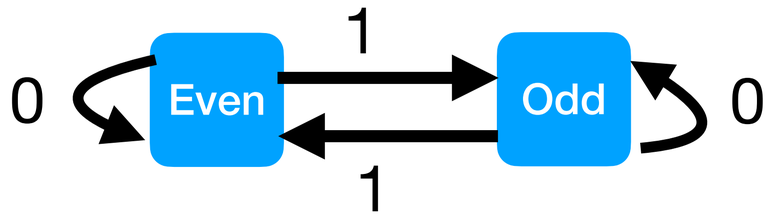
The PowerSet Functor

- Consider the coalgebra: X —> PowerSet(X)
  - Example: X = {0,1}, PowerSet = {{}, {0}, {1}, {0,1}}
- Why is PowerSet a functor?
  - Objects: Sets, like X
  - Arrows: functions on sets f:X —> Y
- How does PowerSet act on the arrows?
  - On arrows: Powerset(f): PowerSet(X) —> PowerSet(Y)

Category theory gives us a new way to define states!

(X, X —> T(X)): Coalgebra

$$(Q, \alpha : Q \mapsto (Q^A \times B))$$



$$(Q, A, B, \delta : Q \times A \to Q, \beta : Q \to B)$$

Hard attention Transformers cannot compute parity!

[Hahn, ACL, 2020]

ELSEVIER

Theoretical Computer Science 249 (2000) 3–80

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fundamental Study

Universal coalgebra: a theory of systems

J.J.M.M. Rutten

CWI, P.O. Box 94079, 1090 GB Amsterdam, Netherlands

Communicated by M.W. Mislove

Abstract

In the semantics of programming, finite data types such as finite lists, have traditionally been modelled by initial algebras. Later final *coalgebras* were used in order to deal with *infinite* data types. Coalgebras, which are the dual of algebras, turned out to be suited, moreover, as models for certain types of automata and more generally, for (transition and dynamical) *systems*. An important property of initial algebras is that they satisfy the familiar principle of induction. Such a principle was missing for coalgebras until the work of Aczel (Non-Well-Founded sets, CSLI Leethre Notes, Vol. 14, center for the study of Languages and information, Stanford, 1988) on a theory of non-wellfounded sets, in which he introduced a proof principle nowadays called *coinduction*. It was formulated in terms of *bisimulation*, a notion originally stemming from the world of concurrent programming languages. Using the notion of *coalgebra homomorphism*, the definition of bisimulation on coalgebras can be shown to be formally dual to that of congruence on algebras. Thus, the three basic notions of universal algebra: algebra, homomorphism of algebras, and congruence, turn out to correspond to coalgebra, homomorphism of coalgebras, and bisimulation, respectively. In this paper, the latter are taken as the basic ingredients of a theory called *universal coalgebra*. Some standard results from universal algebra are reformulated (using the aforementioned correspondence) and proved for a large class of coalgebras, leading to a series of results on, e.g., the lattices of subcoalgebras and bisimulations, simple coalgebras and coinduction, and a covariety theorem for coalgebras similar to Birkhoff's variety theorem.

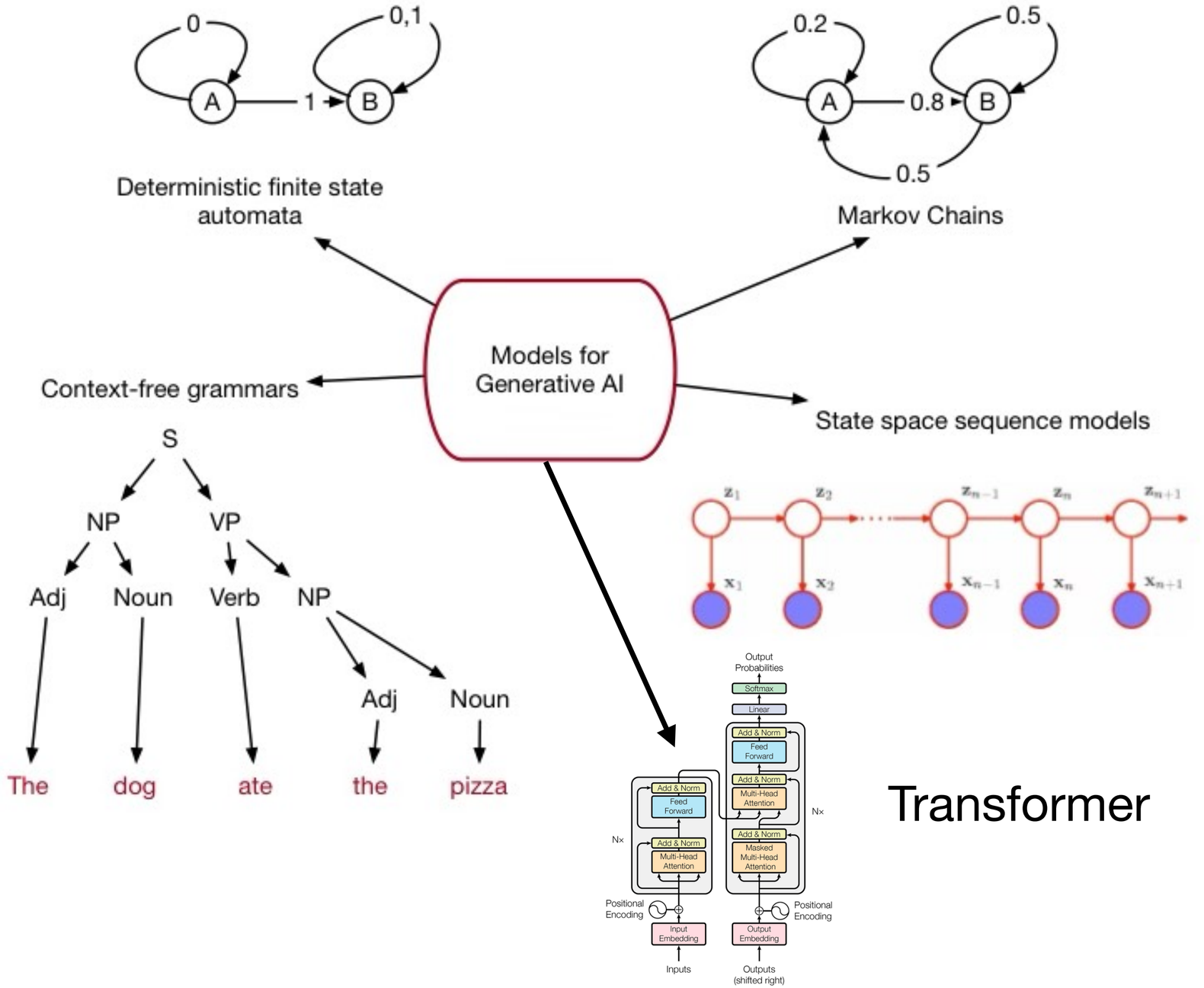
MSC: 68Q10; 68Q55

PACS: D.3; F.1; F.3

Keywords: Coalgebra; Algebra; Dynamical system; Transition system; Bisimulation; Universal coalgebra; Universal algebra; Congruence; Homomorphism; Induction; Coinduction; Variety; Covariety

E-mail address: janr@cwi.nl (J.J.M.M. Rutten).

PII: S0304-3975(00)00056-6



Generative models as

Universal coalgebras

Deep Learning as a Functor

Backprop as Functor: A compositional perspective on supervised learning

Brendan Fong David Spivak Rémy Tuyéras

Department of Mathematics, Massachusetts Institute of Technology

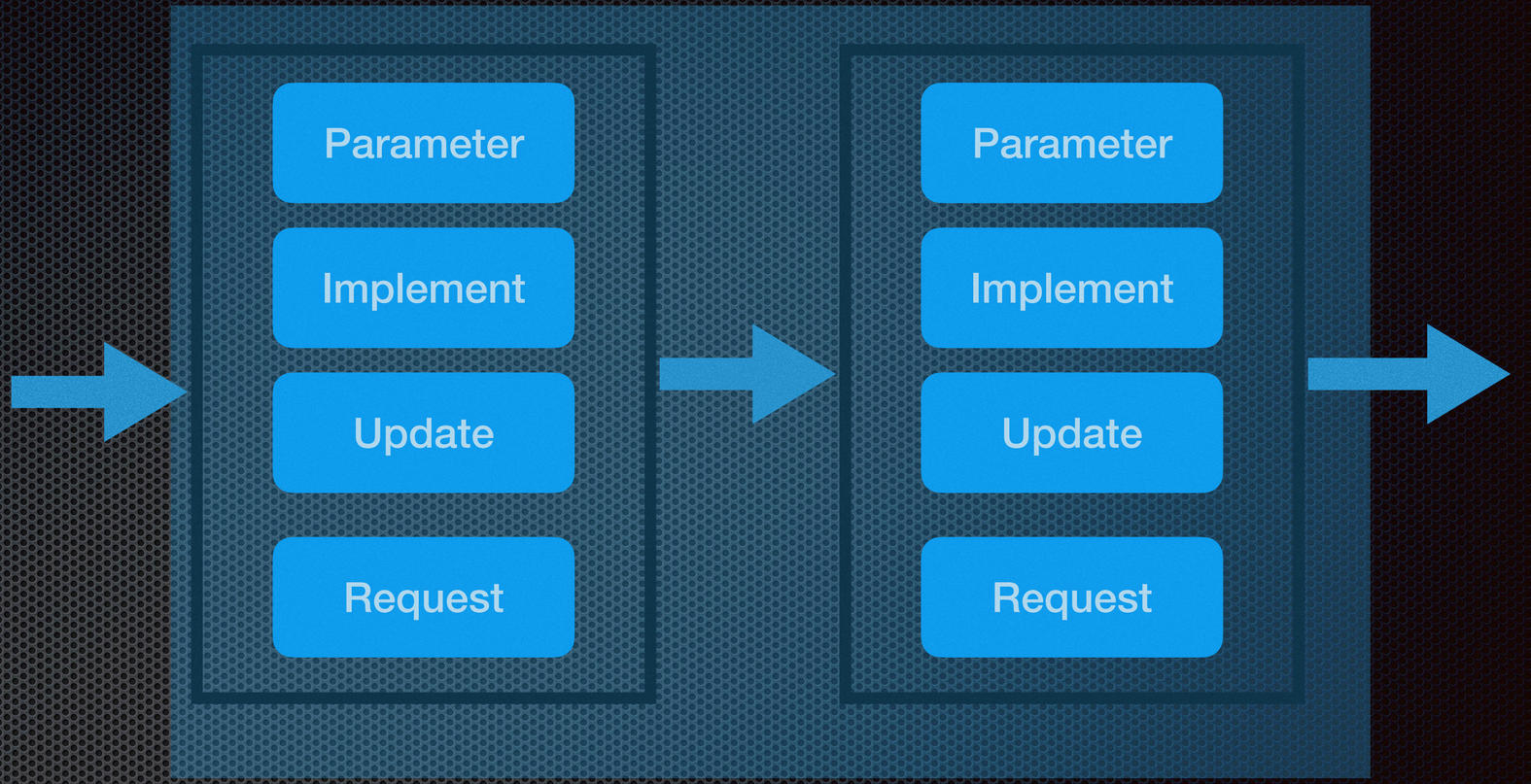
Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology

***Abstract*—A supervised learning algorithm searches over a set of functions $A \to B$ parametrised by a space P to find the best approximation to some ideal function $f\colon A \to B$. It does this by taking examples $(a, f(a)) \in A \times B$, and updating the parameter according to some rule. We define a category where these update rules may be composed, and show that gradient descent—with respect to a fixed step size and an error function satisfying a certain property—defines a monoidal functor from a category of parametrised functions to this category of update rules. A key contribution is the notion of request function. This provides a structural perspective on backpropagation, giving a broad generalisation of neural networks and linking it with structures from bidirectional programming and open games.**

Consider a supervised learning algorithm. The goal of a supervised learning algorithm is to find a suitable approximation to a function $f\colon A \to B$. To do so, the supervisor provides a list of pairs $(a, b) \in A \times B$, each of which is supposed to approximate the values taken by f, i.e. $b \approx f(a)$. The supervisor also defines a space of functions over which the learning algorithm will search. This is formalised by choosing a set P and a function $I\colon P \times A \to B$. We denote the function at parameter $p \in P$ as $I(p, -)\colon A \to B$. Then, given a pair $(a, b) \in A \times B$, the learning algorithm takes a current hypothetical approximation of f, say given by $I(p, -)$,

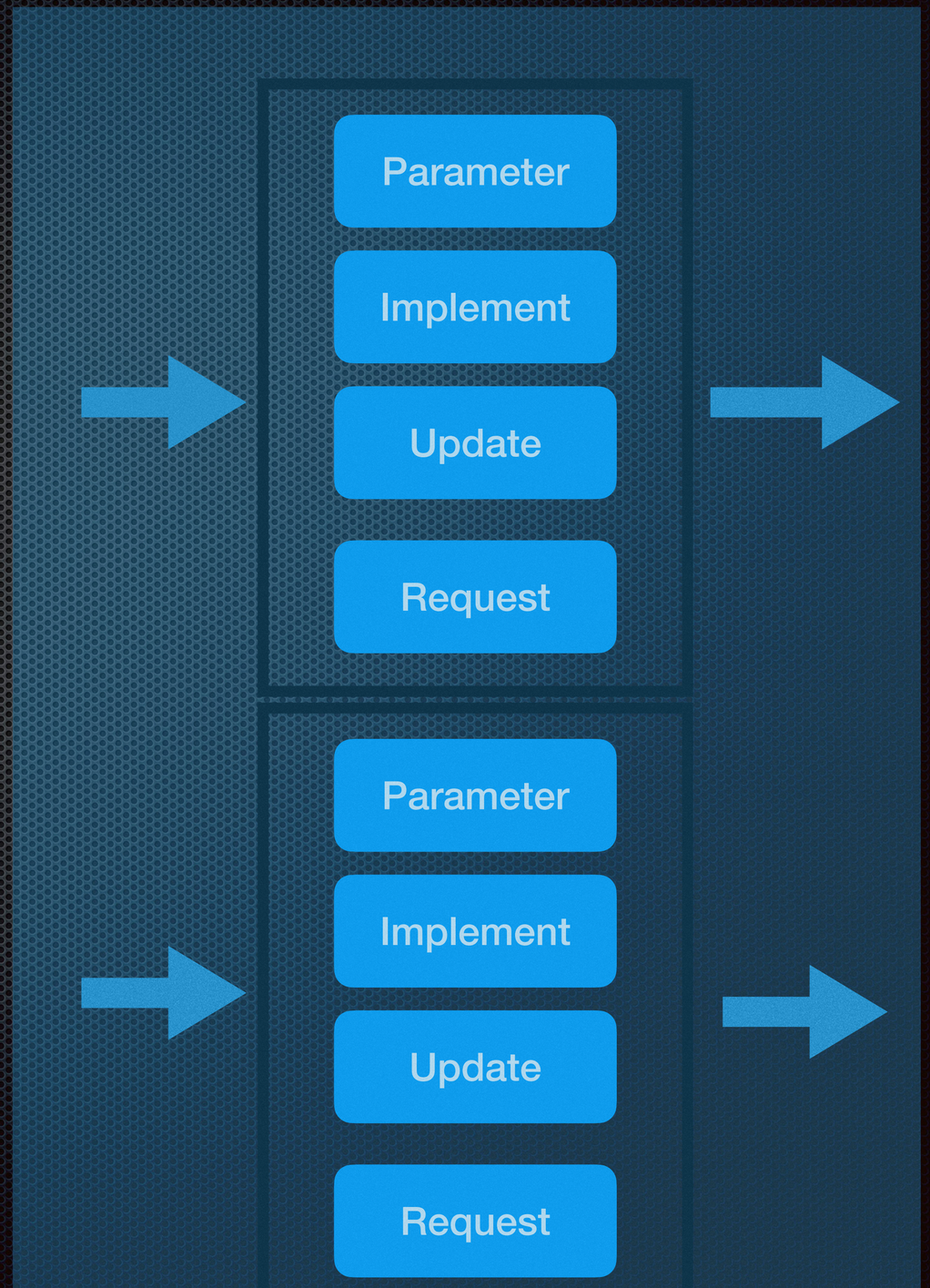
Combine two learners

Serial connection

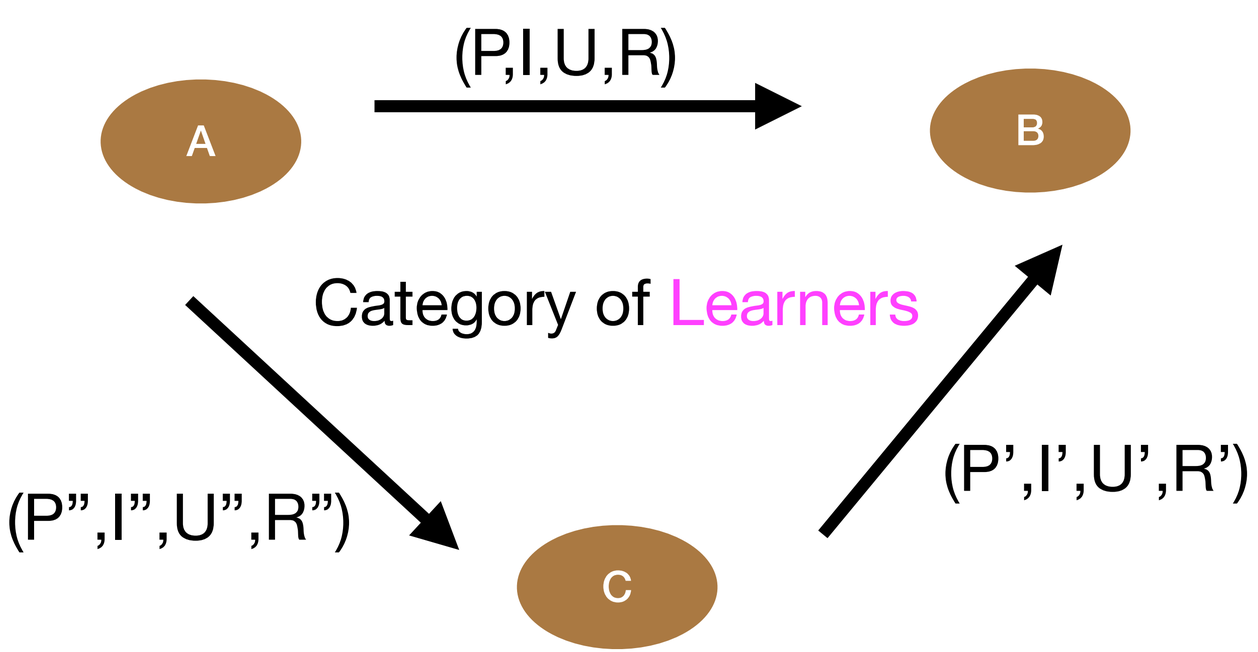


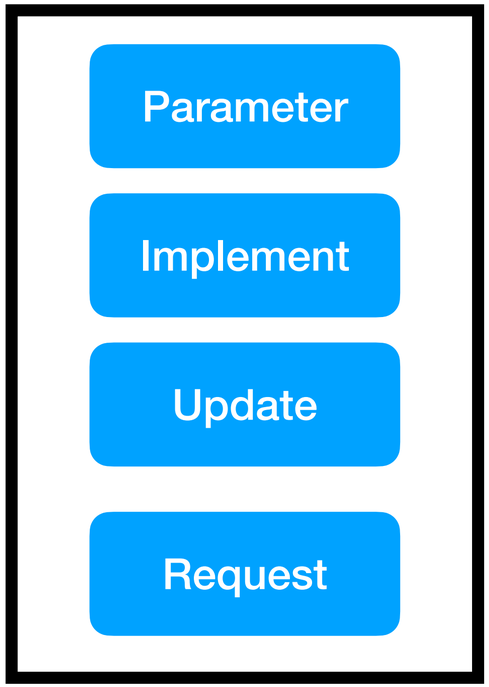
Combine two learners

Parallel connection



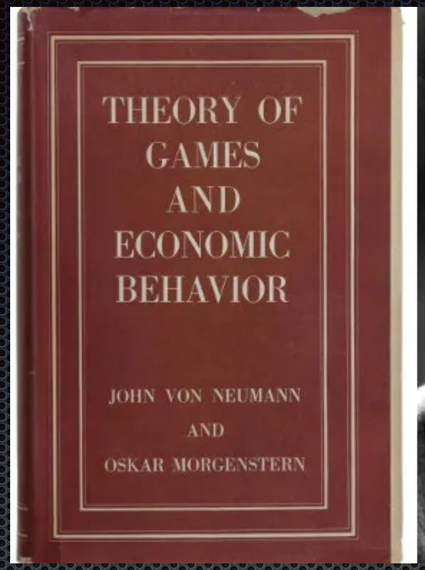
Category of Compositional Learners

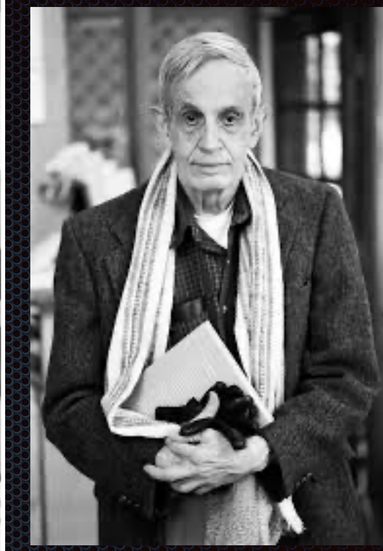




Arrows are Learners!

Game Theory





Von Neumann, Morgenstern and Nash revolutionalized economics with their invention of this decision-making model

		Column	
		Confess	Don't
Row	Confess	(–10, –10)	(0, –20)
	Don't	(–20, 0)	(–1, –1)

Challenge

Define a category of games

What are the objects?

What are the arrows?

Company A \ Company B	Advertise	Don't advertise
Advertise	100 / 100	0 / 200
Don't advertise	200 / 0	0 / 0

Open Games

Breakthrough in defining a category of games

Unique concept: a “co-play” function

Compositional Game Theory

Neil Ghani Jules Hedges Viktor Winschel Philipp Zahn

***Abstract*—We introduce open games as a compositional foundation of economic game theory. A compositional approach potentially allows methods of game theory and theoretical computer science to be applied to large-scale economic models for which standard economic tools are not practical. An open game represents a game played relative to an arbitrary environment and to this end we introduce the concept of coutility, which is the utility generated by an open game and returned to its environment. Open games are the morphisms of a symmetric monoidal category and can therefore be composed by categorical composition into sequential move games and by monoidal products into simultaneous move games. Open games can be represented by string diagrams which provide an intuitive but formal visualisation of the information flows. We show that a variety of games can be faithfully represented as open games in the sense of having the same Nash equilibria and off-equilibrium best responses.**

I. Introduction

The concept of *compositionality* is well-known and almost commonplace in computer science, where it is what ultimately allows programmers to scale software to large systems. However, in many other fields compositionality is essentially unknown and hence its benefits are not available. In this paper we introduce compositionality into a field where one might not believe it to be possible: the study of strategic games and Nash equilibria. They are of interest in economics and computer science where optimal decisions are taken by interacting agents with conflicting goals.[1]

In contrast to classical game theory, where games are studied monolithically as one global object, compositional game theory works bottom-up by building large and complex games from smaller components. Such an approach is inherently difficult since the interaction between games has to be considered. Moreover, in the compositional approach, the equilibria of larger games should be defined from the equilibria of the component games - but *a priori*, there is no reason why this should be possible.

For example, in the prisoner’s dilemma game, each player’s best option is to defect, although, if they acted as a single agent, they would cooperate. Moreover, if the one-shot prisoner’s dilemma game is repeated, then cooperative equilibria become achievable. More generally, the equilibria of a composite game are not necessarily made up from those of the component games, and locally optimal moves are not guaranteed to be globally optimal. In essence, game theory contains *emergent effects* whereby a composite system exhibits behaviours that are not (simple) functions of the behaviours of the components. Accordingly, emergent effects make compositionality very hard to achieve and the existence of a compositional model of game theory is somewhat surprising. In order to arrive at this goal we had to radically reformulate classical game theory from first principles and rebuild it on *open games*.

Open games represent the relationship between different interactions in two dimensions: in sequence, if an interaction follows another interaction and in parallel, if interactions take place simultaneously. As such, we follow a path taken in the field of *open systems* [21], and in particular *categorical open systems* [6] where compositional approaches to general systems are studied. Here, systems are modelled as morphisms $f : X \to Y$ in a symmetric monoidal category, where the objects X and Y describe the *boundaries* of the open system, where it interacts with its environment. This means that systems $f : X \to Y$ and $f' : X' \to Y'$ can be composed in parallel using the monoidal product to yield $f \otimes f' : X \otimes X' \to Y \otimes Y'$, and two systems $f : X \to Y$ and $g : Y \to Z$ sharing a common boundary can be composed sequentially by glueing along this boundary to yield $g \circ f : X \to Z$. Ordinary, closed systems are recovered as scalars [1], i.e. endomorphisms $f : I \to I$ of the monoidal unit, which represents a trivial boundary. Open games are accordingly the morphisms of a symmetric monoidal category.

A compositional model of game theory does not only have to model a game but also the interactions of the game with all other games and environments. This can be seen as a form of *continuation passing style*. This would still be hardly tractable if the environment of an open game included arbitrary other open games. The crucial technical feature underlying our approach is to describe the behaviour of an open game relative to a simplified notion of an environment which we call a *context*, in which the future is abstracted into a single utility function. In this way, we reduce an arbitrarily complex game to a set of individual decisions. The circularity of a Nash equilibrium, where all players play mutually best replies, is finally handled by the composition operators.

The theory of open games is based on two main predecessors. Firstly, in [17] games are defined as processes and in [2] the dynamics but not the equilibria are treated compositionally. The second predecessor is the theory of higher order games and selection functions, for example in [4] and [12], which give a theory of equilibria relative to an environment but are not strongly compositional. Selection functions can be used to model *goals* of agents compositionally [11]. Combining features of these approaches into a single mathematical object required the innovations mentioned above and led us to discover the idea of an open game. After we developed open games, connections to lenses and the geometry of interaction were noticed respectively by Jeremy Gibbons and Tom Hirschowitz.

We omit proofs in this paper, which can be found in [8] and [10]. We also work over the category of sets to keep notation and overheads to a minimum – for a full categorical account, see once more [8]. The rest of this paper is structured as follows: The next section introduces selection functions as a key ingredient to open games. Section III introduces the definition of an open game and discusses its elements, followed by some examples in Section IV. The monoidal category of open games is introduced in Section V and the string diagrams attached to this category in Section VI. We then turn to examples built compositionally: in Section VII we discuss simultaneous move games and in Section VIII sequential move games. Section IX concludes the paper with an outlook on further work.

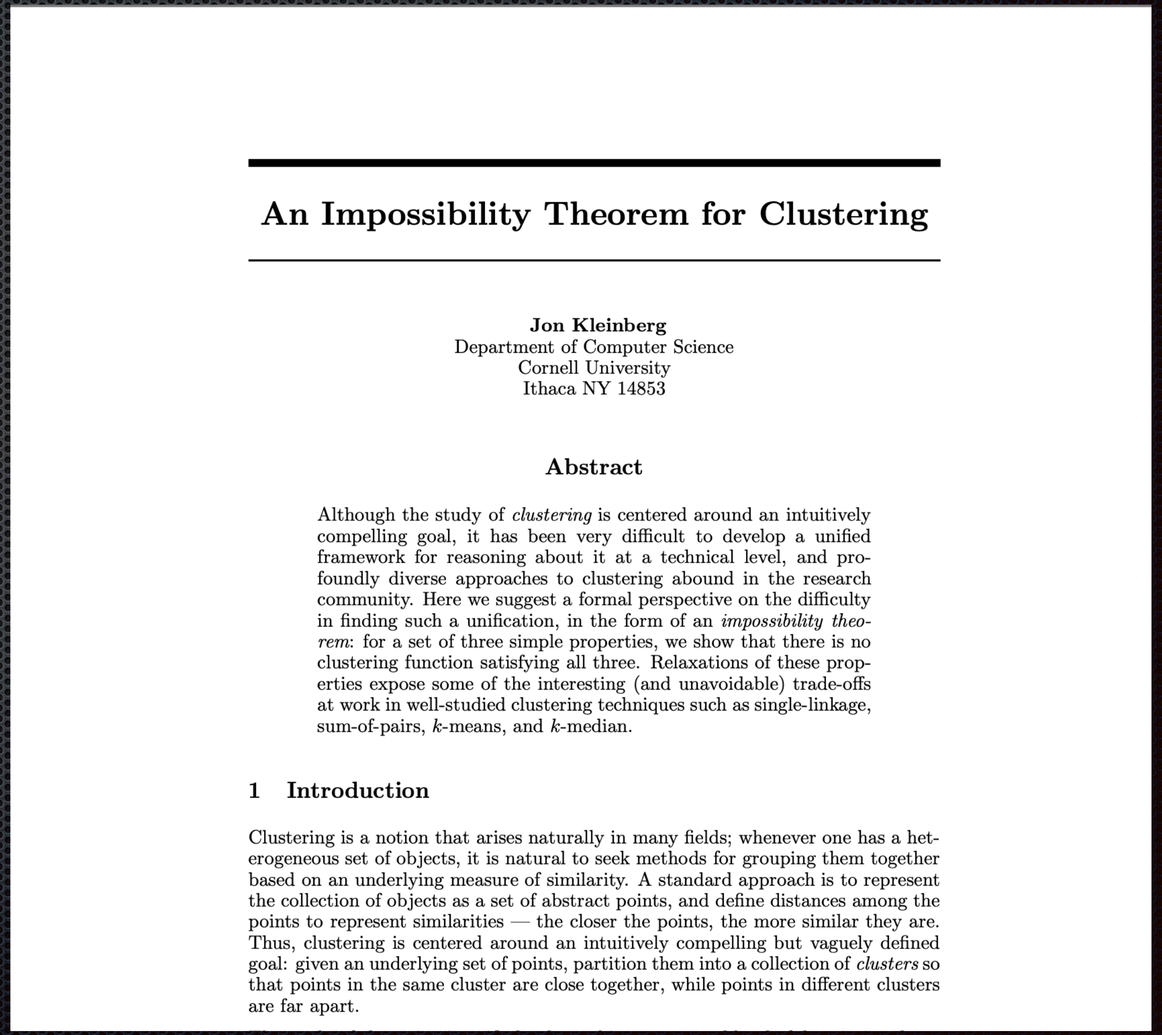
II. Selection functions and higher order games

For reasons of space, we assume the reader knows some basic game theory, such as the definitions of normal-form and extensive-form games and Nash equilibrium. These basic concepts can be found for example in [13] or many online lecture notes. Neverthless, in this

[1] Games in the sense of *game semantics* are compositional, but they avoid several difficult defining features of game theory by restricting to the 2-player zero-sum setting.

An Impossibility Theorem for Clustering

- Kleinberg identified three simple requirements for clustering
  - Scale invariance
  - Completeness
  - Monotonocity
- He showed no clustering algorithm satisfied all three requirements



An Impossibility Theorem for Clustering

Jon Kleinberg
Department of Computer Science
Cornell University
Ithaca NY 14853

Abstract

Although the study of *clustering* is centered around an intuitively compelling goal, it has been very difficult to develop a unified framework for reasoning about it at a technical level, and profoundly diverse approaches to clustering abound in the research community. Here we suggest a formal perspective on the difficulty in finding such a unification, in the form of an *impossibility theorem*: for a set of three simple properties, we show that there is no clustering function satisfying all three. Relaxations of these properties expose some of the interesting (and unavoidable) trade-offs at work in well-studied clustering techniques such as single-linkage, sum-of-pairs, k-means, and k-median.

1 Introduction

Clustering is a notion that arises naturally in many fields; whenever one has a heterogeneous set of objects, it is natural to seek methods for grouping them together based on an underlying measure of similarity. A standard approach is to represent the collection of objects as a set of abstract points, and define distances among the points to represent similarities — the closer the points, the more similar they are. Thus, clustering is centered around an intuitively compelling but vaguely defined goal: given an underlying set of points, partition them into a collection of *clusters* so that points in the same cluster are close together, while points in different clusters are far apart.

Clustering as a Functor!

- Stanford mathematician Gunnar Carlsson showed functors provide the resolution
- Clustering should be functorial!

CLASSIFYING CLUSTERING SCHEMES

GUNNAR CARLSSON AND FACUNDO MÉMOLI

Abstract. Many clustering schemes are defined by optimizing an objective function defined on the partitions of the underlying set of a finite metric space. In this paper, we construct a framework for studying what happens when we instead impose various structural conditions on the clustering schemes, under the general heading of *functoriality*.

Functoriality refers to the idea that one should be able to compare the results of clustering algorithms as one varies the data set, for example by adding points or by applying functions to it. We show that within this framework, one can prove a theorems analogous to one of J. Kleinberg [Kle02], in which for example one obtains an existence and uniqueness theorem instead of a non-existence result.

We obtain a full classification of all clustering schemes satisfying a condition we refer to as excisiveness. The classification can be changed by varying the notion of maps of finite metric spaces. The conditions occur naturally when one considers clustering as the statistical version of the geometric notion of connected components. By varying the degree of functoriality that one requires from the schemes it is possible to construct richer families of clustering schemes that exhibit sensitivity to density.

1. Introduction

Clustering techniques play a very central role in various parts of data analysis. This type of methods can give important clues to the structure of data sets, and therefore suggest results and hypotheses in the underlying science. There are many interesting methods of clustering available, which have been applied to good effect in dealing with many datasets of interest, and are regarded as important methods in exploratory data analysis.

Most methods begin with a data set equipped with a notion of distance or metric. Starting with this input, a method might select a partition of the data set as one which optimizes a choice of an objective function. Other methods such as single, average, and complete linkage define clusterings, or rather nested families of clusterings, using a *linkage function* defined between clusters.

Desirable properties of clustering algorithms come from practitioners who have intuitive notions of what is a good clustering: "they know it when they see it". This is of course not satisfactory and a theoretical understanding needs to be developed. However, one thing this intuition reflects is the fact that density needs to be incorporated in the clustering procedures. Single linkage clustering, a procedure that enjoys several nice theoretical properties, is notorious for its insensitivity to density, which is manifested in the so called *chaining effect* [LW67].

Other methods such as average linkage, complete linkage [JD88, Chapter 3] and k-means

Reading for the Week

- Read Chapter 1 of Riehl's textbook
- Read Chapter 1 of my lecture notes
- Try sample GitHub Python code

GitHub Repository Demos

- Backprop as a Functor demo
- Clustering as a Functor demo
- If you are curious about the Geometric Transformer
  - GT for language modeling
  - GT for Sudoku
  - GT works via Diagrammatic Backpropagation (DB)
  - Key idea: turn lack of functoriality into a novel curvature loss function!