# A Non-Sculpting Theorem in Non-Interleaving Models for Concurrency

*Thesis submitted in partial fulfillment of the requirements for the degree of*
***Master of Science (MSc)***
*to the The Faculty of Mathematics and Natural Sciences,*
*at the University of Oslo.*


**Christopher A. Trotter**
*Reliable Systems (PSY)*
*Department of Informatics, University of Oslo*




*Supervisors:*
**Christian Johansen**, University of Oslo
**Ulrich Fahrenberg**, Ecole Polytechnique, France
**Martin Steffen**, University of Oslo
**Olaf Owe**, University of Oslo


November 15, 2018

# Abstract

The geometric model, Higher-dimensional automata (HDA), is a useful and general model for non-interleaving concurrency. In a non-interleaving approach to concurrency, more than one event may happen concurrently and one differentiates between concurrent and interleaving executions, satisfying $a \parallel b \neq a.b + b.a$. HDA also encompass all other commonly used models of concurrency. However, due to generality HDAs are challenging to work with. Vaughan Pratt introduced sculptures and Chu spaces as models which retain some of the good properties of HDA as well as being easier to work with. Recently, Johansen has introduced ST-structures as another event-based formalism for the same purpose.

This thesis gives a precise definition of sculptures, following the intuition of Pratt, where one can think of the process of modelling a concurrent system using higher-dimensional automata as a *sculpting* process as follows. Take one single higher-dimensional cube, having enough concurrency, and remove cells until the desired concurrent behaviour is obtained. This is different from *composition* where a system is built by composing together smaller systems, which in higher-dimensional automata is done by gluing together cubes.

We also develop an algorithm to decide whether a higher-dimensional automaton is a sculpture or not, and use this to show that some simple acyclic higher-dimensional automata are not sculptures. We believe that this contradicts Pratt's intuition that sculptures suffice for the modelling of concurrent behaviour, because not all higher-dimensional automata can be sculpted.

We show that sculptures, Chu spaces and ST-structures are in close correspondence. This nicely captures Pratt's event-state duality and tightens the correlation between ST-structures and higher-dimensional automata, which was left as an open problem by Johansen.

# Acknowledgements

*A significant portion of this thesis represents joint work with my supervisors Christian Johansen and Ulrich Fahrenberg. They have been instrumental in shaping my ideas of concurrent systems and my view of research in general, and their support and intellectual guidance have made this thesis possible. I am also indebted to my other supervisors, Martin Steffen and Olaf Owe. Thank you for sharing your knowledge, offering advice, and for every one of our helpful discussions.*

*So many others have also offered their help along the way. Among them I would like to thank my brother James for taking the time to listen and for offering his valuable insight and perspective. I would also like to thank Bibek Kabi for all the coffee breaks with insightful conversations while I was in Paris as part of my internship. I am grateful to my fellow students who have never turned me down when I have asked questions. In particular, I want to mention Marius Andresen, Mateusz Zych, Norah Nguyen and Ratan Thapa.*

*I would like to thank my family and friends who have supported me along the way. Thank you so much.*

*Last, but certainly not least. Thank you, Zahra.*

*Christopher A. Trotter,*
*November 2018*

# Contents

# List of Figures

# Chapter 1

# Introduction

The geometric model of concurrency, studied by Pratt and van Glabbeek [40, 41, 13], is a useful and general model of non-interleaving concurrency. This model was named *higher-dimensional automata* (HDA) by Pratt [41]. In non-interleaving concurrency, we differentiate between concurrent and interleaving executions, using CSS notation [32], $a \parallel b \neq a.b + b.a$. Also, more than one event may happen in non-interleaving concurrency. This differentiation is opposite from that of interleaving concurrency where one assumes that $a \parallel b = a.b + b.a$. Since in interleaving concurrency certain actions may be atomic, meaning that these actions are indivisible and instantaneous.

Intuitively, an higher-dimensional automaton is an automaton with nicely incorporated squares, cubes and higher-dimensional cubes, which represent the independence of events. Such a concurrency model is interpreted geometrically, roughly as topological spaces in which paths correspond to executions and two executions are equivalent when the corresponding paths are homotopic. A deformation, where one path can be "*continuously deformed*" into another path, is called a homotopy between the two paths as the two paths are considered homotopic. If two paths are homotopic, then we may consider two paths as being a single path, hence reducing the state space. For our purpose, topological spaces are not exactly the right notion. We need a *directed* variant to incorporate a notion of irreversible time, in other words making it possible to determine the order of an execution.

Higher-dimensional automata are able to also encompass interleaving models and all the other commonly used models of concurrency, such as *transition systems* [49], *asynchronous transition systems* [45], *petri nets* [36], *event structures* [33] and *configuration structures* [17]. However, because of the generality of higher-dimensional automata they are challenging to work with. From the work in [28] on *ST-structures*, which were introduced as event-based counterpart of higher-dimensional automata, we know that it is not always possible to precisely identify the events in a higher-dimensional automaton. Vaughan Pratt introduced *sculptures* [40] and *Chu spaces* [27, 38] as models which retain some good aspects of higher-dimensional automata as well as being easier to work with. Chu spaces have been developed as a response to the event-state duality argued by Pratt [39]. However, a study of sculptures has yet to be done.

The subject matter of this thesis is to give a precise definition of sculptures, following the intuition of Pratt, where one can think of the process of modelling a concurrent system using higher-dimensional automata as a *sculpting* process as follows. Take one single higher-dimensional cube, having enough concurrency (that is, enough events) and remove cells until the desired concurrent behaviour is obtained. The model obtained is a sculpture.

We investigate a conjecture posed by Vaughan Pratt at the end of the introduction of [40]. The conjecture is that any higher-dimensional automata can be obtained using the sculpting method. We develop an algorithm to decide whether a higher-dimensional automaton can be sculpted or not, and show several simple examples of *acyclic* higher-dimensional automata which are not sculptures. We believe that this contradicts Pratt's claim cited above. We also show that there is a close correspondence between sculptures, Chu spaces over 3 [40] and ST-structures.

In Chapter 2 and 3, we will review some models of concurrency. We will see the ideas which they contributed to understanding higher-dimensional automata, and also see how they fall short of our intuitions of concurrent behaviour. We will also review some models of concurrency, in Chapter 3 and 4, which contributed ideas to understanding the sculpting method. These are in particular higher-dimensional automata, ST-structures and Chu spaces. Before further delving into the technical details of higher-dimensional automata and sculptures, some general considerations about the nature of concurrency models and their geometric notion might serve as an introduction.

## 1.1 Models of concurrency

Models of concurrency which have been studied and used within theoretical computer science depend on mathematics as a foundation in which to describe and reason about the behaviour of concurrency. Whenever a programmer has written a program, they tend to have an idea of how the system should behave. For example, if a programmer writes *if*$(a$ *and* $b)$ *then* $c$, then the programmer has some idea of how the system would evaluate the statement $a$ *and* $b$. This may, or may not, correspond to how the statement is actually evaluated by the system. Nevertheless, the purpose of a mathematical model is to make precise the intuitions, so that the programmer perceives the program to behave as the actual implementation. In concurrency a program behaves unambiguously making mathematical models absolutely necessary. The goal of mathematical models is to provide an understanding of systems and their behaviour in theory, and to contribute to methods of analysis and design in practice.

Models of concurrency have a concept of states, where a state is a snapshot of a system at a particular moment. These models also have a concept of transitions, which is a way of moving from one state to another. Various models represent these transitions as *actions* and *events*. Transition systems, asynchronous transition

systems and higher-dimensional automata, are *action based*, and also state-based, models because any transition can occur repeatedly. Configuration structures, event structures and ST-structures are *event-based* models, where each event can occur at most once during the execution of a process. Intuitively, actions are labels on events, and may occur repeatedly, and events occur only once and are labeled by actions. For example, consider someone sending an email to two different people. We have that the "send" action is repeated, however, these are considered two distinct events since the email is sent to two different recipients.

Event-based models use labelling functions to represent multiple occurrences of an action, so that each event is labeled with an action. In an event-based model there are no loops, that is, after moving out from a state a system can never return to that state. Also, event-based models are able to capture the entire history of a process. The main disadvantage of storing the entire history is that processes which can be represented by small structures in a state-based model may need an infinite structure in the event-based model. Because of the repetitive nature of state-based models, for example, loops in a transition system. However, representing each event explicitly allows event-based models to be formulated as categories. A category is an abstraction of a mathematical concept that provides a way of relating mathematical structures by structure-preserving functions.[1] Having a history of each occurrence of an action makes the information content of each state very clear, making them close to domain theory and providing them with a schedule-automata duality [33, 17], also called *event-state duality*.

Another way to classify models of concurrency is as *interleaving* and *non-interleaving*. Interleaving models have actions which are *atomic* or indivisible, and the parallel execution of two atomic actions is considered equivalent to executing them in either order, that is, satisfying the law: $a \parallel b = ab \cup ba$. Transition systems are examples of interleaving models. Non-interleaving models allow the splitting up of an action into several smaller actions, and the parallel execution of two actions are considered to have no information about the order relation of those actions. This is regarded as different from $ab \cup ba$, which represents the choice between $ab$ and $ba$. Asynchronous transition systems, event structures, higher dimensional automata, ST-structure and Chu spaces are non-interleaving models.

In Section 2.3, we attempt to provide some intuition on the difference between interleaving and non-interleaving models, which have been shown to interpret the event-state duality differently [41, 39]. In interleaving models, the interpretation of the event-state duality is as shown by Winskel et al. [33] and van Glabbeek [17]. For non-interleaving models, there is still a need of further investigation of the event-state duality. However, Chu spaces were created as a response to solve the event-state duality [27]. In Section 4.3, we touch upon the event-state duality of the models of concurrency presented in Chapter 3.

[1] The standard introduction to category theory is the classical book by Mac Lane [30], which is aimed at mathematicians. For an introduction to category theory in computer science, the book by Pierce [37] is aimed at theoretical computer scientist.

Interleaving models assume actions to be instantaneous, such that for some kinds of observations the two processes $a \parallel b$ and $ab \cup ba$ are observably the same [18]. With this assumption, we are able to have a tractable algebraic theory. On the other hand, the interleaving assumption forces actions to be regarded as atomic. Atomic actions impose a certain abstraction level to be the lowest granularity level [19, 15, 41]. For example, a programmer may give details on actions that, at the time, are regarded as atomic. However, at a future release the programmer may discover that the actions have substructures, see [19, Example 1.1]. The programmer cannot refine the action since it is atomic. In Section 2.3, we advocate for refinement of actions and provide Pratt's interpretation of refinement [41]. Hence, we are interested in non-interleaving models, presented in Chapter 3, that have an algebraic structure. Having an algebraic structure provides a tractable algebraic theory to reason about the behaviour of concurrency mathematically.

## 1.2 Refinement of actions

Concurrent systems should have the possibility to be developed incrementally by starting from an abstract specification and gradually obtaining more details. An incremental approach would be able to update existing systems rather than implementing new systems. Naturally, such systems would be considered efficient and reliable. In concurrent systems, the incremental development can be done by refining actions.

Refinement of actions can be considered for both interleaving and non-interleaving models. For interleaving, a refinement is often found in process algebras where an implementation refines a specification by reducing the set of execution traces. The set of execution traces is reduced when the non-determinism of a specification is removed. Non-interleaving models regard refinement to be a refinement of actions, called *action refinement*. Action refinement is the method of developing a system by starting with an abstract specification, and gradually refining its action, by providing more details. Hence, an action can be changed from being instantaneous, as with interleaving, to having structure, or duration.

If we consider the use of action refinement in the programmer example above, then the programmer is able to provide an abstract specification at first release and at future releases give more details to actions. Action refinement will not be presented here, however, for a detailed example consider the example presented by van Glabbeek in [19, Example 1.1].

With non-interleaving models we are able to consider a system at any abstraction level by refining actions. In Section 3.1, we will show that some non-interleaving models are only able to reason about the independence of pairs of events, where we might naturally need to reason about an arbitrary number of independent events.

## 1.3 Geometric models of concurrency

Geometric models of concurrency are non-interleaving models with an algebraic structure, which provides an algebraic theory for non-interleaving models. Geometric models are useful and general since they are able to capture the behaviour of other interleaving and non-interleaving models [25], such as transition systems and asynchronous transition systems [9]. Geometric models enable the use of techniques from algebraic topology in order to study concurrent behaviour. Van Glabbeek in [13] considered higher-dimensional cubes as a natural way of interpreting concurrency, which lead to the development of higher-dimensional automata [13, 20, 41]. Goubault furthered the development of higher-dimensional automata by showing that many of the other models of concurrency, such as event structures, transition systems and asynchronous transition systems, could be interpreted in terms of higher-dimensional automata [20, 25]. Higher-dimensional automata can be seen as a generalization of other models of concurrency.

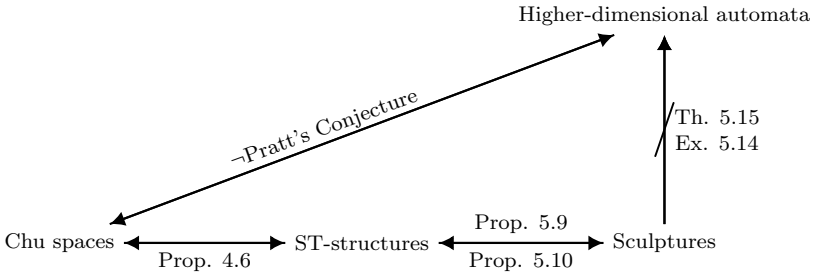
ST-structures were introduced by Johansen in [28] as the event-based counterpart of higher-dimensional automata. With an event-based model, we are able to make the information content of each state very clear. As shown by Johansen in [28], higher-dimensional automata are highly expressive and capture the main characteristic of interleaving and non-interleaving models. However, higher-dimensional automata cannot precisely identify events, such as the asymmetric conflict shown in [28, Figure 5]. We will attempt to identify events in higher-dimensional automata in Chapter 4. We will also present Chu spaces since they are considered to be a response to the state-event duality of Pratt [39]. With Chu spaces we can reconcile the relationship between ST-structures and higher-dimensional automata, but also introduce sculptures.

In the meanwhile, we provide the following short summary. We will be considering traditional models of concurrency to see the ideas which they contribute to understanding higher-dimensional automata. We are interested in studying models for concurrency that follow the principle of non-interleaving concurrency, but in relation to the state-based model of higher-dimensional automata. In particular, models based on sets of events. With such models, we are able to precisely identify the events making the information content of each state very clear. We focus on identifying events in higher-dimensional automata. Specifically, by transforming Chu spaces [27], ST-structures [28] and sculptures into equivalent higher-dimensional automata. Our goal is to challenge a conjecture posed by Vaughan Pratt in [40], at the end of the introduction of [40].

In higher-dimensional automata one can think of the process of modelling a concurrent system using higher-dimensional automaton as a *sculpting* process as follows. Take one single higher-dimensional cube, having enough concurrency, that is, enough events, and remove cells until the desired concurrent behaviour is obtained. This is different than what is usually done in process algebras where a system is built by composing together smaller systems, which in higher-dimensional automata is

done by gluing together cubes.

The conjecture is that any higher-dimensional automaton can be obtained using the sculpting method. To answer this we first need to define precisely the sculpting method, following again the intuition of Pratt that sculpting is similar to subalgebras. More intuitions and examples are in Chapter 5 where we define sculptures. We develop an algorithm to decide whether an higher-dimensional automaton can be sculpted or not, and show several simple examples of *acyclic* higher-dimensional automata which are not sculptures. We believe that this contradicts Pratt's conjecture. We also identify exactly the class of higher-dimensional automata that are sculptures.




In Proposition 4.6, we show the isomorphism between Chu spaces and ST-structures. Furthermore, we provide further proof in Proposition 5.9 and Proposition 5.10 that ST-structures and sculptures are isomorphic. Hence, sculptures are also isomorphic to Chu spaces. To strengthen the proofs, we provide examples of higher-dimensional automata which are not sculptures, see Example 5.14. We use this example to show the *non-sculpting* theorem.

## 1.4 The contributions of this thesis

Part of this thesis is dedicated to giving an introduction to concurrency models, where we focus on non-interleaving models of concurrency capable of capturing the independence of arbitrary number of events. In order to achieve this, we have included a review of several existing models of concurrency in Chapter 2 and 3, including transition systems, event structures, configuration structures, asynchronous transition systems, higher dimensional automata, ST-structure and Chu spaces. This part does not represent new work in itself, and the relationships between the models in Chapter 4 have been investigated and are known in the geometric models of concurrency literature.

On the other hand, we show the ideas the above mentioned models of concurrency have contributed to understanding both higher-dimensional automata and

sculptures. In Chapter 2, we expend some effort to explain the difference between interleaving and non-interleaving models, and the need for non-interleaving models to incrementally develop concurrent systems by action refinement. Also, we investigate the event-state duality of non-interleaving models to better understanding the current research being done. Specifically, investigating event-state duality in higher-dimensional automata by ST-structures and Chu spaces.

The main contribution is the method of sculpting that is presented in Chapter 5. Sculpting is a method of modelling concurrent behaviour of higher-dimensional automata. We investigate a conjecture posed by Vaughan Pratt in [40]. The conjecture is whether any higher-dimensional automata can be obtained using the sculpting method. We first define precisely the sculpting method, following again the intuition of Pratt that sculpting is similar to subalgebras. We develop an algorithm to decide whether an higher-dimensional automaton can be sculpted or not, and show several simple examples of *acyclic* higher-dimensional automata which are not sculptures.

We attempt to precisely identify the class of higher-dimensional automata which by sculpting can identify events. From the work by Johansen [28], we known higher-dimensional automata are in general not good at identifying events. However, as we show in this thesis, sculptures are well suited to represent the events, thus overcoming the problem identified in [28, Figure 5]. For example, the asymmetric conflict cannot be represented as a higher-dimensional automata, but can be faithfully represented as a sculpture or as an ST-structure. Going further, sculptures are one response to the event-state duality of Pratt [39] in higher-dimensional automata.

## 1.5 The organisation of this thesis

In this section, we briefly present the organisation of the thesis with an overview of each chapter.

**Chapter 2** presents transition systems and event structures. Transition systems are interleaving models of concurrency and can be interpreted, geometrically, as one-dimensional spaces consisting of edges (its transitions) meeting and branching at vertices (its states). Transition systems provide the foundation to understand the generalization of the geometric model of concurrency. Also, transition systems are automata and considered one side of a duality, where schedules, or events, is the other. Event structures are schedules which provide the necessary background for understanding ST-structures and provide the intuition of the state-event duality presented in Chapter 4.

**Chapter 3** presents the non-interleaving models of concurrency, specifically, asynchronous transition systems, higher-dimensional automata, ST-structures and Chu spaces. We consider asynchronous transition systems to be the bridge between transition systems and higher-dimensional automata. Asynchronous transition

systems are seen, geometrically, as a two-dimensional space extending the transition systems by considering surfaces (its independence relations) between edges (its transitions) and vertices (its states). Higher-dimensional automata are a generalization of both transition systems and asynchronous transition systems. We consider higher-dimensional automata as an $n$-dimensional space consisting of points, segments, squares, cubes, and higher-dimensional cubes. ST-structures are the event-based counterpart of higher-dimensional automata capable of capturing a main characteristic of higher-dimensional automata, that is, to be able to see what happens during a concurrent execution. Chu spaces are considered a response to the state-event duality problem and introduce the notion of "*sculpting*", which we present in Chapter 5.

**Chapter 4** presents the relationship between certain non-interleaving models of concurrency. Specifically, we will relate higher-dimensional automata and ST-structures as well as ST-structures and Chu spaces. Higher-dimensional automata are not good at identifying events as show in Example 4.2, but can be faithfully interpreted as ST-structures to identify events. In Example 4.2, isomorphic higher-dimensional automata are interpreted as non-isomorphic ST-structures. Hence, there is not an embedding, that is, an injective morphism, from $\mathbb{ST}$ to $\mathbb{HDA}$. We also show in Figure 4.2 that non-isomorphic higher-dimensional automata interpreted as ST-structures become isomorphic ST-structures, meaning there is not an embedding from $\mathbb{HDA}$ to $\mathbb{ST}$. Hence, higher-dimensional automata are neither more, or less, expressive than ST-structures. We investigate the relationship between Chu spaces and ST-structure to better understand the state-event duality.

**Chapter 5** presents the method "*sculpting*" which has not been studied for higher-dimensional automata before. We investigate the relationship between sculptures and ST-structures to tighten the correlation between ST-structures and higher-dimensional automata, which was left open in [28]. We develop an algorithm to decide whether an higher-dimensional automaton can be sculpted or not, and show several simple examples of acyclic higher-dimensional automata which are not sculptures. We believe that this contradicts Pratt's conjecture.

**Chapter 6** presents some concluding remarks, summarizes the contributions of the thesis and offers some ideas that can be pursued in future work.

# Chapter 2

# An introduction to traditional models of concurrency

For the most part, early models of computation, such as Turing Machines [47], are sequential, meaning that there is a strict order in which computational steps happen. The strict ordering ensures that the sequence of computational steps are unambiguous, such that one could provide a single history of the executed computation. For sequential computation, the $\lambda$-calculus has become a fundamental unifying formalism. This framework is also accommodated by a rich variety of developments such as types and higher-type computation, program logic, operational semantics, and denotational semantics. However, concurrent computation has yet to see a unifying framework in the way that $\lambda$-calculus is for sequential computation.

There has been extensive research within the field of concurrency theory to approach a unifying concept, and an outline of many deep, fundamental results was done by Garavel in [10]. In this thesis, we will investigate one of the opportunities mentioned by Garavel and present the notion of sculpting for higher-dimensional automata, as a result. The opportunity described was the idea of having models everywhere and a way to transform one model into another. Specifically, models for different purposes and a way to translate between them. Considering this opportunity would unify several pre-existing concepts such as translation and refinement.

Since the 1960s, a wide variety of models of concurrency have been proposed. One of the first proposals was Petri nets. During the past decades, there have been other proposals for models of concurrency, such as transition systems, higher dimensional automata and event structures. Even though the nature of these models are very different, they have shown to have a tight relation to each other, as presented by Winskel [49] and Goubault [25]. Because of this relation, we classify models of concurrency as state-based, *automata*, or event-based models, *schedules*. Also, we further classify these as interleaving or non-interleaving models. For sequential behaviour these classes are considered equivalent. In this thesis, we will focus on the classification of interleaving versus non-interleaving models. We will first review *transition systems* as an interleaving model and show how they fall short of

our intuition of concurrent behaviour.

## 2.1 Transition systems

Transition systems are well established semantic models for both sequential and concurrent systems. A transition system can be viewed as a directed graph, called a state transition graph, whose vertices are states and edges are transitions between states. We may consider a transition system, as shown in Figure 2.1, where we have four states $s_1$, $s_2$, $s_3$, and $s_4$, for example, representing a program changing a value of a variable at different times of its execution. We also have four transitions, two of which are labeled $a$ and another two which are labeled $b$. We can also have the same label $a$ for both events, giving rise to the notion of *autoconcurrency*. These four transitions, represent two instructions of a program that change the state of the machine from the source of its arrow to its corresponding target. This is formalized by using a transition relation. We will refer to this transition system as the *interleaving square*, similar to what has been done by Pratt [40] and van Glabbeek [13]. We will use the same notation and definitions as Winskel and Nielsen in [49].

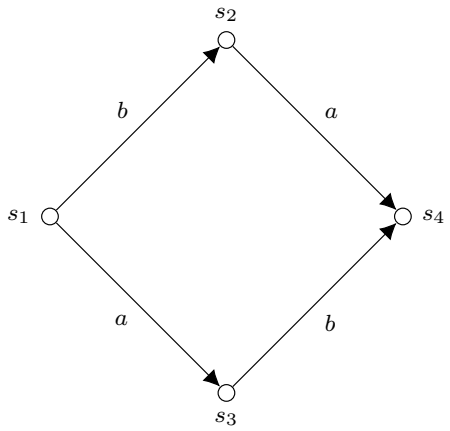



**Figure 2.1** A transition system with four states, $s_1, s_2, s_3$, and $s_4$, and four transitions, two of which are labeled $a$ and the other two are labeled $b$. These four transitions present the interleaving of two instructions. The interleaving is namely the permutation of all possible executions of these two instructions. This transition system is called the interleaving square because of the way the interleaving of two instructions form a square.

**Definition 2.1** (Transition system [49]). *A transition system is a structure $(\mathcal{S}, i, \mathcal{L}, Tran)$ where,*

- *$\mathcal{S}$ is a set of states with initial state $i$,*
- *$\mathcal{L}$ is a set of labels, and*
- *$Tran \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is the transition relation*

We will write $s \xrightarrow{a} s'$ to indicate that $(s, a, s') \in Tran$.

Morphisms between transition systems often describe the relationships between processes. These kind of relations are described by Winskel and Nielsen in [49], where one process may be a component of another or perhaps one process may refine another process. By defining morphisms as simulations between transition systems, we are able to describe the relationship between processes. For example, a simulation is where a transition system $\mathcal{T}$ simulates a transition system $\mathcal{T}'$, meaning that when $\mathcal{T}'$ can execute some action $a$ then there must exists an $a$ in $\mathcal{T}$ that can be executed.

**Definition 2.2** (Morphism of transition systems [49])**.** *A morphism from one transition system, $\mathcal{T}$, to another, $\mathcal{T}'$, will be a pair $(\sigma, \lambda)$ in which*

- *$\sigma$ is a function from the states of $\mathcal{T}$ to those of $\mathcal{T}'$.*
- *$\lambda$ is a partial function from the labels of $\mathcal{T}$ to those of $\mathcal{T}'$ such that for any transition $s \xrightarrow{a} s'$ of $\mathcal{T}$ if $\lambda(a)$ is defined, then $\sigma(s) \xrightarrow{\lambda(a)} \sigma(s')$ is a transition of $\mathcal{T}'$; otherwise, if $\lambda(a)$ is undefined, then $\sigma(s) = \sigma(s')$.*

In our notation, we have used $\mathcal{L}$ to stand for the set of labels for events, as well as to stand for the actual set of those events. However, sometimes it might be useful to make a distinction between the events themselves and to explicitly give a labeling as a function. For instance, this is important when treating *relabeling* which leads to categorical fibrational situations [49]. To make the distinction between labels of events and the actual events clearer, we will replace $\mathcal{L}$ with $E$ and refer to its elements as *events*.

**Definition 2.3** (Labeled transition system [49])**.** *A* labeled *transition system consists of a transition system $\mathcal{T} = (\mathcal{S}, i, E, Tran)$ together with a set $\mathcal{L}$ of labels, and a function $l : E \to \mathcal{L}$. We denote it by $(\mathcal{T}, \mathcal{L}, l)$.*

**Definition 2.4** (Morphism of labeled transition systems [49])**.** *A morphism, $(\sigma, \tau, \lambda) : (\mathcal{T}, \mathcal{L}, l) \to (\mathcal{T}', \mathcal{L}', l')$ between labeled transition systems consists of a morphism $(\sigma, \tau) : \mathcal{T} \to \mathcal{T}'$ between the underlying transition systems together with a partial function $\lambda : \mathcal{L} \to \mathcal{L}'$ such that $l' \circ \tau = \lambda \circ l$.*

We write $\mathbb{TS}$ for the category of transition systems. Also, we name $\mathbb{TS}_E$ its subcategory where we restrict to transition systems labeled on an alphabet $E$. The alphabet $E$ can be considered the same as the set of events $E$. The notion of transition systems and of morphisms between them is clearly related to directed graphs, labeled directed graphs, and labeled transition systems, but we will need to consider labeled cubical sets, which we will not do here. However, we will introduce precubical sets[1] in Section 3.2.

[1] In topology, the difference between cubical and precubical sets are similar to that of simplicial and presimplicial sets. The differences are subtle and will not be considered here. The distinction between cubical and precubical sets is presented in [20].

## 2.2 Event structures

Event structures were developed in [48] as an attempt to bridge Petri net theory[2] and domain theory. Specifically, event structures is considered an extension to stable families, which are sets of all events which may have occurred by some stage, or history, in the evolution of a process. An event structure represents a process. We will adopt the same notion as used by Winskel [33] and Van Glabbeek & Goltz [17] that families are called *configuration structures*. This bridge between Petri nets and domain theory would allow concurrent systems to be translated into automata, or schedules, providing a schedule-automata duality [33, 17].

**Definition 2.5** (Event structure [49]). *An event structure $(E, \leq, \#)$ consists of a set $E$, of events which are partially ordered by $\leq$ (the causal dependency relation) and a binary, symmetric, irreflexive relation $\# \subseteq E \times E$ (the conflict relation) which satisfy*

- $\{e' \mid e' \leq e\}$ *is finite (axiom of* finite causes*), and*
- $e\#e'$ *and* $e' \leq e'' \Rightarrow e\#e''$ *(conflict is* hereditary*).*

*for all* $e, e', e'' \in E$.

Event structure is formally defined as a triple $(E, \#, \leq)$, where $E$ is the set of events, $\# \subseteq E \times E$ is a conflict relation, and $\leq \subseteq E \times E$ is a partial order. If two events are in conflict, not both of them can happen in a single execution. Also, for the event $e$ to happen, all the events before it in the partial order must have happened. We consider that events $e$ and $e'$ are *concurrent*, denoted $e \; co \; e'$, as follows:

$$e \; co \; e' \;\text{ iff } \textit{not } ((e \leq e') \text{ or } (e' \leq e) \text{ or } e\#e')).$$

If $e$ and $e'$ are not dependent on each other in either order and there is no conflict, then we may run $e$ and $e'$ concurrently.

The events in a prime event structure can be thought of as event occurrences without significant duration, in any history an event appear at most once. The relation of *immediate* dependency $e \rightarrow e'$ means $e$ and $e'$ are distinct with $e \leq e'$ and also that there is no event in between.

**Definition 2.6** (Configurations [49]). *Let $(E, \leq, \#)$ be an event structure, and its* configurations, *$\mathcal{D}(E, \leq, \#)$, to consist of those subsets $x \subseteq E$ which are*

- *conflict-free:* $\forall e, e' \in x. \neg(e\#e')$*and*

[2]Petri nets are considered a classical model of concurrency with important ideas to understand the current research. Research into Petri nets has been done extensively in models of concurrency literature. Hence, we will not investigate Petri nets. One can find an introduction to Petri nets in [36], and one can also find a recent comparison to modern models of concurrency in [25].

- *downwards-closed:* $\forall e, e'.e' \leq e \in x \Rightarrow e' \in x$.

As shown by Winskel in [49, Proposition 18], we can recover the important relations, such as conflict and partial order, associated with an event structure from its finite configurations. We will not present that here. We will, however, use configurations in our definition of a category of event structures. First, we have to introduce morphisms of event structures.

**Definition 2.7** (Morphisms of event-structures [49])**.** *Let* $ES = (E, \leq, \#)$ *and* $ES' = (E', \leq', \#')$ *be event structures. A* morphism *from* $ES$ *to* $ES'$ *consists of a partial function* $\eta : E \to_* E'$ *on events which satisfies*

- $x \in \mathcal{D}(ES) \Rightarrow \eta x \in \mathcal{D}(ES')$,
- $\forall e, e' \in x.\eta(e), \eta(e')$ *are both defined, and*
- $\forall e, e' \in x.\eta(e) = \eta(e') \Rightarrow e = e'$.

If we let $(E, \leq, \#)$ be an event structure and let $x, x'$ be configurations, then we write $x \xrightarrow{e} x' \Leftrightarrow e \notin x$ and $x' = x \cup \{e\}$. From Proposition 19 in [49], we have that two events $e$ and $e'$ are concurrent if and only if there exists configurations $x_1, x_2, x_3$ and $x_4$ such that we get the interleaving square, as shown in Figure 2.2.

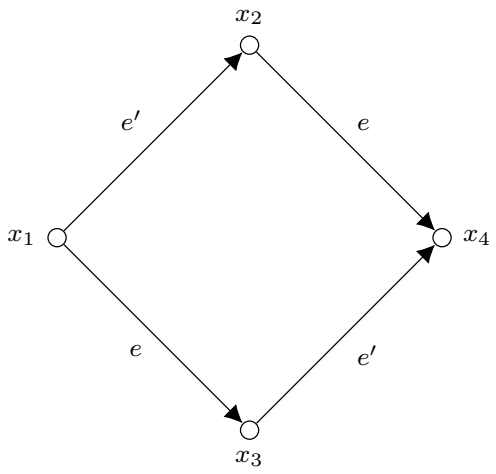



**Figure 2.2** An event structure with two concurrent events $e$ and $e'$ such that we get the configurations $x_1, x_2, x_3$, and $x_4$. The configurations generate the interleaving square because of the way the concurrent events $e$ and $e'$ form a square.

We write $\mathbb{ES}$ for the category of event structures. Also, we name $\mathbb{ES}_E$ its subcategory where we restrict to event structures labeled on an alphabet $E$. The alphabet $E$ can be considered the same as the set of events $E$. Events are considered one side of a duality [39], where automata is the other side. In Section 3.4 we will introduce Chu spaces [27], which is a general framework where automata and schedules, with the

Chu duality, can be converted into one another. Both automata and schedules have applications in different fields of computer science such as hardware design and analysis of asynchronous circuits, network diagnostics, distributed computation and logics of programs.

## 2.3 Summary

When considering the classification of concurrency models of Winskel et al. [34], we have that models are either simulating concurrency, known as *interleaving*, or dealing directly with concurrency, namely *non-interleaving*. The difference being that mutually exclusive occurrences are not distinguishable from truly independent ones in the former, but distinguishable in the latter.

Interleaving models are not able to distinguish mutually exclusive events from truly independent events because the models depend on which events a process takes to be *atomic* [42, page 4]. When we say atomic, we mean an indivisible action that must complete without interruption, in other words be considered instantaneous. If we suppose that an atomic event can be *refined* in the same manner as how Pratt explains refinement of atomic actions in [41], then the interleaving model can be seen to not have interleaved subactions. We will consider such a refinement as *action refinement*. Action refinement is the method of developing a system by starting from a abstract specification, and gradually refining its action by providing more details [35, 19].

In the literature, it has been shown that interleaving models are not well behaved under action refinement [19, 15]. The model is forced to commit itself in advance to a particular level of granularity. For example, a programmer may give details on actions that, at the time, are regarded as atomic. However, at a future release the programmer may discover that the actions have substructures, see [19, Example 1.1]. The programmer cannot refine the action since it is atomic. There might be occurrences where the interleaving model can be seen not to have interleaved subactions. The failure to interleave subactions is not characteristic of non-interleaving concurrency, but rather of a hidden assumption of excluded middle, or *mutual exclusion* as it is more commonly known in concurrent processes [40]. We consider mutual exclusion to be where no actions can happen simultaneously, and must occur after each other.

From our definition of transition systems, we are only able to simulate the parallel execution of two actions as the "*interleaving*" of *ab* or *ba*, see Figure 2.1. In the next chapter we will introduce asynchronous transition systems which extends the transition system to include an independence relation that is equipped to distinguish the mutual exclusion and independent occurrence of two events.

# Chapter 3

# An introduction to non-interleaving models of concurrency

In this chapter we present non-interleaving models of concurrency that generalize the independence of two, or more, events. We present a geometric model of concurrency which is an algebraic structure able to capture the main characteristic of *both* interleaving and non-interleaving models. We show both a state-based model, higher-dimensional automata, and an event-based model, ST-structures, capable of distinguishing an arbitrary number of events. Also, we present Chu spaces as a model able to present state-based and event-based models symmetrically, such that there is a way to present the event-state duality symmetrically.

The models mentioned so far have been based on interleaving computation steps to capture all the possible behaviours of a concurrent system. Recall the interleaving square, in Figure 2.1, where we have four states $s_1$, $s_2$, $s_3$ and $s_4$ and four transitions, where two of the transitions are labeled $a$ and the other two transitions are labeled $b$. We see that these two actions $a$ and $b$ can be executed in either order, ending in the same state each way, such that $a$ and $b$ are mutually exclusive. By design, interleaving models depend on the notion of *atomic* actions, that is, actions that are indivisible.

The assumption of atomic actions is reasonable for events that are mutually exclusive, meaning they cannot occur simultaneously. For example, a vending machine able to provide a single item at a time, like a snack or a beverage. If we want two items, such as a chocolate and a beverage, from the vending machine, then either the chocolate will be released followed by the beverage or first the beverage followed by the chocolate. However, for events that are truly independent, such as a vending machine able to provide two, or more, items at a time, then their order of occurrence would be rendered irrelevant − two items would be released at the same time from the vending machine. Models that only allow interleaving are unable to distinguish these two vending machine. Instead, we will consider the non-interleaving models of concurrency, such as *asynchronous transition systems* [49] and *higher-dimensional automata* [41], *ST-structures* [28] and *Chu spaces* [27], which are able to distinguishing between these two vending machines. Thus, we allow receiving multiple items after each other, or all at the same time. Going further, we

may even have a combination of the alternatives.

For didactic and historical reasons, we will review asynchronous transition systems where we introduce an independence relation between the occurrence of events. The independence relation is able to express the mutual exclusion and non-interleaving of two events. Asynchronous transition systems may be considered a bridge between transition systems, used up until now, and the higher-dimensional automata, introduced in Section 3.2. Higher-dimensional automata are expressive and can encode the independence of an arbitrary number of events, rather than only pairs of events.

Higher-dimensional automata are *algebraic structures* which can be interpreted geometrically, roughly as topological spaces with a sense of direction to incorporate a notion of irreversible time. A topological space with a sense of direction makes it possible to determine the order of an execution. Intuitively, a higher-dimensional automaton is an automaton with nicely incorporated squares, cubes and higher-dimensional cubes, which represent the independence of events. A main characteristic of higher-dimensional automata is to be able to capture what happens *during* an execution. ST-structure is the event-based counterpart of higher-dimensional automata which is able to precisely identify events, and also capture what happens during an execution. As shown by Johansen in [28], higher-dimensional automata cannot precisely identify events. For example, the asymmetric conflict shown in [28, Figure 5] and presented in Section 4.1. However, we may faithfully represent the asymmetric conflict as a ST-structure.

## 3.1 Asynchronous transition systems

Asynchronous transition systems were introduced independently by Bednarczyk [1] and Shields [46]. They extend transition systems by including a *binary* independence relation between the occurrence of events. In this model, transitions are events bearing this independence relation. Asynchronous transition systems can be viewed as asynchronous graphs, whose vertices are states and whose edges are transition between states. Transitions take the commutation between events into consideration [9, Section 3.3]. If two events commute, then these may be executed at the same time. Meaning that we can consider two transitions of events as a single transition with two events.

**Definition 3.1** (Asynchronous transition system [49])**.** *An* asynchronous transition system *is a tuple* $(\mathcal{S}, i, E, \mathcal{I}, Tran)$ *where*

- $(\mathcal{S}, i, E, Tran)$ *is a transition system and*
- $\mathcal{I} \subseteq E \times E$ *is an irreflexive symmetric relation, called the independence relation, such that:*

  1. $a \in E \Rightarrow \exists s, s' \in \mathcal{S}, (s, a, s') \in Tran.$

2. $(s, a, s') \in Tran \wedge (s, a, s'') \in Tran \Rightarrow s' = s''$.
3. $a\mathcal{I}b \wedge (s_1, a, s_2) \in Tran \wedge (s_1, b, s_3) \in Tran \Rightarrow \exists u, (s_2, b, u) \in Tran \wedge (s_3, a, u) \in Tran$.
4. $a\mathcal{I}b \wedge (s_1, a, s_2) \in Tran \wedge (s_2, b, u) \in Tran \Rightarrow \exists s_3, (s_1, b, s_3) \in Tran \wedge (s_3, a, u) \in Tran$.

Condition (1) says every event should appear as a transition. Condition (2) says that transitions with an event from a state has to be unique, meaning that the transition system should be deterministic. Condition (3) and (4), are pictured respectively in Figure 3.1 and 3.2, exhibiting the transitions of events bearing the independence relation that deals with independent events.

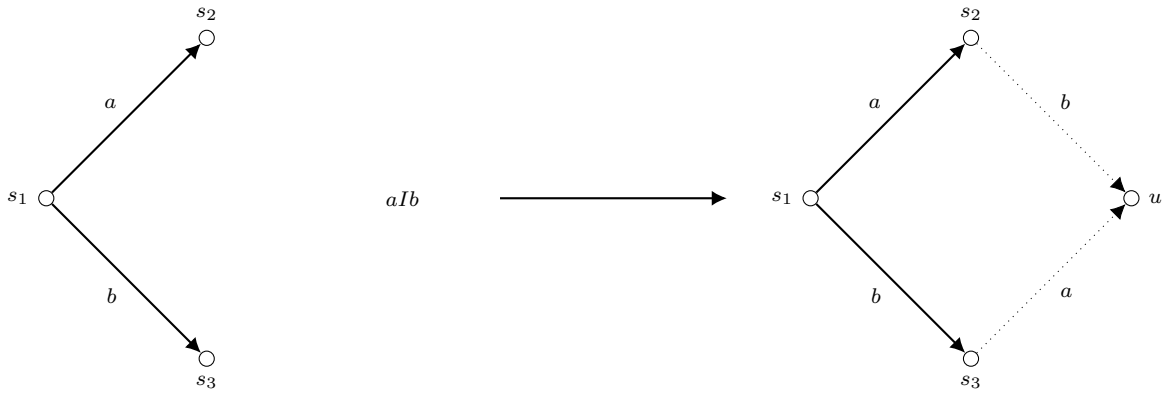



**Figure 3.1** Condition (3) for asynchronous transition systems, where the right-hand side shows how to interpret two independent events coming from a common state. If two events, $a$ and $b$, occur independently from a common state $s$, then they should be able to form the right-hand side.

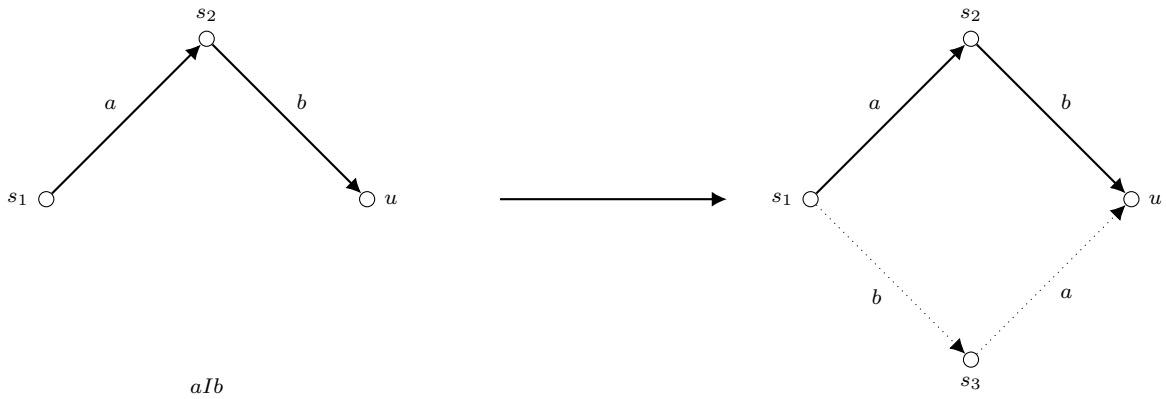



**Figure 3.2** Condition (4) for asynchronous transition systems, where the right-hand side shows how to interpret two independent events occurring immediately after each other. If two events, $a$ and $b$, occur one immediately after the other, then they should be able to occur with their order interchanged.

Morphisms between asynchronous transition systems are morphisms between their underlying transition systems which preserve the additional relations of independence.

**Definition 3.2** (Morphisms of asynchronous transition system [49])**.** *Let* $\mathcal{T} = (\mathcal{S}, i, E, \mathcal{I}, Tran)$ *and* $\mathcal{T}' = (\mathcal{S}', i', E', \mathcal{I}', Tran')$ *be asynchronous transition systems. A morphism* $\mathcal{T} \to \mathcal{T}'$ *is a morphism of transition systems*

$$(\sigma, \lambda) : (\mathcal{S}, i, E, Tran) \to (\mathcal{S}', i', E', Tran')$$

*such that*

$$a \mathcal{I} b \text{ and } \lambda(a), \lambda(b) \text{ are both defined } \implies \lambda(a) \mathcal{I} \lambda(b).$$

Morphisms of asynchronous transition systems compose as morphisms between their underlying transition systems. We can turn asynchronous transitions systems into a category, written $\mathbb{ATS}$. The category of asynchronous transition systems restricted over the events $E$ is named $\mathbb{ATS}_E$.

In interleaving models for concurrency, the number of executions grow exponentially with the size of the program. Every possible execution of a program must be considered, which is not feasible in practice. Non-interleaving models take the commutation of instructions into account, that is, allowing instructions to be executed at the same time. By commuting instructions, it minimizes the number of possible executions of a program by considering certain instructions to be equivalent.

From the independence relation, we can observe that many of the schedules are equivalent in the sense that one can be obtained from the other by permuting independent instructions. Furthermore, such equivalent executions will always lead to the same result. Consequently, if one of those executions can be shown not to lead to an error, neither will any other execution which is equivalent to it.

An independence relation is able to deal directly with concurrency. However, the relation is binary meaning that at most two events can be distinguished. To be able to distinguish an arbitrary number of events, we could extend the independence relation to be an *n-ary* relation. However, this relation should be interpreted as a tractable algebraic theory, which has shown to be challenging.

## 3.2 Higher-dimensional automata

The generalization of independence to $n$ events has lead to a geometric model for concurrency, studied by Pratt and van Glabbeek [41, 40, 13]. Pratt named this model

Higher-Dimensional Automata (HDA) [41]. With higher-dimensional automata we can distinguish between the execution of $n$ non-interleaving concurrent events and of $n$ mutually exclusive events. Higher-dimensional automata are considered to be of high expressive power because of their property to distinguish $n$-events [29]. Hence, providing a generalization of differences, and common features, of various other models of concurrency, as done in [13] and [25]. Higher-dimensional automata also retain the state-based view and model the interleaving square with its interior filled, shown in Figure 3.5.

A higher-dimensional automaton is a precubical set that encodes the independence of events by *mappings*. Intuitively, we may consider a precubical set as $n$-dimensional cubes [9] together with their faces − each $n$-dimensional cube has $2n$ faces. In other words, a front and a back face in each direction $i$ with $0 \leq i < n$. Mappings are families of functions that identify these faces, we call these *face maps*. Face maps that satisfy the *precubical identity* [3, 5, 7, 22, 21] are able to interpret the independence of events. In the concurrency theory literature, we find that the precubical identity is named the *cubical law* [13, 28], since applications of algebraic topology are still being investigated in concurrency theory. Here, we will follow the topological notion, and use the naming precubical identity.

Intuitively, the precubical identity is considered to be the idea of "*filling in holes*" of a $n$-dimensional cube [41, Section 2]. This notion is shown in Figure 3.5, where the interior of the interleaving square is filled. With higher-dimensional automata we are able to capture the main characteristic of both transition systems and asynchronous transition systems by the notion of "*filling in holes*".

**Definition 3.3** (Precubical set)**.** *A* precubical set *is a graded set* $\mathcal{X} = \bigcup_{n\in\mathbb{N}} \mathcal{X}_n$*, with* $\mathcal{X}_n \cap \mathcal{X}_m = \emptyset$ *for* $n \neq m$*, together with mappings* $s_{k,n}, t_{k,n} : \mathcal{X}_n \to \mathcal{X}_{n-1}$*, with* $1 \leq k \leq n$*, satisfying the* precubical identities

$$\alpha_{k,n-1}\beta_{\ell,n} = \beta_{\ell-1,n-1}\alpha_{k,n} \qquad (1 \leq k < \ell \leq n)$$

*for* $\alpha, \beta \in \{s, t\}$.

A *graded* set is a *family of sets* where its elements are $n$-cells. A *family of sets* is the disjoint union of these $n$-cells, $\mathcal{X}_n \cap \mathcal{X}_m = \emptyset$ for $n \neq m$. In the beginning of this Section, we called these $n$-cells for $n$-dimensional cubes. The naming $n$-dimensional cubes comes from Fajstrup et al. [9], where $n$-dimensional cubes are precubical sets together with their faces. In this thesis, we will call elements of $\mathcal{X}_n$ for $n$-cells, or simply cells, even though there are different names for these elements such as *n-dimensional cubes* [40, 9], *n-transitions* [20] and *hypercubes* [13]. If $x \in \mathcal{X}_n$, then we say that $x$ is of dimension $n$ and is written $dim x = n$.

The mappings $s_{k,n}$ and $t_{k,n}$ are called *face maps*, and we will usually omit the extra subscript $n$ and simply write $s_k$ and $t_k$. Face maps are families of functions that identify the faces of the $n$-cells, in the same manner as described with $n$-dimensional cubes. Families of functions are usually considered as a set of functions whose

equations have a similar form. However, in this context we will interpret them geometrically as faces of a cube.

Each $n$-cell $x \in \mathcal{X}_n$ has *n lower faces* $s_1x, \ldots, s_nx$ and *n upper faces* $t_1x, \ldots, t_nx$, and the precubical identity expresses the fact that $(n-1)$-faces of an $n$-cell meet in common $(n-2)$-faces, see Figure 3.4.

Figure 3.3 and 3.4 represent a 2-cell showing both the four faces $s_1x$, $t_1x$, $s_2x$, $t_2x$ and the four possible mappings of the precubical identity. Figure 3.3 is more of a geometrical picture of the 2-cell with its interior filled.

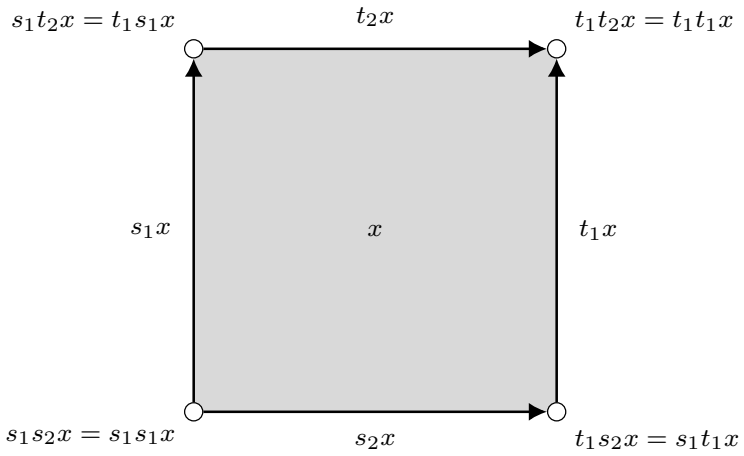



**Figure 3.3** We have a geometrical picturing of the 2-cell with its interior filled, four faces $s_1x$, $t_1x$, $s_2x$, $t_2x$ and four corners representing the mappings of the precubical identity.

While, Figure 3.4 shows how the four mappings of the precubical identity is satisfied, and the fact that 1-faces of a 2-cell meet in common 0-faces.

We need a notion of *morphisms* between precubical sets to be able to study the relationship between higher dimensional automata. From Section 2.1, we have that morphisms are considered to be simulations. Morphisms as simulation was an idea from Winskel and Nielsen in [49]. We want to extend morphisms to be structure-preserving maps that preserve orientation, shape, and time [20, Section 2.2].

Preserving orientation is addressed by *directed* topology, where the object of study are topological spaces that have a sense of direction. Specifically, a topological space with a directed variant to incorporate the notion of irreversible time. Topology is a branch in mathematics that studies geometric shapes, and directed topology considered these geometric shapes to have a sense of direction. Shapes are preserved if we can present the geometry, we are interested in, as unions of *points, segments, squares, cubes, ..., n-dimensional cubes* as collections of *n-cells* ($n \in \mathbb{N}$). Preserving time means that we can reason about the directed variant and that every transition is mapped onto a transition.

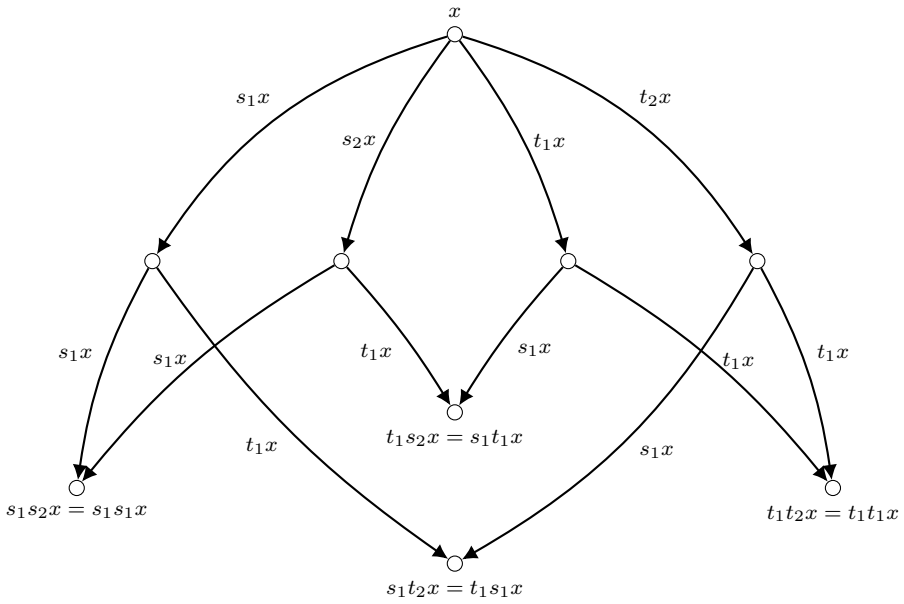



**Figure 3.4** We have a 2-cell showing the four faces $s_1x$, $t_1x$, $s_2x$, $t_2x$ as edges and the four mappings of the precubical identity as the bottom nodes of the 2-cell.

**Definition 3.4** (Morphism of precubical sets)**.** *Morphisms $f : \mathcal{X} \to \mathcal{Y}$ of precubical sets are graded functions $f = \{f_n : \mathcal{X}_n \to \mathcal{Y}_n\}_{n\in\mathbb{N}}$ which commute with the face maps:*

$$\alpha_k \circ f_n = f_{n-1} \circ \alpha_k \qquad \text{for all } n \in \mathbb{N},\ k \in \{1,\dots,n\},\ \text{and } \alpha \in \{s,t\}.$$

Higher-dimensional automata are simply defined in terms of the precubical sets, and the relationship between higher-dimensional automata is described by the morphisms of the underlying precubical sets. Every figure presented thus far, may be considered as a higher-dimensional automata. We will from now on refer to the previous figures as higher-dimensional automata even though we first introduced them as transition systems, asynchronous transition systems and $n$-cells.

**Definition 3.5** (Higher-dimensional automata [13, 28])**.** *A precubical set $\mathcal{H} = (\mathcal{Q}, s, t)$ is a precubical set $\mathcal{Q}$ together with $s$ and $t$ being the collection of all the face maps, that is, for all $n$.*

*A higher-dimensional automata is a tuple $(\mathcal{Q}, s, t, l, \mathcal{I}, \mathcal{F})$ over an alphabet $\Sigma$ such that $l(s_i(q)) = l(t_i(q))$ for all $q \in \mathcal{Q}_2$ and $i \in \{1,2\}$, and with a designated initial cell $\mathcal{I} \in \mathcal{Q}_0$ and $\mathcal{F} \subseteq \mathcal{Q}_0$ final cells.*

The concurrent execution of a higher-dimensional automata is modeled by including the two-dimensional surface. Intuitively, a concurrent execution can be seen as moving across the surface such that the execution preserves the directed variant, that is, to incorporate a notion of irreversible time. This is pictured in Figure 3.5.

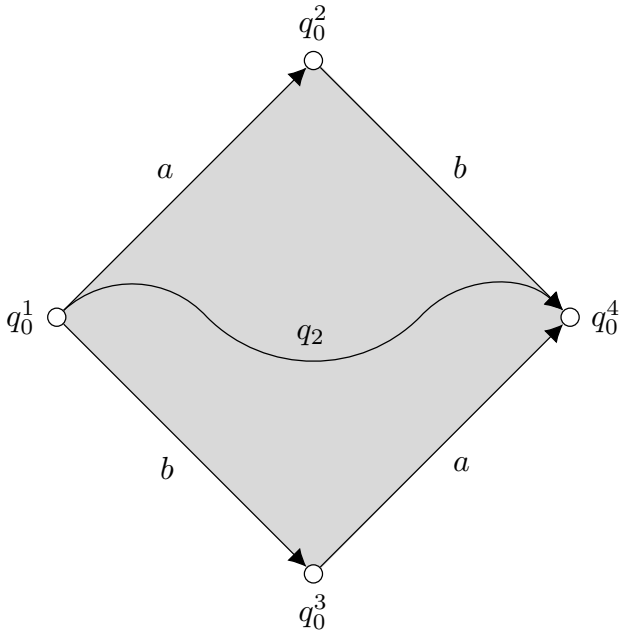



**Figure 3.5** Example of a HDA with four states, $q_0^1, q_0^2, q_0^3$ and $q_0^4$, four transitions, two of which are labeled $a$ and the other two are labeled $b$, and a surface, $q_2$. The surface represents the notion of "filling in holes" [41, Section 2], which is a continuous deformation of instructions $ab$ into $ba$, or vice versa.

A computation is a path in this higher-dimensional automaton.

**Definition 3.6** (Paths in HDAs [28])**.** *A* single step *in a HDA is either*

1. $q_{n-1} \xrightarrow{s_i} q_n$ *with* $s_i(q_n) = q_{n-1}$ *or*
2. $q_n \xrightarrow{t_i} q_{n-1}$ *with* $t_i(q_n) = q_{n-1}$,

*where* $q_n \in \mathcal{Q}_n$ *and* $q_{n-1} \in \mathcal{Q}_{n-1}$ *and* $1 \leq i \leq n$.

*A* path $\pi \triangleq q^0 \xrightarrow{\alpha^1} q^1 \xrightarrow{\alpha^2} q^2 \xrightarrow{\alpha^3} \dots$ *is a sequence of single steps* $q^j \xrightarrow{\alpha^{j+1}} q^{j+1}$*, with* $\alpha^j \in \{s, t\}$*. We say that* $q \in \pi$ *iff* $q = q^j$ *appears in one of the steps in* $\pi$*. The first cell in a path is denoted* $st(\pi)$ *and the ending cell in a finite path is* $en(\pi)$*.*

The markings of the steps by $s$ may be seen as going from a lower cell to a higher cell, and steps by $t$ as the opposite, in the higher dimensional automaton. In many of the later proofs it is though useful to have the exact map easily visible, namely, the index that the step uses, rather than explicitly assuming the index every time. If there is no $en(\pi)$, then the path is infinite. When the index is not important we write $\xrightarrow{s}$ or $\xrightarrow{t}$.

**Definition 3.7** (Histories for HDAs [28])**.** *In a* HDA *two paths are* adjacent*, denoted* $\pi \overset{adj}{\longleftrightarrow} \pi'$*, if one can be obtained from the other by replacing, for* $q, q' \in \mathcal{Q}$ *and* $i < j$*,*

1. *a segment* $\xrightarrow{s_i} q \xrightarrow{s_j}$ *by* $\xrightarrow{s_{j-1}} q' \xrightarrow{s_i}$*, or*
2. *a segment* $\xrightarrow{t_j} q \xrightarrow{t_i}$ *by* $\xrightarrow{t_i} q' \xrightarrow{t_{j-1}}$*, or*
3. *a segment* $\xrightarrow{s_i} q \xrightarrow{t_j}$ *by* $\xrightarrow{t_{j-1}} q' \xrightarrow{s_i}$*, or*
4. *a segment* $\xrightarrow{s_j} q \xrightarrow{t_i}$ *by* $\xrightarrow{t_i} q' \xrightarrow{s_{j-1}}$*.*

*Two finite paths are l-adjacent* $\pi \overset{l}{\longleftrightarrow} \pi'$ *when the segment replacement happens at position* $l + 1$*; that is, q is the* $l + 1$ *cell in the path.* Homotopy *is the reflexive and transitive closure of adjacency. Two homotopic paths are denoted* $\pi \overset{hom}{\longleftrightarrow} \pi'$ *and share their respective start and end cells. The homotopy class of a rooted path is denoted* $\overleftrightarrow{[\pi]}$*. A homotopy class with end cell q is said to be* a history of *q. One cell may have several histories, as is the case with the interleaving square* HDA *from Figure 3.5. Whenever a cell has a unique history we use the notation* $\overleftrightarrow{[q]}$*, instead of* $\overleftrightarrow{[\pi]}$ *with* $en(\pi) = q$*.*

Homotopy is defined for all paths such that a cell of higher dimension has a history in the same way that the inside of a square has a history. The homotopy is provided by Johansen in [28], where the homotopy is different compared to the definition in [25, Section 1.6] because not only the state cells of dimension 0, that is, vertices that form the corners of a cube, have histories, but also cells of higher dimensions.

History unfoldings of process graphs have been defined by Van Glabbeek in [12, Section 3]. Inspired by the definition of history unfoldings of process graphs, we will define the same notion for higher-dimensional automata. Johansen in [28], provides a definition of history unfolding for higher-dimensional automata which can be correlated with the definition of unfoldings from [12]. Also, alternative definitions of unfoldings for HDA can be found in [3, 4].

**Definition 3.8** (History unfolding for HDAs)**.** *The* history unfolding *of a higher dimensional automaton* $\mathcal{H}$ *is a* HDA *denoted* $\mathsf{U}(\mathcal{H})$*, respecting the cubical laws, as shown in [28, Proposition 3.30], and given by:*

- $Q_n^{\mathsf{U}(\mathcal{H})}$ *is the set of histories that end up in cells on level* $Q_n$ *of* $\mathcal{H}$*;*
- *has the labelling copied from* $\mathcal{H}$*:* $l^{\mathsf{U}(\mathcal{H})}(\overleftrightarrow{[\pi]}) = l^H(en(\pi))$*;*
- *initial cell the empty rooted history;*

- *the $s/t$ maps are built from the corresponding maps between the end cells of the histories:*

$$s_i(\overleftrightarrow{[\pi]}) = \overleftrightarrow{[\pi']} \quad \textit{iff} \quad s_i(q) = q' \wedge \pi' \xrightarrow{s_i} \pi \wedge en(\pi') = q' \wedge en(\pi) = q;$$

$$t_i(\overleftrightarrow{[\pi]}) = \overleftrightarrow{[\pi']} \quad \textit{iff} \quad t_i(q) = q' \wedge \pi \xrightarrow{t_i} \pi' \wedge en(\pi') = q' \wedge en(\pi) = q.$$

The notion of *unfolding* removes iteration and is commonly used to turn a complicated model such as higher-dimensional automata into a simpler, but potentially infinite one. Unfolding is important in relation to the sculpting method introduced in Chapter 5. For HDA this picture is more complicated. Figure 5.1 (left) shows a simple HDA which is a sculpture. However, we will show that its unfolding (right) is *not* a sculpture.

**Definition 3.9** (Morphism of HDAs [28])**.** *A morphism between two* HDA*s, $f : \mathcal{H} \to \mathcal{H}'$ is a dimension preserving map between their cells $f : \mathcal{Q} \to \mathcal{Q}'$, such that:*

1. *the initial cell is preserved: $f(\mathcal{I}) = \mathcal{I}'$*
2. *the labelling is preserved: $l'(f(q_1)) = l(q_1)$ for all $q_1 \in \mathcal{Q}_1$,*
3. *the mappings are preserved, for any $q_n \in \mathcal{Q}_n$ and $1 \leq i \leq n$:*
   - *$s'_i(f(q_n)) = f(s_i(q_n))$ and*
   - *$t'_i(f(q_n)) = f(t_i(q_n))$.*

*When a morphism is bijective we call it isomorphism. Two HDAs are isomorphic, denoted $\mathcal{H} \cong \mathcal{H}'$, when there exists an isomorphism between them.*

We write $\mathbb{HDA}$ for the category of higher-dimensional automata, where the object of the category are higher-dimensional automata and morphisms are as defined in Definition 3.9.

In the definition of precubical sets, we consider them to be non-degenerate. Many of the results from this thesis assumes higher-dimensional automata to be non-degenerate. Also, we consider the directed variant, mentioned so far, to be higher-dimensional automata that are acyclic. Johansen in [28], defines these notions for higher dimensional automata:

**Definition 3.10** (Acyclic and non-degnerate HDAs [28])**.** *A higher-dimensional automata is called acyclic if no path visits a cell twice. A higher-dimensional automata is non-degenerate if for any cell $q$ all its faces exist and are different, in the sense of $\forall i \neq j : \alpha_i(q) \neq \beta_i(q) \wedge \alpha, \beta\{s, t\}$, no two transitions with the same label share both their end states.*

The notion of acyclic does not allow cycles, or loops, in the higher-dimensional automata, and the notion of non-degeneracies is related to the underlying precubical sets[1]. The restriction on higher-dimensional automata being non-degenerate is similar to that of Fajstrup et al. [9] and that of van Glabbeek [13]. Also, the restriction above is the same for precubical sets where two opposite s-maps and t-maps can be equal, such that $s_i(q) = t_i(q)$ is allowed. We assume all s-maps and t-maps are to be total, as done in [28]. A study of partial higher-dimensional automata, where s-maps and t-maps are partial, has been considered by Fahrenberg in [4].

## 3.3 ST-structures

ST-structures are first introduced by Johansen in [28]. They are an extension of *configuration structures* [16] and *unrestricted* event structures [17].

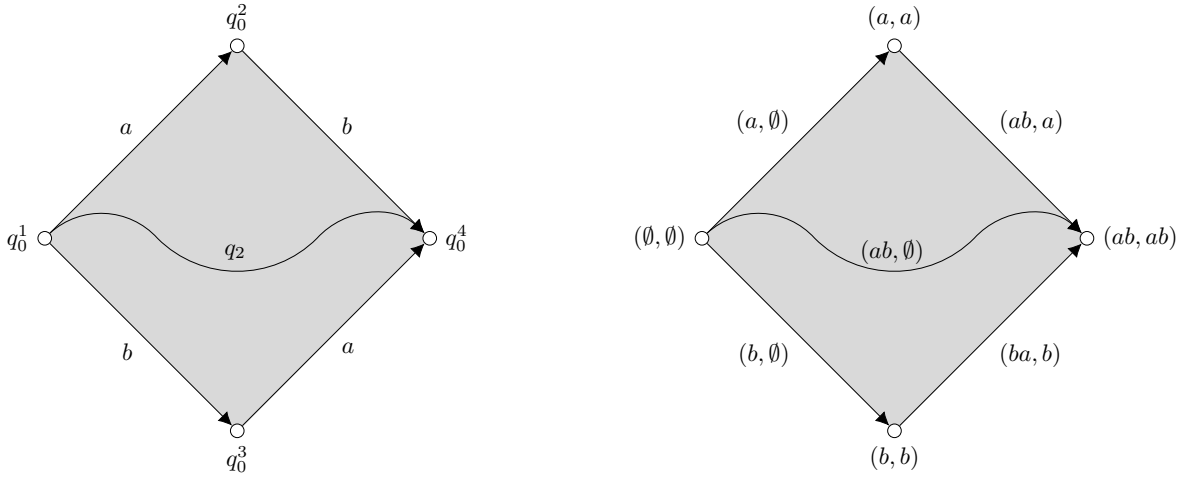



**Figure 3.6** Example of a higher-dimensional automaton with two concurrent events labelled by $a$ and $b$: with a geometrical picture of the higher-dimensional automaton (left) and of the ST-structure (right).

In the case of configuration structures and event structures, we have that the classical notion of concurrency, causality, and conflict are not interdefinable. In the case of higher-dimensional automata these notions can be interdefinable. In Chapter 4, we relate ST-structures to higher-dimensional automata by identifying a corresponding class of ST-structures with the particular property of *adjacent-closure*. With the adjacent-closure property, ST-structures are able to capture the higher-dimensional case of the notions of concurrency, causality and conflict. We will use the same notation and definitions as Johansen in [28].

[1] In topology, The distinction between non-degeneracies and degeneracies are subtle and will not be considered here. In [20], the distinction between non-degeneracies and degeneracies is considered.

**Definition 3.11** (ST-configuration [28]). *An ST-configuration over some set $E$ of events is a pair of finite sets $(\mathcal{S}, \mathcal{T})$ $(\mathcal{S}, \mathcal{T} \subseteq E)$ respecting the property:*

$$\textit{(start before terminate) } \mathcal{T} \subseteq \mathcal{S}.$$

In the following, $\mathcal{S}$ contains the events that have started and $\mathcal{T}$ the events that have terminated. In the current ST-configuration, we see the events $\mathcal{S} \setminus \mathcal{T}$ as being executed *concurrently*. We will call $|\mathcal{S} \setminus \mathcal{T}|$ the concurrent degree of this ST-configuration, similar to how we defined dimensions for higher-dimensional automata. The notion of degree, and dimension, is the main characteristic captured by ST-structures which is found in higher-dimensional automata, but not found in classical event-based models like configuration structures or event structures. In ST-structures, an event can be seen to have started and not terminated yet, capturing the notion of *duration*.

**Definition 3.12** (ST-structures [28]). *An ST-configuration structure (also called ST-structure) is a tuple $ST = (E, ST, l)$ with $ST$ a set of ST-configurations over $E$ satisfying the constraint:*

$$\textit{if } (\mathcal{S}, \mathcal{T}) \in ST \textit{ then } (\mathcal{S}, \mathcal{S}) \in ST,$$

*and $l : E \to \Sigma$ a labelling function with $\Sigma$ the set of labels. we often omit the set of events $E$ from the notation when there is no danger of confusion.*

Intuitively, the constraint above ensures that every event that is represented, and that has also started, has to be terminated. We denote $\mathbb{ST}$ as the set of all ST-structures.

**Definition 3.13** (Stable ST-structures [28]). *A ST-structure $(E, ST, l)$ is called:*

1. *rooted iff $(\emptyset,\emptyset) \in ST$;*
2. *connected iff for any non-empty $(\mathcal{S}, \mathcal{T}) \in ST$, either $\exists e \in \mathcal{S} : (\mathcal{S} \setminus e, \mathcal{T}) \in ST$ or $\exists e \in \mathcal{T} : (\mathcal{S}, \mathcal{T} \setminus e) \in ST$;*
3. *closed under bounded unions iff for any $(\mathcal{S}, \mathcal{T})$,$(\mathcal{S}', \mathcal{T}')$,$(\mathcal{S}'', \mathcal{T}'') \in ST$ if $(\mathcal{S}, \mathcal{T}) \cup (\mathcal{S}', \mathcal{T}') \subseteq (\mathcal{S}'', \mathcal{T}'')$ then $(\mathcal{S}, \mathcal{T}) \cup (\mathcal{S}', \mathcal{T}') \in ST$;*
4. *closed under bounded intersection iff for $(\mathcal{S}, \mathcal{T})$,$(\mathcal{S}', \mathcal{T}')$,$(\mathcal{S}'', \mathcal{T}'') \in ST$ if $(\mathcal{S}, \mathcal{T}) \cup (\mathcal{S}', \mathcal{T}') \subseteq (\mathcal{S}'', \mathcal{T}'')$ then $(\mathcal{S}, \mathcal{T}) \cap (\mathcal{S}', \mathcal{T}') \in ST$;*

*a ST-structure is called stable iff it is rooted, connected, and closed under bounded unions and intersections.*

Stable ST-structures satisfy the properties of being rooted, connected, and closed under bounded unions and intersections. Rooted is where the ST-structure has a ST-configuration that has no events running or terminated. Connected means that for every non-empty configuration, that is, an event has terminated or is running, there must exist another configuration where that event has not started, or terminated, yet. Both closed under bounded union and intersection, are operations that provide configurations that already exist in the ST-structure.

**Definition 3.14** (Stable ST-structures [28])**.** *A step between two ST-configurations is defined as either:*

**s-step** $(\mathcal{S},\mathcal{T}) \xrightarrow[s]{e} (\mathcal{S}',\mathcal{T}')$ *when* $\mathcal{T}=\mathcal{T}', \mathcal{S}\subset\mathcal{S}', \mathcal{S}'\setminus\mathcal{S}=\{e\}$*; or*

**t-step** $(\mathcal{S},\mathcal{T}) \xrightarrow[t]{e} (\mathcal{S}',\mathcal{T}')$ *when* $\mathcal{S}=\mathcal{S}', \mathcal{T}\subset\mathcal{T}', \mathcal{T}'\setminus\mathcal{T}=\{e\}$.

*When the type is unimportant we denote a step by* $\xrightarrow{e}$ *for* $\xrightarrow[s]{e} \cup \xrightarrow[t]{e}$*. A* path *of a ST-structure, denoted* $\pi$*, is a sequence of steps, where the end of one is the beginning of the next, in other words,*

$$\pi \triangleq (\mathcal{S},\mathcal{T}) \xrightarrow{e} (\mathcal{S}',\mathcal{T}') \xrightarrow{e'} (\mathcal{S}'',\mathcal{T}'')\dots$$

*A path is* rooted *if it starts in* $(\emptyset,\emptyset)$*.*

A *computational interpretation* for ST-structures is defined by defining simple steps between ST-configurations, similar to interpretations for other concurrency models like configurations structures. However in [28, Theorem 3.10], the computational interpretation for ST-structures is shown to be more fine-grained than other models such as configuration structures and event structures, since ST-structures can capture the notion of *duration*.

The notion of duration is similar to what is captured in higher-dimensional automata, but from a state-based perspective. Hence, ST-structures are capable of showing a natural sequence of *observable information* as ST-traces in a similar manner as higher-dimensional automata [13, Section 7.3]. Following the notion of Johansen in [28], we may consider ST-structures the richest formalisation of observable content for concurrent systems.

**Definition 3.15** (Paths and traces [28])**.** *A path of a ST-structure, denoted* $\pi$*, is a sequence of steps, where the end of one is the beginning of the next, that is,*

$$\pi \triangleq (\mathcal{S}_0,\mathcal{T}_0) \xrightarrow{a} (\mathcal{S}_1,\mathcal{T}_1) \xrightarrow{b} (\mathcal{S}_2,\mathcal{T}_2)...$$

*A path is rooted if it starts in (∅,∅). The ST-trace of a rooted path $\pi$, denoted $st(\pi)$, is the sequence of labels of the steps of $\pi$ where each label is annotated as $a^0$ if it labels an s-step or as $a^n$ if it labels a t-step. where $n \in \mathbb{N}$ is determined by counting from the beginning the number of steps until the s-step that has added the event* e *to the $\mathcal{S}$ set, which* e *being the event that has been added to $\mathcal{T}$ in the current t-step.*

If we consider rooted and connected ST-structures, then the notion of ST-trace matches with the one in [18, Definition 2.5] and in [13, Section 7.3]. Below we define the notion of concurrency and causality for a particular ST-configuration as Johansen in [28].

**Definition 3.16** (Concurrency and causality [28])**.** *For a particular ST-configuration $(\mathcal{S},\mathcal{T}) \in \mathsf{ST}$ where $e, e' \in \mathcal{S}$ satisfies*

- ***Concurrency:*** *$e \parallel e'$ iff there exists $(\mathcal{S}',\mathcal{T}') \subseteq (\mathcal{S},\mathcal{T})$ such that*
  - *$(\mathcal{S}',\mathcal{T}') \in \mathsf{ST}$, and*
  - *$\{e, e'\} \subseteq \mathcal{S}' \setminus \mathcal{T}'$.*
- ***Causality:*** *$e < e'$ iff $e \neq e'$ and for any $(\mathcal{S}',\mathcal{T}') \subseteq (\mathcal{S},\mathcal{T})$ such that*
  - *$(\mathcal{S}',\mathcal{T}') \in \mathsf{ST}$, and*
  - *$e' \in \mathcal{S}' \Rightarrow e \in \mathcal{T}'$.*

*Concurrency* for ST-structures is satisfied if we can find a ST-configuration where two events have started, not yet terminated, and is persistent throughout an execution. Other event-based models such as event structures represent concurrency as having events that are not in conflict and not partial ordered. ST-configurations represent concurrency by providing information about the currently concurrent events.

*Causality* for an ST-configurations is similar to the way causality is defined as a partial order for configuration structures, such that causality is a local definition for an ST-configuration.

For an arbitrary ST-structure, concurrency and causality are not interdefinable such that concurrency is the complement of causality, as in the standard way [15, Definition 5.6]. If we consider well behaved ST-structures such as stable ST-structures, then concurrency and causality can be seen as interdefinable in the standard way.

**Definition 3.17** (Conflict [28])**.** *For a ST-structure ST the notion of global conflict is defined as a predicate over sets of events $E' \subseteq E$:*

$$\#E' \text{ iff } \nexists(\mathcal{S},\mathcal{T}) \in ST \text{ with } E' \subseteq \mathcal{S}.$$

In the following, the ST-structure describes conflicting events as a predicate where conflicting events in a ST-structure can never appear in the same configuration.

Conflicting events is defined as a general notion on the whole ST-structure, and is considered to be similar to the standard notion of binary conflict for event structures [49].

**Definition 3.18** (Adjacent-closure [28]). *We call a ST-structure ST adjacent-closed if the following are respected:*

1. *if* $(\mathcal{S},\mathcal{T}),(\mathcal{S}\cup e,\mathcal{T}),(\mathcal{S}\cup\{e,e'\},\mathcal{T})\in ST$, *with* $(e\neq e')\notin\mathcal{S}$, *then* $(\mathcal{S}\cup e',\mathcal{T})\in ST$;
2. *if* $(\mathcal{S},\mathcal{T}),(\mathcal{S}\cup e,\mathcal{T}),(\mathcal{S}\cup e,\mathcal{T}\cup e')\in ST$, *with* $e\notin S\wedge e'\notin\mathcal{T}\wedge e\neq e'$, *then* $(\mathcal{S},\mathcal{T}\cup e')\in ST$;
3. *if* $(\mathcal{S},\mathcal{T}),(\mathcal{S}\cup e,\mathcal{T}),(\mathcal{S},\mathcal{T}\cup e')\in ST$, *with* $e'\notin\mathcal{S}\wedge e'\notin\mathcal{T}\wedge e\neq e'$, *then* $(\mathcal{S}\cup e,\mathcal{T}\cup e')\in ST$;
4. *if* $(\mathcal{S},\mathcal{T}),(\mathcal{S},T\cup e),(\mathcal{S},T\cup\{e,e'\})\in ST$, *with* $(e\neq e')\notin\mathcal{T}$, *then* $(\mathcal{S},\mathcal{T}\cup e')\in ST$;

In the following, adjacent-closure for ST-structure is related to the definition of higher-dimensional automata, in other words, the precubical identities of higher-dimensional automata. In the definition of *adjacent* [14, Definition 19], the correlation becomes more visible when looking at the homotopy over higher-dimensional automata. The homotopy over higher-dimensional automata essentially define histories of higher-dimensional automata, see Definition 3.7. Similarly, the above adjacent-closure on ST-structures makes sure that the histories of ST-configurations are represented.

**Definition 3.19** (Morphisms of ST-structures [28]). *A morphism* $f : ST \to ST'$ *between two ST-structures* $ST = (E, ST, l)$ *and* $ST' = (E', ST', l')$ *is defined as a partial function on the events,* $f : E \to E'$ *which:*

- *preserves ST-configurations, if* $(\mathcal{S},\mathcal{T})\in ST$ *then* $f(\mathcal{S},\mathcal{T})=(f(\mathcal{S}),f(\mathcal{T}))\in ST'$,
- *preserves the labelling when defined, that is,* $l'(f(e))=l(e)$ *if* $f$ *is defined for* $e$, *and*
- *is locally injective and total, that is, for any* $(\mathcal{S},\mathcal{T})\in ST$ *the restriction* $f_{\restriction S}$ *i s injective and total.*

Morphisms of ST-structures are structure-preserving maps such that if $\mathcal{T}\subseteq\mathcal{S}$ then $f(\mathcal{T})\subseteq f(\mathcal{S})$. If morphisms are not local and total, then morphisms are not guaranteed to preserve steps as shown by Johansen in [28, Proposition 2.21].

**Definition 3.20** (Isomorphic ST-structures [28]). *A function* $f$ *is an isomorphism of two ST-configurations* $(\mathcal{S},\mathcal{T})f(\mathcal{S}',\mathcal{T}')$ *iff* $f$ *is a bijection between* $\mathcal{S}$ *and* $\mathcal{S}'$ *that agrees on the sets* $\mathcal{T}$ *and* $\mathcal{T}'$ *(that is,* $f_{\restriction T}=\mathcal{T}'$*). Two ST-structures ST and ST are isomorphic, denoted* $ST\cong ST'$, *iff there exists a bijection* $f$ *on their events that is also a morphism between the two ST-structures.*

**Definition 3.21** (Category of ST [28]). *We can define a category* $\mathbb{ST}$ *to have objects ST-structures and the morphisms from Definition 3.19 because composition of morphisms is well defined for any ST-structure there exists a unique identity morphism which is the total function taking an event to itself.*

## 3.4 Chu Spaces

The model of Chu spaces has been developed by Gupta and Pratt [27, 38, 40] as an attempt to study the event-state duality as argued for by Pratt in [39]. It is a model that attempt to capture both the event-based and state-based aspect of a model symmetrically, where the translation between a schedule and automaton should be considered equally expressive.

Chu spaces represent processes as automata or schedules, and the Chu duality gives a simple way of translating between automata and schedules. A Chu space is a binary relation between two sets. The view of states and events are organized into two complementary spaces, namely, state space, *automata*, and event space, *schedules*.

**Definition 3.22** (Chu spaces). *A* Chu space *over K is any triple* $\mathcal{A}$ *= (E,r,X) where E and X are sets and* $r : E \times X \to K$ *is an arbitrary function called the* matrix *of the Chu space.*

A Chu space can be viewed as a matrix with entries from $K$, where rows represent the events from $E$ and columns represent the configurations in $X$. As an example, an entry $r((e, x)) = 0$ says that the event $e$ is not started yet in the configuration $x \in X$. In consequence, a Chu space can also be viewed as the structure $(E, X)$ where $X \subseteq K^E$.

Chu spaces can be viewed in various equivalent ways [27, Chapter 5]. For our setting, we take the view of $E$ as the set of events and $X$ as the set of configurations. The structure $K$ is representing the possible values the events may take. For example, $K = \mathbf{2} = \{0, 1\}$ is the classical case of an event being either not started (0) or terminated (1) where an order of $0 < 1$ would be used to define the steps in the system, that is, steps between states must respect the increasing order when lifted pointwise from $K$ to $X$.

Bringing higher-dimensional automata and ST-structures together is accomplished by restricting higher-dimensional automata while generalizing ST-structures. Higher-dimensional automata are restricted to their acyclic case, in other words, a single cube. ST-structures are generalized to capture the "*during*" aspect in the event-based setting, extending configuration structures with this notion. Therefore we need another structure $K = \mathbf{3} = \{0, \frac{1}{2}, 1\}$ with the order $0 < \frac{1}{2} < 1$, introducing the value $\frac{1}{2}$ to stand for *during*, or *in transition*. Chu space over **3** is also called a triadic Chu

space. Note that Gupta studied in [27] Chu spaces over **2**, whereas Pratt proposed to study Chu spaces over **3** and other structures in [43].

**Definition 3.23** (Morphism of Chu spaces)**.** *A morphism of Chu spaces (E,r,X) $\to$ (A,s,Y) is a pair of functions ($f : E \to A, g : Y \to X$) satisfying* adjointness *condition for all e and for all y:*

$$g(y)(e) = y(f(e)) \tag{3.1}$$

$$e(g(y)) = f(e)(y) \tag{3.2}$$

The category of Chu spaces, $\mathbb{CHU}$, has morphisms, called Chu transforms in [27, Chapter 4], between Chu spaces $(E, X)$ and $(A, Y)$ defined to be a pair $(f, g)$ of maps $f : E \to A$ and $g : Y \to X$ that satisfy the adjointness condition.

The method of "*sculpting*" is considered in regards to the Chu space approach to represent higher-dimensional automata. One starts with a single large, possibly infinite, dimensional cube and "*sculpts*" the desired process by removing unwanted faces. Initially, a cube constitutes the events to be a discrete and unstructured set, forming a complete cube. By removing states, the event set becomes structured. For example, sculpting two events by removing all the states distinguishing them renders them equivalent. The sculpting method point of view has been considered elsewhere in the higher-dimensional automata literature [24, 6].

In [40], Pratt describes the ways to specify concurrent processes by sculpting, composition, and transformation as follows:

> The programming as a sculpture: start from a sufficient large cube and hew out the desired process by chiseling away the unwanted states. While this view is attractively simple conceptually, it is not by itself a practical way of specifying a concurrent process. An alternative approach is composition, in which complex processes are built from smaller ones with suitable operators, including intrinsically concurrent operators such as asynchronous parallel composition. Yet another approach is transformation, in which new processes are constructed from old by reshaping them appropriately.
>
> These three activities, sculptures, composition, and transformation, are simultaneously compatible and complementary, and can therefore usefully be taken as a basis for concurrent programming. Very loosely speaking they correspond respectively to subalgebras, products, and homomorphisms, which play central and complementary roles in the algebraic approach to both logic and programming.

## 3.5 Summary

As presented asynchronous transition systems are able to deal directly with concurrency, however, the model is only able to distinguish two events. The geometric model, higher-dimensional automata, is a non-interleaving model with an algebraic structure able to distinguish an arbitrary number of events. Higher-dimensional automata are useful and general and able to capture the behaviour of other interleaving and non-interleaving models [25], such as transition systems and asynchronous transition systems [9]. However, higher-dimensional automata cannot precisely identify events, such as the asymmetric conflict shown in [28, Figure 5]. We will present identifying events in higher-dimensional automata with ST-structures in the next chapter. We will also use Chu spaces as a way to better understand the event-state duality of higher-dimensional automata and ST-structures.

# Chapter 4

# Relationships between non-interleaving models of concurrency

In this chapter we show the relationship between non-interleaving models of concurrency reviewed in the previous chapter. Specifically, we will relate higher-dimensional automata and ST-structures as well as ST-structures and Chu spaces. The relationship between Chu spaces and higher-dimensional automata has been investigated by Pratt in [39], and the relationship between ST-structures and higher-dimensional automata has been investigated by Johansen in [28]. One of our motivations is to investigate an open problem of [28] where there was no embedding between ST-structures and higher-dimensional automata. We attempt to investigate the relationships of the mentioned non-interleaving models of concurrency to better understand the event-state duality of Pratt [39].

Higher-dimensional automata are not good at identifying events as show in Example 4.2, but can be faithfully interpreted as ST-structures to identify events. In Example 4.2, isomorphic higher-dimensional automata are interpreted as non-isomorphic ST-structures. Hence, there is not an embedding, that is, an injective morphism, from $\mathbb{ST}$ to $\mathbb{HDA}$. We also show in Figure 4.2 that non-isomorphic higher-dimensional automata interpreted as ST-structures become isomorphic ST-structures, meaning there is not an embedding from $\mathbb{HDA}$ to $\mathbb{ST}$. Hence, higher-dimensional automata are neither more, or less, expressive than ST-structures.

We will consider Chu spaces as a way to reconcile the relationship between higher-dimensional automata and ST-structures, which was left as an open problem in [28]. Chu spaces are able to translate automata and schedules into each other by the Chu duality. Chu spaces are one response to the event-state duality, where automata and schedules are represented symmetrically.

Higher-dimensional automata and Chu spaces have been investigated by Pratt [39]. The relationship between ST-structures and Chu spaces has been investigated by Johansen [28]. However, we will present the relationship between ST-structures and Chu spaces to better answer the event-state duality question.

## 4.1 ST-structures and HDAs

For ST-structures to precisely identify events in higher-dimensional automata we need a way of translating ST-structures into higher-dimensional automata. Johansen in [28] provides a way of translating ST-structures into higher-dimensional automata as follows.

**Definition 4.1** ($\mathbb{ST}$ to $\mathbb{HDA}$ [28])**.** *We define a mapping* $\mathsf{H} : \mathbb{ST} \to \mathbb{HDA}$ *from ST-structures into* HDA*s which for an* $\mathsf{ST} = (E, ST, l)$ *with the events linearly ordered as a list* $\overrightarrow{E}$ *(that is, each event being indexed by a natural number, as in a sequence) returns the* HDA $\mathsf{H}(\mathsf{ST})$ *which*

- *has cells* $\mathcal{Q} = \{q^{(\mathcal{S},\mathcal{T})} \in \mathcal{Q}_n \mid (\mathcal{S},\mathcal{T}) \in ST \text{ and } |\mathcal{S} \setminus \mathcal{T}| = n\}$;
- *for any two cells* $q^{(\mathcal{S},\mathcal{T})}$ *and* $q^{(\mathcal{S}\setminus e,T)}$ *add the map entry* $s_i(q^{(\mathcal{S},\mathcal{T})}) = q^{(\mathcal{S}\setminus e,\mathcal{T})}$ *where* $i$ *is the index of the event* $e$ *in the listing* $\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$;
- *for any two cells* $q^{(\mathcal{S},\mathcal{T})}$ *and* $q^{(\mathcal{S},\mathcal{T}\cup e)}$ *add the map entry* $t_i(q^{(\mathcal{S},\mathcal{T})}) = q^{(\mathcal{S},\mathcal{T}\cup e)}$ *where* $i$ *is the index of the event* $e$ *in the listing* $\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$;
- *has labelling* $l(q^{(\mathcal{T}\cup e,\mathcal{T})}) = l(e)$ *for any* $q^{(\mathcal{T}\cup e,\mathcal{T})} \in \mathcal{Q}_1$.

*More precisely, by* $\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$ *we represent the listing of the events in* $\mathcal{S} \setminus \mathcal{T}$*, namely, a list of dimension* $|\mathcal{S} \setminus \mathcal{T}|$ *obtained from the original listing* $\overrightarrow{E}$ *by removing all other events. This new listing has the events of* $\mathcal{S} \setminus \mathcal{T}$ *in the same original order but with new indexes attached (ranging from* 1 *to* $|\mathcal{S} \setminus \mathcal{T}|$*).*

As shown in [28, Theorem 3.36], the mapping H associates a rooted, connected, and adjacent-closed ST-structure with $\mathsf{H}(\mathsf{ST})$, which is a higher-dimensional automaton respecting the precubical identities and is acyclic. The non-degenerate property of higher-dimensional automata is defined by the underlying precubical set. Hence, the property of non-degeneracies is preserved. As shown by Johansen in [28, Lemma 3.37], the way the events are picked in the definition of the mapping H do not matter.

**Example 4.2** (Strong asymmetric conflict)**.** *This example is taken from [28, Example 3.38], but can also be found in [17, Example 3], and in [43, page.22] the asymmetric conflict is considered* strong*.*

*The example has no concurrency and involves two events, imposing the only restriction that once event a happens, then event b cannot occur anymore.*

*Higher-dimensional automata are not good at identifying the particular events, making the representation of the example as a higher-dimensional automaton challenging. Higher-dimensional automata are state-based models that represent events by actions, where actions are labels for the particular events. These actions may occur multiple times, where*

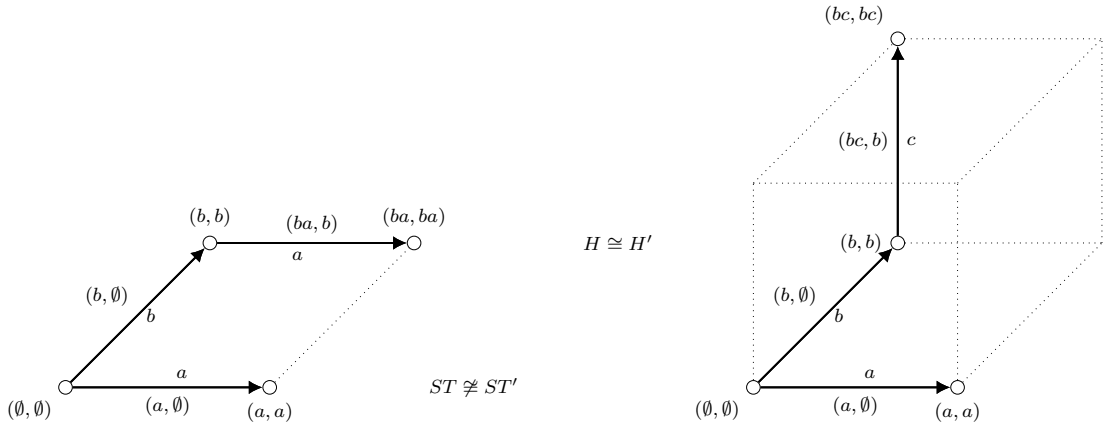



**Figure 4.1** Example of ST-structures and higher-dimensional automata representing the asymmetric conflict $a + b$. The higher-dimensional automaton (left) is isomorphic to the higher-dimensional automaton (right). On the other hand, we see that the ST-structure (left) is not isomorphic to the ST-structure (right). Since, the ST-structure (left) is on two events and the ST-structure (right) is on three events.

*events only occur once in an execution. The standard way of identifying events in higher-dimensional is by equivalence classes of transitions, where two transitions are equivalent when they are parallel in the boundary of a filled square, see Figure 3.5.*

*The equivalence classes of transitions for higher-dimensional automata, shown in Figure 4.1 (left), would not result in the corresponding 2-event ST-structure, which is what we want. However, would result in the 3-event ST-structure of Figure 4.1 (right). The 2-event ST-structure is not isomorphic to the 3-event ST-structure. On the other hand, the two representations of the higher-dimensional automata from Figure 4.1 (left and right) are isomorphic.*

In [28, Proposition 3.39], the mapping $\mathsf{H}$ from Definition 4.1 preserves isomorphism. However, it may collapse non-isomorphic ST-structures into isomorphic higher-dimensional automata. Consider the two ST-structures from Figure 4.1 which are not isomorphic since the left one is defined by two events while the right one is defined by three events, whereas the higher-dimensional automata that the mapping $\mathsf{H}$ associates are isomorphic. This means that $\mathsf{H}$ is not an embedding from $\mathbb{ST}$ to $\mathbb{HDA}$ since it looses information, namely, the events. A natural question to raise is if a mapping from $\mathbb{HDA}$ to $\mathbb{ST}$ is then an embedding from $\mathbb{HDA}$ to $\mathbb{ST}$. We will first define a mapping from $\mathbb{HDA}$ to $\mathbb{ST}$.

**Definition 4.3** (Events equivalence relation of $\mathsf{HDA}$ [28])**.** *Define a relation* $\overset{\mathsf{ev}}{\sim} \subseteq \mathcal{Q}_1 \times \mathcal{Q}_1$ *on transitions of* $\mathcal{H}$ *as*

$$q_1 \overset{\mathsf{ev}}{\sim} q_1' \quad \textit{iff} \quad \exists q_2 \in \mathcal{Q}_2 : \alpha_i(q_2) = q_1 \wedge \beta_i(q_2) = q_1'$$

*for some* $i \leq 2$ *and* $\alpha, \beta \in \{s, t\}$*. Consider the reflexive and transitive closure of the above relation, and denote it the same. This is now an equivalence relation on* $\mathcal{Q}_1$*.*

**Definition 4.4** ($\mathbb{HDA}$ to $\mathbb{ST}$ [28])**.** *Define a map* $\mathsf{ST} : \mathbb{HDA} \to \mathbb{ST}$ *which builds an ST-structure* $\mathsf{ST}(\mathcal{H})$ *by associating to each rooted path* $\pi \in \mathcal{H}$ *an ST-configuration as follows.*

1. *for the minimal rooted path which ends in I associate* $(\emptyset, \emptyset)$*;*
2. *for any path* $\pi$ *which ends in a transition* $en(\pi) = q_1 \in \mathcal{Q}_1$ *then*
   (a) *add the ST-configuration* $\mathsf{ST}_\pi(\pi) = \mathsf{ST}_\pi(\pi_s) \cup ([q_1], \emptyset)$*, where* $\pi_s$ *is a shorter path reaching through an s-map the homotopy class of* $\pi$*, that is,* $\pi_s \xrightarrow{s} q_1 \in \overleftrightarrow{[\pi]}$*;*
   (b) *add the ST-configuration* $\mathsf{ST}_\pi(\pi \xrightarrow{t} q_0) = \mathsf{ST}_\pi(\pi) \cup (\emptyset, [q_1])$*;*
3. *for any path* $\pi$ *which ends in a higher cell* $en(\pi) = q_n \in \mathcal{Q}_n$*, with* $n \geq 2$*, then add the ST-configuration* $\mathsf{ST}_\pi(\pi) = \mathsf{ST}_\pi(\pi^i) \cup \mathsf{ST}_\pi(\pi^j)$*, with* $\pi^i \neq \pi^j$*,* $\pi^i \xrightarrow{s} q_n \in \overleftrightarrow{[\pi]}$*, and* $\pi^j \xrightarrow{s} q_n \in \overleftrightarrow{[\pi]}$*.*

As shown in [28, Proposition 3.42], the mapping ST associates an acyclic and non-degenerate higher-dimensional automata with $\mathsf{ST}(H)$, which is a rooted, connected, and adjacent-closed ST-structure. Furthermore, in [28, Proposition 3.44], the mapping ST from Definition 4.4 preserves isomorphism of reachable parts. However, it may collapse non-isomorphic higher-dimensional automata into isomorphic ST-structures. This can be seen by considering the two higher-dimensional automata from Figure 4.2, which are translated into the same ST-structure by ST. The mapping ST considers the two higher-dimensional automata which are not isomorphic to be two ST-structures which are isomorphic. The higher-dimensional automata are not isomorphic since the right one has a split corner whereas the left one is a square with its faces nicely placed, forming a complete square.

The mapping ST is not an embedding from $\mathbb{HDA}$ to $\mathbb{ST}$ since it looses information, which in the example is the ability to announce the difference between when $a$ happens before $b$, or $b$ happens before $a$. This can also be seen, geometrically, as not being able to see the difference between the split corner of a square and a nicely shaped square.

In Figure 4.1, we can see that higher-dimensional automata are not able to precisely identify events, but can be faithfully interpreted as ST-structures. ST-structures are able to precisely capture events while events in higher-dimensional automata are not easily captured. As we have shown in Figure 4.2, ST-structures are not able to capture the difference between a higher-dimensional automaton and the unfolding of that higher-dimensional automaton. Hence, some processes might only be able to be represent as higher-dimensional automata, and identifying precisely these events of the processes would be challenging. We are interested in finding a model that is able to capture the aspect of higher-dimensional automata, but also be faithfully translated to ST-structures. In this regard, we will investigate Chu spaces and ST-structures.

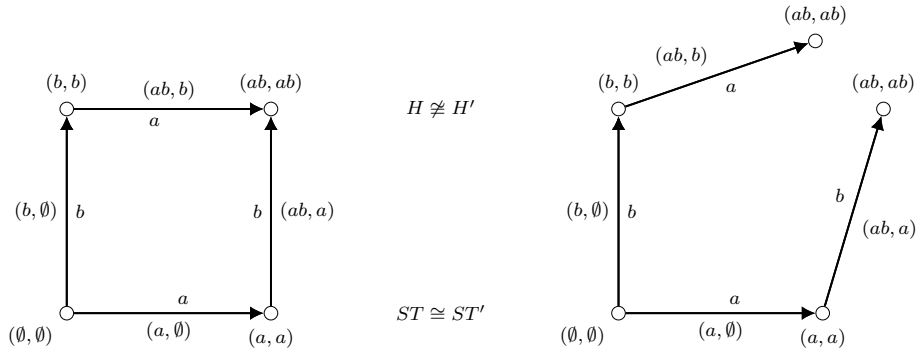



**Figure 4.2** Example of ST-structures and higher-dimensional automata representing the unfoldings of a square HDA through $\mathsf{ST}_\pi$. The ST-structure (left) is isomorphic to ST-structure (right). On the other hand, we see the higher-dimensional automaton (right) is not isomorphic to the higher-dimensional automaton (left). Since, the higher-dimensional automaton (right) has a split corner and the higher-dimensional automaton (left) is a nicely shaped square.

## 4.2 Chu spaces and ST-structures

Chu spaces can be viewed in various equivalent ways, as mentioned in Section 3.4. A Chu space can be viewed as a matrix with entries from $K$, where rows represent the events from $E$ and columns represent the configurations in $X$.

ST-structures capture the "*during*" aspect in the event-based setting, extending configuration structures with this notion. We consider Chu spaces $K = \mathbf{3} = \{0, \frac{1}{2}, 1\}$ with the order $0 < \frac{1}{2} < 1$, introducing the value $\frac{1}{2}$ to stand for *during*, or *in transition*. We see that Chu space over **3** is similar to the sculpting way of looking at Chu spaces as mentioned at the end of Section 3.4. Hence, revealing the intrinsic symmetry of events and states. Chu space over **3** is also called a triadic chu space.

**Definition 4.5** (Translations between ST and Chu)**.** *For an ST-structure* $\mathsf{ST}$ *over* $E$ *construct* $(E, X)^{\mathsf{ST}}$ *the associated Chu space over* **3** *with* $E$ *the set of events from* $\mathsf{ST}$*, and* $X \subseteq \mathbf{3}^E$ *containing for each ST-configuration* $(\mathcal{S}, \mathcal{T})$ *the state* $x^{(\mathcal{S},\mathcal{T})} \in X$ *formed by assigning to each* $e \in E$*:*

- $e \to 0$ *if* $e \notin \mathcal{S}$ *and* $e \notin \mathcal{T}$*;*
- $e \to \frac{1}{2}$ *if* $e \in \mathcal{S}$ *and* $e \notin \mathcal{T}$*;*
- $e \to 1$ *if* $e \in \mathcal{S}$ *and* $e \in \mathcal{T}$*.*

*The possibility that* $e \notin \mathcal{S}$ *and* $e \in \mathcal{T}$ *is dismissed by the requirement* $\mathcal{T} \subseteq \mathcal{S}$ *of ST-configurations. Call this mapping* $\mathsf{ChuST}(\mathcal{S}, \mathcal{T})$ *when applied to a ST-configuration and* $\mathsf{ChuST}(\mathsf{ST})$ *when applied to an ST-structure.*

*The other way, translate a Chu space* $(E, X)$ *into a ST-structure over* $E$ *with one ST-configuration* $(\mathcal{S}, \mathcal{T})^x$ *for each state* $x \in X$ *using the inverse of the above mapping. We use*

$\mathsf{STchu}(x)$ *for the ST-configuration obtained from the event listing* $x$*. For example, for an event listing* $x = (1, \frac{1}{2}, \frac{1}{2}, 0)$ *make the ST-configuration* $\mathsf{STchu}(x) = (\{e_1, e_2, e_3\}, \{e_1\})$ *where the last event* $e_4$ *does not appear neither in the first nor the second set of the ST-configuration.*

**Proposition 4.6** ([28, Section 3.4])**.** *ST-structures are isomorphic to Chu spaces over* **3***, such that* $\mathsf{STchu}(\mathsf{ChuST}(\mathsf{ST})) \cong \mathsf{ST}$*.*

Thus, a ST-configuration can be seen as a listing/tuple with values from **3**; which exact listing of the events $E$ is irrelevant once fixed. Therefore, when we later use ST-configurations to *label* cells of an HDA, we can alternatively use the Chu spaces notation, interchangeably.

**Lemma 4.7.** *For any ST-structure* $\mathsf{ST}$ *the Chu space* $\mathsf{ChuST}(\mathsf{ST})$ *is* extenssional*, meaning that no two states are identical, that is,* $\forall x, x'.\exists e : x(e) \neq x'(e)$*.*

*Proof.* In short, since ST-structures work with sets, we have that in the set of ST-configurations there are no two ST-configurations that are the same, then the states produced by ChuST would also be different by the virtue of the assignment from Definition 4.5 which associates unique events valuation to a ST-configuration.

In detail, for any $x^{(\mathcal{S},\mathcal{T})} \neq x^{(\mathcal{S}',\mathcal{T}')}$ they are created from some different $(\mathcal{S},\mathcal{T}) \neq (\mathcal{S}',\mathcal{T}')$, which implies one of the two cases:

1. When $\mathcal{S} \neq \mathcal{S}'$ then pick some $e \in \mathcal{S}$ such that $e \notin \mathcal{S}'$ (or the other way around if needed) then the states generated by ChuST would have the valuations: $x^{(\mathcal{S},\mathcal{T})}(e) \in \{\frac{1}{2}, 1\}$ and $x^{(\mathcal{S}',\mathcal{T}')}(e) = 0$, thus making them different.
2. When $\mathcal{S} = \mathcal{S}'$ but $\mathcal{T} \neq \mathcal{T}'$ the pick some $e \in \mathcal{T}$ such that $e \notin \mathcal{T}'$ (or the other way around if needed) then the states would have $x^{(\mathcal{S},\mathcal{T})}(e) = 1$ (because $e \in \mathcal{T} \subseteq \mathcal{S}$) and $x^{(\mathcal{S}',\mathcal{T}')}(e) = \frac{1}{2}$, thus making them different.

□

In Section 3.4, a morphism between Chu spaces $(E, X)$ and $(A, Y)$ is defined to be a pair $(f, g)$ of maps $f : E \to A$ and $g : Y \to X$ that satisfy the adjointness condition. We can extend ChuST to a functor between the two categories $\mathbb{ST}$ and $\mathbb{CHU}$ by defining its application to morphisms as $\mathsf{ChuST}(f) = (f, g)$ with $g$ defined in terms of $f$ using the equation 3.1.

**Lemma 4.8.** *The* ChuST *is a functor between* $\mathbb{ST}$ *and* $\mathbb{CHU}$*.*

*Proof.* Since the Chu-spaces generated by ChuST are extensional, then we define $g$ by defining what the values of all the events are from $A$ in that state from $X$

as $g(y)(e) = y(f(e))$. This now makes the two mapings $(f, g)$ to respect the adjointness condition of the category $\mathbb{CHU}$, that is, to be a proper morphism in this category. □

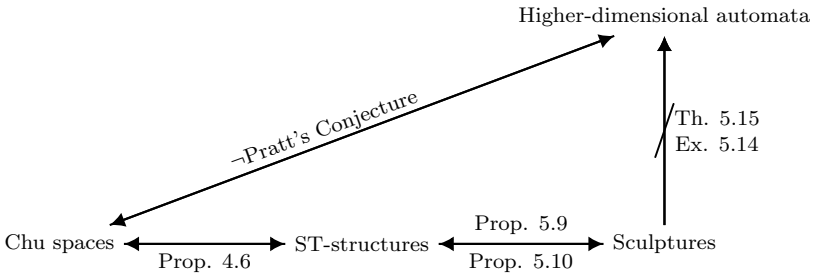
Note that STchu cannot be a functor because the Chu-transforms are too weak to allow us to prove the configuration-preserving property of the ST-morphisms.

## 4.3 Summary

From the open problem of [28], we have that there is no embedding between higher-dimensional automata and ST-structures. By introducing Chu spaces, we were able to define a translation between Chu spaces over **3** and ST-structures, and the translation can be used interchangeably. Also, Chu spaces over **3** are shown to be isomorphic to ST-structures.

At the end of Section 3.4, Chu spaces provided the notion of a sculpting method for higher-dimensional automata. A conjecture posed by Vaughan Pratt was that any higher-dimensional automata could be obtained using the sculpting method. As shown, we have that Chu spaces over **3** are isomorphic to ST-structures as well as there is no existing embedding between higher-dimensional automata and ST-structures. Hence, as the main result of our study, we show that not all higher-dimensional automata can be sculpted. We provide a diagram of the correlation between Chu spaces, ST-structures, Sculptures and higher-dimensional automata. In the diagram, we provide the propositions that prove the isomorphism between ST-structures and Chu spaces as well as ST-structures and Sculptures. Also, we provide the counter-example, see Example 5.14, which shows a higher-dimensional automaton which is not a sculpture.

From the result of Chu spaces over **3** being isomorphic to ST-structures, we will challenge Pratt's conjecture by developing an algorithm to decide whether an higher-dimensional automaton can be sculpted or not, and show several simple examples of *acyclic* higher-dimensional automata which are not sculptures. We believe that this contradicts Pratt's conjecture. Below is a diagram of the propositions which show the correlation between Chu spaces, ST-structures and Sculptures. Together with a counter-example of the conjecture, showing a higher-dimensional automaton which is not a sculpture.




We have shown a proof for Proposition 4.6, but need to provide proofs for Proposition 5.9 and Proposition 5.10 to show the correlation between Chu spaces, sculptures and ST-structures. In the next chapter, we will provide the remaining proofs needed to show both the counter-example of the conjecture and correlation of the models.

# Chapter 5

# Sculpting in concurrency

In this chapter we define a method of modelling concurrent behaviour of higher-dimensional automata. We call this method *sculpting*. In other words, the process of modelling a concurrent system using higher-dimensional automata may be considered as a sculpting process − take one single higher-dimensional cube, having enough concurrency, meaning enough events, and remove cells until the desired concurrent behaviour is obtained. The model obtained is called a *sculpture*.

We investigate the method of sculpting which has not been studied for higher-dimensional automata before. One goal is to tighten the correlation between ST-structures and higher-dimensional automata, which was left open in [28] where neither model could be embedded into the other. The main result of our study is the fact that the sculpting method cannot build all higher-dimensional automata, but only a strict subset of these. We identify the category of sculptures and its relation to the category of higher-dimensional automata and the category of ST-structures.

## 5.1 Sculptures

The intuition for the method of programming as sculptures was first introduced by Pratt in [40] as a method of identifying events in higher dimensional automata. It was considered in regards to the Chu space approach to higher dimensional automata, where one instead starts with a single cube of very large, possibly infinite, dimension and "*sculpts*" the desired process by removing unwanted faces. Initially, a cube constitutes the events to be a discrete and unstructured set, forming a complete cube. By removing states, the event set becomes structured. For example, sculpting two events by removing all the states distinguishing them renders them equivalent. The removal of states from higher-dimensional automata may be understood as furnishing higher-dimensional automata with a certain structure.

A sculpture consists of a higher-dimensional automaton, a bulk and an embedding. A *bulk* is a *non-selflinked* precubical set which is generated by a single cube of very large, possibly infinite, dimension. A precubical set $\mathcal{X}$ is *non-selflinked* if it holds

for all $x, y \in \mathcal{X}$ that there exists at most one index sequence $i_1 \leq \ldots \leq i_n$ such that $x = \alpha^1_{i_1} \cdots \alpha^n_{i_n} y$ for $\alpha^1, \ldots, \alpha^n \in \{s, t\}$. Hence, there is unique representation for each cell in the precubical set. An embedding is an injective morphism such that there is a mapping from the precubical set to the bulk. Intuitively, an embedding is considered to be the same as sculpting the desired process by removing unwanted faces.

**Definition 5.1** (Bulk)**.** *A precubical set $\mathcal{X}$ is a* bulk *if it is non-selflinked and generated by precisely one n-cube, that is, if $|\mathcal{X}_{\dim X}| = 1$.*

Any two $d$-dimensional bulks are isomorphic, where the isomorphism is generated by a permutation of the $d$ directions of the generating cubes. Hence we may talk of *the* $d$-dimensional bulk, and denote it by $B^d$.

We develop a naming scheme for bulks inspired by Chu spaces.

**Definition 5.2** (Canonical naming)**.** *Fix $d \in \mathbb{N}$ and let $B^d = \{0, \frac{1}{2}, 1\}^d$. For $n = 0, \ldots, d$, let $B_n = \{(x_1, \ldots, x_d) \in B \mid |\{i \mid x_i = \frac{1}{2}\}| = n\}$ be the set of tuples with precisely $n$ occurrences of $\frac{1}{2}$. For $n = 1, \ldots, d$, define face maps $s_k, t_k : B_n \to B_{n-1}$ $(k = 1, \ldots, n)$ as follows: for $x = (x_1, \ldots, x_d) \in B_n$ with $x_{i_1} = \cdots = x_{i_n} = \frac{1}{2}$, let $s_k x = (x_1, \ldots, 0_{i_k}, \ldots, x_d)$ and $t_k x = (x_1, \ldots, 1_{i_k}, \ldots, x_d)$ be the tuples with the $k$-th occurrence of $\frac{1}{2}$ set to 0 or 1, respectively. We call this the* canonical naming *for the bulk $B^d$.*

The above construction essentially labels the $d$-bulk with lists of Chu-labels, namely, with states from the Chu space on $d$ events. Any cell $q^b$ in a bulk $B^d$ is reached from the highest cell $q_d$ through a sequence of face maps applications $\alpha_{i_1} \cdots \alpha_{i_k} q_d = q^b$ of correct indices.

**Lemma 5.3.** *The structure $B^d$ as defined above is the $d$-dimensional bulk.*

*Proof.* It is trivial to check that $B$ is a precubical set. $B^d$ is also non-selflinked, and $|B_d| = |\{(\frac{1}{2}, \ldots, \frac{1}{2})\}| = 1$, thus by uniqueness up to isomorphism, $B^d$ is the $d$-dimensional bulk. □

The *initial state* $i_{B^d}$ of the bulk $B^d$ is the cell named $(0, \ldots, 0)$ in the canonical naming. (any automorphism of $B^d$ fixes this cell.) This turns bulks into HDA.

The $d$-dimensional bulk can be embedded into any bulk of dimension $d' \geq d$ by using the embedding $em^{d'}_d : B^d \hookrightarrow B^{d'}$ which maps any cell $(t_1, \ldots, t_d)$ to $(t_1, \ldots, t_d, 0, \ldots, 0)$ in the canonical naming. It can easily be shown that up to isomorphism, $em^{d'}_d$ is the *only* HDA morphism from $B^d$ to $B^{d'}$, hence also that there are no HDA morphisms $B^d \to B^{d'}$ for $d' < d$.

**Definition 5.4** (Sculpture). *A* sculpture *is a* HDA $\mathcal{X}$ *together with a bulk* $B^d$ *and a* HDA *embedding morphism, in other words, an injective morphism,* $em : X \hookrightarrow B^d$*. We denote a sculpture by* $(\mathcal{X}, B^d, em)$*.*

morphism *of sculptures* $em : \mathcal{X} \hookrightarrow B^d$, $em' : \mathcal{X}' \hookrightarrow B^{d'}$ *is a pair of* HDA *morphisms* $f : \mathcal{X} \to \mathcal{X}'$, $b : B^d \to B^{d'}$ *such that the square*

$$\begin{array}{ccc} \mathcal{X} & \xrightarrow[f]{} & \mathcal{X}' \\ {\scriptstyle em}\downarrow & & \downarrow{\scriptstyle em'} \\ B^d & \xrightarrow{b} & B^{d'} \end{array}$$

*commutes, namely,* $b \circ em = em' \circ f$*. By the above considerations, this entails that* $d' \geq d$ *and* $b$ *is injective, hence also* $f$ *must be injective. To sum up, morphisms of sculptures are commuting squares of embeddings. Two sculptures are isomorphic, denoted* $\cong$*, when the morphism between them has both* $f, b$ *bijections; thus making the bulks of the same dimension.*

*Remark* 5.5. One precubical set can be seen as two different sculptures, for example, from two different dimensional bulks, in both cases being a simplistic sculpture, meaning it all depends on the embedding morphism. Then this precubical set enters as source object of several sculpture morphisms, as seen in Figure 4.1.

The major problem with both translations is related to the fact that it is not clear how to identify the events in a higher-dimensional automaton. The best example for this is the fact that the mapping ST destroys the interleaving square, which is due to the fact that the standard method of identifying events in a higher-dimensional automaton by equivalent transitions opposite in a filled square fails for this unfilled square; see Figure 5.1 (left). This same issue about events is also the one that causes the problem for the other mapping H where we could not say in the higher-dimensional automaton that was generated whether this was representing two or three events.

However, we show how sculpting allows to identify the events in a higher-dimensional automaton in the same way as ST-structures work with events (we still use the standard intuitive method of seeing events as equivalence classes of transitions opposite in a filled square).

We show that the ST-structures captures precisely the higher-dimensional automaton that can be seen as sculptures.

## 5.2 Sculptures and ST-structures

We show that sculptures and ST-structures are isomorphic while also respecting the computation steps. This result resolves the open problems noticed in [28, Section 3.3] that there is no adjoint between the two translations given there between ST-structures and higher-dimensional automata. We are using here the same translations, but add more information that is given by the bulk and the embedding of the sculpture, which allows us to obtain a one-to-one mapping between ST-structures and this class of higher-dimensional automata that sculptures form.

Through the observation from Section 4.2, the results in this Section extend to Chu spaces over **3** as well. Thus, Chu spaces are not enough to capture the concurrency that general higher-dimensional automata can express.

From Section 4.1, we considered the mapping of $\mathbb{ST}$ into $\mathbb{HDA}$ which was taken from [28]. We showed that for a rooted, connected, and adjacent-closed ST-structure ST the mapping associates a higher dimensional automaton respecting all precubical identities and is acyclic and non-degenerate. Also, we showed that the order in which the events were picked did not matter.

This mapping can be extended to generate a sculpture, in other words, the dimension of the sculpture and the embedding, thus retaining the information about the events from the ST-structure it encodes.

**Definition 5.6** ($\mathbb{ST}$ to sculptures)**.** *We define a mapping* $\mathsf{H}^{\mathsf{s}}$ *that for any ST-structure generates an* HDA*, as well as a bulk and an embedding, thus a sculpture, as follows. Consider an* $\mathsf{ST} = (E, ST)$*, with the events linearly ordered as a list* $\overrightarrow{E}$*. Then* $\mathsf{H}^{\mathsf{s}}(\mathsf{ST})$ *returns the* HDA *which*

- *has cells* $\mathcal{Q} = \{q^{(\mathcal{S},\mathcal{T})} \in \mathcal{Q}_n \mid (\mathcal{S},\mathcal{T}) \in ST \text{ and } |(\mathcal{S} \setminus \mathcal{T})| = n\}$*;*
- *for any two cells* $q^{(\mathcal{S},\mathcal{T})}$ *and* $q^{(\mathcal{S}\setminus e,\mathcal{T})}$ *add the map entry* $s_i(q^{(\mathcal{S},\mathcal{T})}) = q^{(\mathcal{S}\setminus e,\mathcal{T})}$ *where* $i$ *is the index of the event* $e$ *in the listing* $\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$*;*
- *for any two cells* $q^{(\mathcal{S},\mathcal{T})}$ *and* $q^{(\mathcal{S},\mathcal{T}\cup e)}$ *add the map entry* $t_i(q^{(\mathcal{S},\mathcal{T})}) = q^{(\mathcal{S},\mathcal{T}\cup e)}$ *where* $i$ *is the index of the event* $e$ *in the listing* $\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$*.*

$\overrightarrow{E}{\downarrow_{(\mathcal{S}\setminus\mathcal{T})}}$ *is the listing* $\overrightarrow{E}$ *restricted to the set* $\mathcal{S} \setminus \mathcal{T}$*.*

*Build the bulk* $B^n$*, with* $n = |(E)|$*, as in Lemma 5.3 using the canonical naming on the same listing of the events* $\overrightarrow{E}$ *as used to generate the above* HDA*. The embedding* $em : \mathsf{H}^{\mathsf{s}}(\mathsf{ST}) \hookrightarrow B^n$ *is defined as* $em(q^{(\mathcal{S},\mathcal{T})}) = \mathsf{ChuST}(\mathcal{S},\mathcal{T})$ *returning the Chu-labelling as in Proposition 4.6 on the same listing of events* $\overrightarrow{E}$*.*

The mapping $\mathsf{H}^{\mathsf{s}}$ translates a rooted, connected, and adjacent-closed ST-structure ST into a higher dimensional automaton, respecting all cubical laws. Moreover, it

is immaterial which listing of the events is picked in the definition (these results are direct adaptations of results from [28]). This mapping preserves isomorphism of ST-structures, and moreover, it does not collapse non-isomorphic ST-structures.

**Proposition 5.7.** *For ST-structures* $\mathsf{ST}$ *and* $\mathsf{ST}'$, $\mathsf{ST} \cong \mathsf{ST}'$ *iff* $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}) \cong \mathsf{H}^{\mathsf{s}}(\mathsf{ST}')$

*Proof.* It is clear that $\mathsf{ST} \cong \mathsf{ST}'$ implies $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}) \cong \mathsf{H}^{\mathsf{s}}(\mathsf{ST}')$. We now show that $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}) \cong \mathsf{H}^{\mathsf{s}}(\mathsf{ST}')$ implies $\mathsf{ST} \cong \mathsf{ST}'$. We show the existence of an isomorphism over the ST-structures, knowing the isomorphism over their translations as sculptures, making use of a fixed listing of events that the map $\mathsf{H}^{\mathsf{s}}$ works with, the same listing for both translations, since we know the translation can choose any listing.

Since the two sculptures $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}) \cong \mathsf{H}^{\mathsf{s}}(\mathsf{ST}')$ are isomorphic it means that the HDAs are embedded in the same bulk, which means we are working with the same set of events $E = E'$ generated by $\mathsf{H}^{\mathsf{s}}$, modulo a reordering of their listing, which is captured by the second map $b$ between the bulks. We know that $b$ is a bijection on the cells of the bulks, and thus, through the canonical naming scheme, $b$ induces a bijection on $E$ which we call $g$ defined by $g(e_i) = e'_i$ where the index $i$ is important (i.e., $g$ associates the event on position $i$ in the listing $\overrightarrow{E}$ with the event on the same position in the listing $\overrightarrow{E'}$). We use $g$ as the isomorphism that we are looking for between the two ST-structures.

We need to show that $g$ preserves configurations, i.e., for some $(\mathcal{S}, \mathcal{T}) \in \mathsf{ST}$ we prove that $g(\mathcal{S}, \mathcal{T}) \in \mathsf{ST}'$. We make use of the commuting square property of the sculpture morphisms. For the ST-configuration there exists the corresponding cell $q^{(\mathcal{S},\mathcal{T})}$ generated by the $\mathsf{H}^{\mathsf{s}}$, which in turn is embedded in the bulk as $\mathsf{ChuST}(\mathcal{S}, \mathcal{T})$. At the same time the HDA morphism associates $f(q^{(\mathcal{S},\mathcal{T})}) = q^{(\mathcal{S}',\mathcal{T}')}$ to some cell in $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}')$ which in turn comes from some ST-configuration $(\mathcal{S}', \mathcal{T}')$. This cell is embedded into the bulk as $\mathsf{ChuST}(\mathcal{S}', \mathcal{T}')$ which is equated to $\mathsf{ChuST}(\mathcal{S}, \mathcal{T}) = b(\mathsf{ChuST}(\mathcal{S}, \mathcal{T}))$ by the bulk isomorphism, meaning that respective events in the two listings receive the same values. Now, going through the translation between Chu states and ST-configurations we can see how $g(\mathcal{S}, \mathcal{T}) = (g(\mathcal{S}), g(\mathcal{T})) = (\mathcal{S}', \mathcal{T}') \in \mathsf{ST}'$.

- If $e_i \notin \mathcal{S}$ then $\mathsf{ChuST}(\mathcal{S}, \mathcal{T})(e_i) = 0$, which means that also $g(e_i) = e'_i$ takes the same value in $\mathsf{ChuST}(\mathcal{S}', \mathcal{T}')(g(e_i)) = 0$. This means that $g(e_i) \notin \mathcal{S}' = g(\mathcal{S})$ because of injectivity.
- If $e_i \in \mathcal{S}$ and $e_i \notin \mathcal{T}$ then $\mathsf{ChuST}(\mathcal{S}, \mathcal{T})(e_i) = \frac{1}{2}$, which means that also $g(e_i) = e'_i$ takes the same value in $\mathsf{ChuST}(\mathcal{S}', \mathcal{T}')(g(e_i)) = \frac{1}{2}$. This means that $g(e_i) \in g(\mathcal{S})$ and $g(e_i) \notin g(\mathcal{T})$ because of injectivity on $\mathcal{S}$.
- If $e_i \in \mathcal{T} \subseteq \mathcal{S}$ then $\mathsf{ChuST}(\mathcal{S}, \mathcal{T})(e_i) = 1$ which means that also $g(e_i) = e'_i$ takes the same value in $\mathsf{ChuST}(\mathcal{S}', \mathcal{T}')(g(e_i)) = 1$. This means that $g(e_i) \in g(\mathcal{T}) \subseteq g(\mathcal{S})$.

This proves that $(g(\mathcal{S}), g(\mathcal{T})) \in \mathsf{ST}'$.

□

**Definition 5.8** (Sculptures to $\mathbb{ST}$)**.** *Define a mapping* $\mathsf{ST_s}$*, which for a sculpture* $\mathcal{H}^n = (\mathcal{H}, B^n, em)$ *associates the ST-structure* $\mathsf{ST_s}(\mathcal{H}^n)$ *as follows. Take a linearly ordered set* $\overrightarrow{E}$ *(of events) of cardinality as the dimension of the bulk* $n = \dim B^n$*. The ST-configurations of* $\mathsf{ST_s}(\mathcal{H}^n)$ *are obtained from the cells of* $H$ *as* $\mathbb{S} = \{\mathsf{STchu}(em(q)) \mid q \in \mathcal{H}\}$.

Intuitively, since we have the bulk we can work with the canonical naming based on a fixed listing of events. Using Proposition 4.6 we can associate an ST-configuration to each cell of the HDA by going through the embedding to the corresponding cell in the bulk. It is clear that $\mathsf{ST_s}(\mathcal{H}^n)$ is rooted, connected and closed under single events, that is, regular.

The following results show a one-to-one correspondence between the ST-structures and sculptures.

**Proposition 5.9.** *For an arbitrary ST-structure* $\mathsf{ST}$ *we have*

$$\mathsf{ST_s}(\mathsf{H^s}(\mathsf{ST})) \cong \mathsf{ST}.$$

*Proof.* To prove the isomorphism between the right-hand $\mathsf{ST} = (E, \mathbb{S})$ and the left-hand $\mathsf{ST_s}(\mathsf{H^s}(\mathsf{ST}))$ we need to show that $\exists f : E \rightharpoonup E'$ such that $f$ preserves ST-configurations. We will consider $E'$ to be the events produced by the left-hand side of the isomorphism, precisely, by $\mathsf{ST_s}(\mathsf{H^s}(\mathsf{ST}))$.

The application of the mapping $\mathsf{H^s}$ generates an HDA, as well as a bulk and an embedding, considering the events of $\mathsf{ST}$ to be linearly ordered as a list $\overrightarrow{E}$. $\mathsf{H^s}$ builds cells $\mathcal{Q} = \{q^{(\mathcal{S},\mathcal{T})} \in \mathcal{Q}_n \mid (\mathcal{S},\mathcal{T}) \in \mathbb{S} \text{ and } |(\mathcal{S} \setminus \mathcal{T})| = n\}$ and the embedding $em(q^{(\mathcal{S},\mathcal{T})}) = \mathsf{ChuST}(\mathcal{S},\mathcal{T})$ into the bulk $B^{|E|}$; thus the sculpture $\mathcal{H}^n = (\mathcal{Q}, B^{|E|}, em(q^{(\mathcal{S},\mathcal{T})}))$. The embedding $em$ returns for each cell $q^{(\mathcal{S},\mathcal{T})}$ the Chu-labelling as in Proposition 4.6 on the same listing of events $\overrightarrow{E}$.

Thus, the ST-structure produced on the left-hand side by $\mathsf{ST_s}$ applied to the above sculpture $\mathcal{H}^s(\mathsf{ST}) = \mathcal{H}^n$ assumes, without loss of generality, to have the same listing of events $\overrightarrow{E}$ as before. The ST-configurations of $\mathsf{ST_s}(\mathcal{H}^n)$ are obtained from the cells of $\mathcal{H}^n$ as $\mathbb{S}' = \{\mathsf{STchu}(em(q^{(\mathcal{S},\mathcal{T})})) \mid q^{(\mathcal{S},\mathcal{T})} \in \mathcal{H}^n\}$, see Definition 5.8.

Since $E'$ is the same set of events $E$, we take the isomorphism map $f : E \to E$ to be identity function on $E$. We check that $f$ preserves the ST-configurations, such that $f(\mathcal{S},\mathcal{T}) := (f(\mathcal{S}), f(\mathcal{T})) = (\mathcal{S},\mathcal{T}) \in \mathbb{S}'$. From this it must be that $\exists q^{(\mathcal{S},\mathcal{T})} \in \mathcal{H}^n$ : $\mathsf{STchu}(em(q^{(\mathcal{S},\mathcal{T})}) = (\mathcal{S},\mathcal{T})$ by Definition 5.8. Further, from Definition 5.6 we know that $\mathsf{STchu}(em(q^{(\mathcal{S},\mathcal{T})}) = \mathsf{STchu}(\mathsf{ChuST}(\mathcal{S},\mathcal{T}))$. Finally, by Proposition 4.6 we have the expected result $\mathsf{STchu}(em(q^{(\mathcal{S},\mathcal{T})}) = \mathsf{STchu}(\mathsf{ChuST}(\mathcal{S},\mathcal{T})) = (\mathcal{S},\mathcal{T})$.

It is easy to see that $f$, as identity function, is locally injective and total. □

**Proposition 5.10.** *For any sculpture $\mathcal{H}^n = (\mathcal{H}, B^n, em)$ we have*

$$\mathsf{H}^{\mathsf{s}}(\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n)) \cong \mathcal{H}^n.$$

*Proof.* To prove the isomorphism between the right-hand $\mathcal{H}^n = (\mathcal{H}, B^n, em)$ and the left-hand $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n))$ we need to show that $\exists f : \mathcal{H} \to \mathcal{H}'$ and $\exists b : B^n \to B^n$ such that $f, b$ are bijective and the square commutes, namely, $b \circ em = em' \circ f$.

$$\begin{array}{ccc} \mathcal{H} & \xrightarrow{f} & \mathcal{H}' \\ {\scriptstyle em}\downarrow & & \downarrow{\scriptstyle em'} \\ B^n & \xrightarrow{b} & B^{n'} \end{array}$$

First, we consider the mapping $\mathsf{ST}_{\mathsf{s}}$ provided in Definition 5.8 which generates an *ST* for a sculpture, as follows. Take a linearly ordered set $\overrightarrow{E}$ (of events) of cardinality as the dimension of the bulk, that is, $|E| = n = \dim B^n$. The ST-configurations of $\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n)$ are obtained from the cells of $\mathcal{H}$, such that $\forall q \in \mathcal{H} : \mathsf{ST}_{\mathsf{s}}(q) = STChu(em(q)) = (\mathcal{S}, \mathcal{T})^q$. This is the ST-structure produced on the left-hand side by $\mathsf{ST}_{\mathsf{s}}$ applied to the sculpture $\mathcal{H}^n$.

We will now consider the mapping $\mathsf{H}^{\mathsf{s}}$ provided in Definition 5.6 which generates a HDA for any ST-structure, as well as a bulk and an embedding, thus a sculpture, as follows. $\mathsf{H}^{\mathsf{s}}$ requires the events of the ST-structure to be linearly ordered. We take the same order produced above by $\mathsf{ST}_{\mathsf{s}}$.

1. Build the bulk $B^{n'}$, with $n' = |E| = n$, as in Lemma 5.3 using the canonical naming on the listing of the events $\overrightarrow{E}$. This is thus the same bulk $B^n$ from $\mathcal{H}^n$. Thus we can take the $b$ part of the sculptures morphism to be the identity function which is thus a morphism between HDA. This is also a bijection.

2. The HDA $\mathsf{H}^{\mathsf{s}}(\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n))$ has cells $\mathcal{Q} = \{p^{(\mathcal{S},\mathcal{T})} \in \mathcal{Q}_n \mid (\mathcal{S}, \mathcal{T}) \in \mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n)$ and $|(\mathcal{S} \setminus \mathcal{T})| = n\}$. Since each ST-configuration corresponds to some $q \in \mathcal{H}^n$ we thus construct one cell $p^{(\mathcal{S},\mathcal{T})^q}$ for each cell $q$; which according to Definition 5.8 is built using the embedding, in other words, $p^{(\mathcal{S},\mathcal{T})^q} = p^{\mathsf{STchu}(em(q))}$.

3. The embedding $em' : \mathsf{H}^{\mathsf{s}}(\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n)) \hookrightarrow B^{n'}$ is defined as $em'(p^{(\mathcal{S},\mathcal{T})}) = Chu(\mathcal{S}, \mathcal{T})$ returning the Chu-labelling as in Proposition 4.6 on the same listing of events $\overrightarrow{E}$.

   We thus take $f : \mathcal{H} \to \mathsf{H}^{\mathsf{s}}(\mathsf{ST}_{\mathsf{s}}(\mathcal{H}^n))$ to be defined as $f(q) = p^{\mathsf{STchu}(em(q))}$ and $f^{-1}(p^{\mathsf{STchu}(em(q))}) = q$; thus obtaining a bijection between the cells of the respective HDAs.

   We show that $f$ is a morphism of HDAs. We thus show that $f$ commutes with the face maps, such that for any $q$ we show $s_i(f(q)) = f(s_i(q))$. We have $s_i(f(q)) = s_i(p^{\mathsf{STchu}(em(q))}) = s_i(p^{(\mathcal{S},\mathcal{T})})$ if we call $\mathsf{STchu}(em(q)) = (\mathcal{S}, \mathcal{T})$.

Definition 5.6 relates this $i$-th map to a cell made from the ST-configuration $(\mathcal{S} \setminus e, \mathcal{T})$ with $e$ being the $i$-th event in the listing $\overrightarrow{E}\restriction_{\mathcal{S}\setminus\mathcal{T}}$. Since the bulk uses the same listing, we thus have this ST-configuration obtained as $\mathsf{STchu}(em(s_i(q)))$. On the right-hand side of the equality we have the same by definition $f(s_i(q)) = p^{\mathsf{STchu}(em(s_i(q)))}$.

We show that the sculptures morphism square commutes. The fact that $f, b$ are bijections finishes the proof, which proves the isomorphism property. We show that $em(q) = em'(f(q))$ by working with the right-hand side: $em'(f(q)) = em'(p^{\mathsf{STchu}(em(q))}) = \mathsf{ChuST}(\mathsf{STchu}(em(q))) = em(q)$.

□

We can also understand $\mathsf{ST}_\mathsf{s}$ as labelling every cell of the sculpture with a ST-configuration, or equivalently (because of Proposition 4.6) with a Chu-3 state, which in the terminology of Lemma 5.3 we call this a Chu-listing or Chu-label. Corollary 5.11 says that this labelling is unique. Thus, in sculptures we have a one-to-one correspondence between HDA states, ST-configurations, and Chu-3 states.

**Corollary 5.11.** *In a bulk every cell has a unique label (either as a ST-configuration or as a Chu-label representation). Thus, there are no two cells of the bulk with the same label.*

*Proof.* This is easy to see from the proofs above. □

## 5.3 Sculptures and HDA

The above mappings do not provide us with a procedure for deciding whether an higher-dimensional automaton, even finite, is a sculpture, because the bulk can be of any dimension, hence there are infinitely many embeddings to check. There is an alternative way of translating higher-dimensional automata into ST-structures that work on paths starting from the shortest path in the initial cell, essentially unfolding the higher-dimensional automaton.

**Definition 5.12** ($\mathbb{HDA}$ to $\mathbb{ST}$ through paths)**.** *Define a map $\mathsf{ST}_\pi : \mathbb{HDA} \to \mathbb{ST}$ which builds a ST-structure $\mathsf{ST}_\pi(\mathcal{H})$ by associating to each rooted path $\pi \in \mathcal{H}$ an ST-configuration as follows.*

1. *for the minimal rooted path, which ends in $\mathcal{I}$, associate $(\emptyset, \emptyset)$;*
2. *for any path $\pi = \pi_s \xrightarrow{s} q_1$ that ends in a transition $en(\pi) = q_1 \in \mathcal{Q}_1$ then*
   (a) *add the ST-configuration $\mathsf{ST}_\pi(\pi) = \mathsf{ST}_\pi(\pi_s) \cup (q_1, \emptyset)$;*

*(b) add the ST-configuration* $\mathsf{ST}_\pi(\pi \xrightarrow{t} q_0) = \mathsf{ST}_\pi(\pi) \cup (\emptyset, q_1)$*;*

3. *for any path* $\pi$ *that ends in a higher cell* $en(\pi) = q_n \in \mathcal{Q}_n$*, with* $n \geq 2$*, then add the ST-configuration* $\mathsf{ST}_\pi(\pi) = \mathsf{ST}_\pi(\pi^i) \cup \mathsf{ST}_\pi(\pi^j)$*, with* $\pi^i \neq \pi^j$*,* $\pi^i \xrightarrow{s} q_n$*, and* $\pi^j \xrightarrow{s} q_n$*.*

Note that in the case 3 above the paths $\pi^i, \pi^j$ always exist because we work with non-selflinked HDAs. All cells are reachable through the paths considered in the above definition when applied inductively on the length of the distance from the initial cell.

For every transition we add one new event to the ST-structure. This adds too many events and does not capture the geometrical intuitions about concurrency, where transitions parallel in the sides of a filled square should represent the same event. Indeed, the construction is similar to an *unfolding* [4], except that no homotopy equivalence is applied. See [28, Def. 3.39] for a related construction.

If we are faced with an HDA which may or may not be a sculpture, then there is a minimal equivalence on its transition cells, which is exactly the equivalence of two cells appearing as opposite faces of a square, and the transitive closure. This equivalence has already been given in Definition 4.3. We will use the *events equivalence relation* to decide if a HDA may or may not be a sculpture. The construction from Definition 5.12 labels each *cell* with one or more ST-configurations as follows:

**Definition 5.13.** *For an HDA* $H$*, and using the notation of Definition 5.12, let* $\rho_0 \subseteq H \times \mathsf{ST}_\pi(H)$ *be the relation given by* $\rho_0 = \{(q, \mathsf{ST}_\pi(\pi)) \mid en(\pi) = q\}$*. For an equivalence relation* $\sim \subseteq Q_1 \times Q_1$*, let*

$$\rho_\sim = \{(q, (S, T)_{/\sim}) \mid (q, (S, T)) \in \rho_0\}.$$

We will apply the minimal equivalence, $\overset{\mathsf{ev}}{\sim}$, on the cell naming as $\rho_{\overset{\mathsf{ev}}{\sim}}$. We use $\rho_{\overset{\mathsf{ev}}{\sim}}$ to prove in Theorem 5.15 that the HDA from Example 5.14 cannot be sculpted.

**Example 5.14** (Broken box example)**.** *This example is taken from [13, Section 9.2.1], and also found illustrated in [28, Figure 6]. As part of van Glabbeek's presentation at EXPRESS 2004, the process displayed in Figure 5.1 (left) was demonstrated with the help of the audience. We name the Figure 5.1 (left) for the broken box since it represent a higher-dimensional automaton which looks almost like a complete cube, but with a single corner split.*

*The example was of two computer scientists A and B travelling from one end of the podium to the other. Their task was to perform the actions a and b, respectively, of crossing a line on the podium. Due to strategic placing of obstacles, the only place where this was possible was at a narrow opening between the obstacles that had room for only one of the scientist A and B at a time. This made the action a and b mutually exclusive, in the sense that they could not occur simultaneously.*

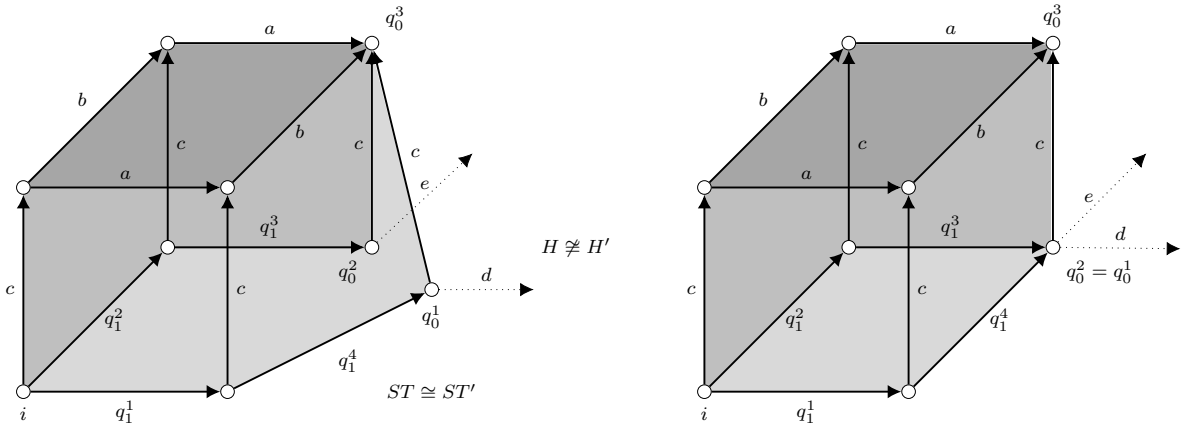



**Figure 5.1** Example of sculptures, with ST-configurations as labels, and higher-dimensional automata representing the broken box example. The higher-dimensional automata (left) is not a sculpture, while the higher-dimensional automata (right) is a sculpture. We also see the higher-dimensional automaton (left) is not isomorphic to the higher-dimensional automaton (right). Since, the higher-dimensional automaton (left) has a split corner and the higher-dimensional automaton (right) is a nicely shaped cube.

*A third computer scientist C, was assigned the task c of removing an obstacle that caused the bottleneck to exist. The action c was executed causally independent of a and b, that is, a and b are concurrent. The actions a and b were mutually exclusive only until c occurred, after which they became causally independent.*

*Finally, a fourth participant was assigned the task of making a statement when a and b had both occurred before the action c started. This statement was going to be d in case A passed the bottleneck before B did, and e in case B passed the bottleneck before A did. Hearing this statement would prevent computer scientist C from carrying out action c.*

*The higher-dimensional automata (left) is not a sculpture, while the higher-dimensional automata (right) is a sculpture. We also see the higher-dimensional automaton (left) is not isomorphic to the higher-dimensional automaton (right). Since, the higher-dimensional automaton (left) has a split corner and the higher-dimensional automaton (right) is a nicely shaped cube.*

**Theorem 5.15** (A non-sculpture)**.** *The* HDA *from Example 5.14 is not a sculpture.*

*Proof.* To show that the broken box cannot be sculpted we apply the labelling strategy described above. First we apply the unfolding procedure $\mathsf{ST}_\pi$ and for the two problematic corner states $q_0^1$ and $q_0^2$ we obtain the following ST-configurations $\mathsf{ST}_\pi(\pi_1) = (\{q_1^1, q_1^4\}, \{q_1^1, q_1^4\})$ respectively $\mathsf{ST}_\pi(\pi_2) = (\{q_1^2, q_1^3\}, \{q_1^2, q_1^3\})$, where $\pi_1$ is the lower rooted path ending in $q_0^1$ and $\pi_2$ is the other lower path ending in $q_0^2$.

The second step is to apply the minimal equivalence $\overset{\mathsf{ev}}{\sim}$, since this is required for any HDA. Indeed, if a HDA can be sculpted, then any cell of higher concurrency would be mapped to a unique cell in the bulk; in particular, each square from the

HDA is mapped to a square in the bulk. Since we identify events in the bulk as opposite sides of bulk squares, then in the HDA any transitions opposite in a square would need to also be considered equivalent (this is what the minimal equivalence is doing).

Now applying $\overset{\text{ev}}{\sim}$ on our example equates $q_1^1 \overset{\text{ev}}{\sim} q_1^3$ because of the three squares, front, top, back, which share the sides labelled with $a$ in the figure. (Transitivity of the equivalence was applied.) The same argument equates $q_1^2 \overset{\text{ev}}{\sim} q_1^4$, this time going through the squares left-side, top, right-side.

We now see that through $\rho_{\underset{\sim}{\text{ev}}}$ we have labelled both $q_0^1$ and $q_0^2$ with the same label $(\{[q_1^2], [q_1^3]\}, \{[q_1^2], [q_1^3]\})$, made of equivalence classes.

However, for a sculpture we cannot have two cells labelled the same. If an HDA can be sculpted, then we can find an embedding into a bulk. The embedding is by definition injective, meaning it maps two different HDA cells into two different bulk cells. But in the bulk each cell has a unique name in the canonical naming, as either a Chu-label or as an ST-configuration. □

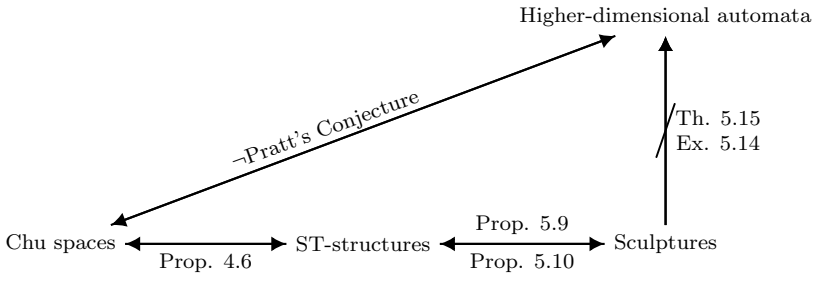
There are more examples of non-sculptures, such as Figure 4.2. The figure is also not representable as ST-structures and is not a sculpture. Both this example and Example 5.14 are HDAs which are also their own unfoldings.

# Chapter 6

# Conclusion

Sculptures have been precisely defined, by following the intuition of Pratt, as well as shown to be in close correspondence with ST-structures and Chu spaces. This nicely captures Pratt's event-state-duality.

We have developed an algorithm to decide whether a higher-dimensional automaton is a sculpture, and it has been used to show that, unexpectedly, some quite simple acyclic higher-dimensional automata are not sculptures. In Theorem 5.15, we showed that a higher-dimensional automata is a sculptures, however, its unfolding is not a sculpture. We believe that this contradicts Pratt's intuition that sculptures suffice for modelling of concurrent behaviour. We may present the results of this thesis by the diagram below:




In Proposition 4.6, we showed the isomorphism between Chu spaces and ST-structures. In Proposition 5.9 and Proposition 5.10, we show that ST-structures and sculptures are isomorphic. Hence, sculptures are also isomorphic to Chu spaces. To strengthen the proofs, we provided examples of higher-dimensional automata which are not sculptures, see Example 5.14. We used this example to show the *non-sculpting* theorem.

# Future work

In this thesis, we presented combinatorial sculpting and is not to be confused with *geometric* sculpting which consists of taking a geometric cube of some dimension and chiseling away hypercubes which one does not want to be part of the structure.

Geometric sculpting has been used by Fajstrup et al. in [7, 6, 9] and other papers to model and analyze so-called PV programs: processes which interact by locking and releasing shared resources. In the simplest case of linear processes without choice or iteration this defines a hypercube with *forbidden hyperrectangles*, which cannot be accessed due to resources access limits. See Fig. 6.1 for an example.

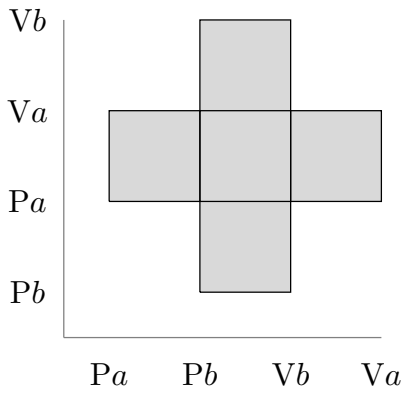



**Figure 6.1** Two PV processes sharing two mutexes. The forbidden area is grayed out.

Technically, geometric sculptures are *Euclidean cubical complexes*; rewriting a proof in [50] will show that such complexes are precisely (combinatorial) sculptures. This will be presented in a current paper, which is being submitted to FOSSACS 2019. Hence, higher-dimensional automata is Euclidean iff it is a sculpture, so that the geometric models for concurrency [7, 6, 9] will show to be closely related to the combinatorial ones [41, 11], through the notion of sculptures. Much work has been done in the *geometric* analysis of Euclidea higher-dimensional automata [8, 23, 6, 31, 44, 50]; by the equivalences of the current work being done in the FOSSACS paper, these results will be made available for combinatorial models.

As we briefly introduced in Section 3.2, the notion of *unfolding* removes iteration and is commonly used to turn a complicated model into a simpler, but potentially infinite one. It is thus expected that even if a higher-dimensional automata cannot be sculpted, its unfolding can, as illustrated by the two simple examples in Fig. 6.2.

However, it can be shown that this is not the case, as witnessed by the example in Figure 6.3 which shows a higher-dimensional automaton which cannot be sculpted and which is its own unfolding. This example features two agents, $a$ and $d$, which compete to choose between two future events $b$ and $c$. If the demon finishes his $d$ event first, then the choice between $b$ and $c$ is a demonic choice, that is, already

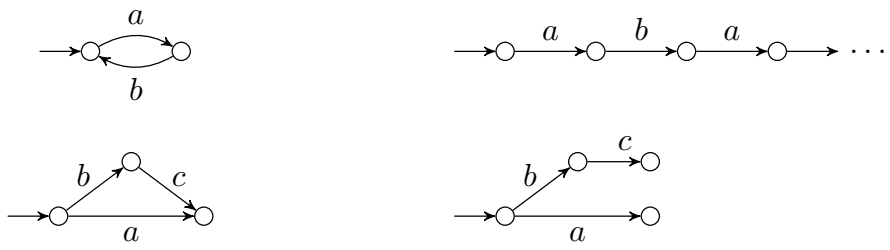



**Figure 6.2** Two simple HDA which cannot be sculpted (left) and their unfoldings (right) which can. (The top-right sculpture is infinite.)

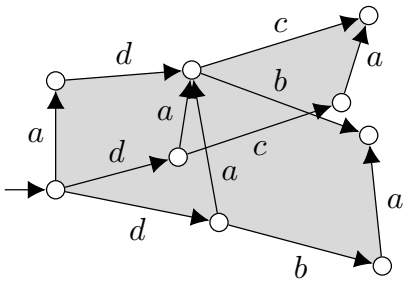



**Figure 6.3** The HDA from [28] of the "speed game of angelic vs. demonic choice".

made when starting the *d* event; if the angel finishes her *a* event first, then we have an instance of an angelic choice between *b* and *c*. This system, introduced in [28], cannot be modeled as a ST-structure, but *can* be modeled as a ST-structure with *cancellation* [28, Section 5].

Even more worrying is the fact that there are (acyclic) HDA which can be sculpted, but their unfoldings cannot; in fact Figure 5.1, is one such example. We showed, in Theorem 5.15, that the unfolding did not return a simpler model, and which seems to contradict Pratt's claim that sculpting suffices for modeling.

In the geometric setting, it can be shown that there are Euclidean cubical complexes whose unfoldings are not Euclidean. Since Goubault-Jensen's seminal paper [24], *directed topology* (or *ditopology*) has been developed in order to analyze concurrent systems as geometric objects [26, 7, 9]. Directed topology has been developed largely in analogy to algebraic topology, by *breaking [its] symmetries* [26], but as shown time and time again, there are unexpected problems turning up.

The mismatch being discovered in the current FOSSACS paper, between Euclidean complexes and unfoldings, is again such an unexpected problem. Unfoldings of higher-dimensional automata have been developed as a directed analogue to *universal covering spaces* in algebraic topology [11, 3], building a universal *dicover* by considering dihomotopy classes of directed paths. There are several other problems with this notion, for example it does not behave well under change of base point,

and finding better definitions of dicovering is active ongoing research, see for example [2].

We may sum up the claims being made here and in the FOSSACS paper as follows:

1. There are acyclic HDA which cannot be sculpted.
2. There is a HDA which cannot be sculpted, but whose unfolding can be sculpted.
3. There is a HDA which can be sculpted, but whose unfolding cannot be sculpted.
4. There is a HDA which can be sculpted, but whose unfolding can be sculpted.

For future work of sculptures and unfoldings of HDA, we would like to apply our decision algorithm to each of the mentioned cases. This thesis has already shown (3) as part of the *non-sculpting* theorem.

# Bibliography


[1] M. A. Bednarczyk. "Categories of Asynchronous Systems". PhD thesis. University of Sussex, 1988.

[2] Jeremy Dubut. "Trees in Partial Higher Dimensional Automata". In: *CoRR* abs/1804.10894 (2018). arXiv: 1804.10894. URL: http://arxiv.org/abs/1804.10894.

[3] Ulrich Fahrenberg. "Higher-Dimensional Automata from a Topological Viewpoint". PhD thesis. Aalborg University, 2005.

[4] Ulrich Fahrenberg and Axel Legay. "Partial Higher-dimensional Automata". In: *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24-26, 2015, Nijmegen, The Netherlands*. 2015, pp. 101–115. DOI: 10.4230/LIPIcs.CALCO.2015.101. URL: https://doi.org/10.4230/LIPIcs.CALCO.2015.101.

[5] Lisbeth Fajstrup. "Dipaths and dihomotopies in a cubical complex". In: *Advances in Applied Mathematics* 35.2 (2005), pp. 188–206. ISSN: 0196-8858. DOI: https://doi.org/10.1016/j.aam.2005.02.003. URL: http://www.sciencedirect.com/science/article/pii/S0196885805000333.

[6] Lisbeth Fajstrup, Eric Goubault, and Martin Raussen. *Detecting Deadlocks in Concurrent Systems*. 1998.

[7] Lisbeth Fajstrup, Martin Raussen, and Eric Goubault. "Algebraic topology and concurrency". In: *Theoretical Computer Science* 357.1-3 (2006), pp. 241–278.

[8] Lisbeth Fajstrup et al. "Components of the Fundamental Category I". In: *Applied Categorical Structures* 12.1 (2004), pp. 81–108. DOI: 10.1023/B:APCS.0000013812.75342.de.

[9] Lisbeth Fajstrup et al. *Directed Algebraic Topology and Concurrency*. 1st. Springer Publishing Company, Incorporated, 2016.

[10] Hubert Garavel. "Reflections on the Future of Concurrency Theory in General and Process Calculi in Particular". In: *Electronic Notes in Theoretical Computer Science* 209 (2008). Proceedings of the LIX Colloquium on Emerging Trends in Concurrency Theory (LIX 2006), pp. 149–164. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2008.04.009. URL: http://www.sciencedirect.com/science/article/pii/S1571066108002247.

[11] R.J. van Glabbeek. *Bisimulations for higher dimensional automata*. Email message, July 7, 1991. 1991. URL: http://theory.stanford.edu/~rvg/hda.

[12] R.J. van Glabbeek. *History preserving process graphs*. Draft. 1996. URL: http://theory.stanford.edu/~rvg/abstracts.html#hppg.

[13] R.J. van Glabbeek. "On the expressiveness of higher dimensional automata". In: *Theoretical Computer Science* 356.3 (2006), pp. 265–290.

[14] Rob J. van Glabbeek. "Petri Nets, Configuration Structures and Higher Dimensional Automata". In: *CONCUR'99*. Vol. 1664. LNCS. (invited talk). Springer, 1999, pp. 21–27.

[15] Rob J. van Glabbeek and Ursula Goltz. "Refinement of actions and equivalence notions for concurrent systems". In: *Acta Informatica* 37.4/5 (2001), pp. 229–327.

[16] Rob J. van Glabbeek and Gordon Plotkin. "Configuration Structures". In: *LICS*. 1995, pp. 199–209.

[17] Rob J. van Glabbeek and Gordon Plotkin. "Configuration structures, event structures and Petri nets". In: *Theor. Comput. Sci.* 410.41 (2009), pp. 4111–4159.

[18] Rob J. van Glabbeek and Frits W. Vaandrager. "The Difference between Splitting in n and n+1". In: *Information and Computation* 136.2 (1997), pp. 109–142.

[19] Rob van Glabbeek and Ursula Goltz. "Equivalence notions for concurrent systems and refinement of actions". In: *Mathematical Foundations of Computer Science 1989*. Ed. by Antoni Kreczmar and Grazyna Mirkowska. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 237–248. ISBN: 978-3-540-48176-8.

[20] Eric Goubault. "Geometric Models for Concurrency". PhD thesis. Ecole Polytechnique, 1995.

[21] Eric Goubault. "Some geometric perspectives in concurrency theory". In: *Homology Homotopy Appl.* 5.2 (2003), pp. 95–136. URL: https://projecteuclid.org:443/euclid.hha/1088453323.

[22] Eric Goubault. "Topological Deformation of Higher Dimensional Automata". In: 2001.

[23] Eric Goubault and Emmanuel Haucourt. "Components of the Fundamental Category II". In: *Applied Categorical Structures* 15.4 (2007), pp. 387–414. DOI: 10.1007/s10485-007-9082-7.

[24] Eric Goubault and Thomas P. Jensen. "Homology of Higher Dimensional Automata". In: Springer-Verlag, 1992, pp. 254–268.

[25] Eric Goubault and Samuel Mimram. "Formal Relationships Between Geometrical and Classical Models for Concurrency". In: *Electronic Notes in Theoretical Computer Science* 283 (2012). Proceedings of the workshop on Geometric and Topological Methods in Computer Science (GETCO), pp. 77–109. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2012.05.007.

[26] Marco Grandis. *Directed Algebraic Topology: Models of Non-reversible Worlds*. New mathematical monographs. Cambridge University Press, 2009.

[27] Vincent Gupta. "Chu Spaces: A Model of Concurrency". PhD thesis. Stanford University, 1994.

[28] Christian Johansen. "ST-structures". In: *J. Log. Algebr. Meth. Program.* 85.6 (2016), pp. 1201–1233. DOI: 10.1016/j.jlamp.2015.10.009. URL: https://doi.org/10.1016/j.jlamp.2015.10.009.

[29] Thomas Kahl. "The homology graph of a higher dimensional automaton". In: *CoRR* abs/1307.7994 (2013).

[30] Saunders MacLane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics, Vol. 5. New York: Springer-Verlag, 1971, pp. ix+262.

[31] Roy Meshulam and Martin Raussen. "Homology of Spaces of Directed Paths in Euclidean Pattern Spaces". In: *A Journey Through Discrete Mathematics: A Tribute to Jiri Matousek*. Ed. by Martin Loebl, Jaroslav Nesetril, and Robin Thomas. Cham: springer, 2017, pp. 593–614. ISBN: 978-3-319-44479-6. DOI: 10.1007/978-3-319-44479-6_24. URL: https://doi.org/10.1007/978-3-319-44479-6_24.

[32] Robin Milner. "Calculi for Synchrony and Asynchrony." In: *Theorethical Computer Science* 25 (1983), pp. 267–310.

[33] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. "Petri nets, event structures, and domain, part I". In: *Theoretical Computer Science* 13 (1981), pp. 85–108.

[34] Mogens Nielsen, Vladimiro Sassone, and Glynn Winskel. "Relationships between models of concurrency". English. In: *A Decade of Concurrency Reflections and Perspectives*. Ed. by J. W. de Bakker, W. -P. de Roever, and G. Rozenberg. Also published as a Technical Report at Department of Computer Science, Aarhus University as DAIMI PB-456. Springer, 1994, pp. 425–476. ISBN: 978-3-540-58043-0. DOI: 10.1007/3-540-58043-3_25.

[35] Hakon Normann, Christian Johansen, and Thomas Hildebrandt. "Declarative event based models of concurrency and refinement in psi-calculi". In: *Journal of Logical and Algebraic Methods in Programming* 85.3 (2016). Interaction and Concurrency Experience, pp. 368–398. ISSN: 2352-2208. DOI: https://doi.org/10.1016/j.jlamp.2015.12.007. URL: http://www.sciencedirect.com/science/article/pii/S2352220815001534.

[36] C. A. Petri. "Concepts of Net Theory." In: *MFCS*. 1973, pp. 137–146.

[37] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. Cambridge, MA, USA: MIT Press, 1991. ISBN: 0-262-66071-7.

[38] Vaughan R. Pratt. "Chu Spaces and their Interpretation as Concurrent Objects". In: *Computer Science Today: Recent Trends and Develop.* Vol. 1000. LNCS. Springer, 1995, pp. 392–405.

[39] Vaughan R. Pratt. "Event-State Duality: The Enriched Case". In: *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*. Ed. by Lubos Brim et al. Vol. 2421. Lecture Notes in Computer Science. Springer, 2002, pp. 41–56. DOI: 10.1007/3-540-45694-5_3. URL: https://doi.org/10.1007/3-540-45694-5_3.

[40] Vaughan R. Pratt. "Higher dimensional automata revisited". In: *Mathematical Structures in Computer Science* 10.4 (2000), pp. 525–548. URL: http://journals.cambridge.org/action/displayAbstract?aid=54599.

[41] Vaughan R. Pratt. "Modeling Concurrency with Geometry". In: *POPL'91*. 1991, pp. 311–322.

[42] Vaughan R. Pratt. "Modeling Concurrency with Partial Orders". In: *J. Parallel Programming* 15.1 (1986), pp. 33–71.

[43] Vaughan R. Pratt. "Transition and Cancellation in Concurrency and Branching Time". In: *Math. Struct. Comput. Sci.* 13.4 (2003), pp. 485–529.

[44] Martin Raussen and Krzysztof Ziemianski. "Homology of Spaces of Directed Paths on Euclidean Cubical Complexes". In: *Journal of Homotopy and Related Structures* 9.1 (2014), pp. 67–84. ISSN: 1512-2891. DOI: 10.1007/s40062-013-0045-4. URL: https://doi.org/10.1007/s40062-013-0045-4.

[45] M. W. Shields. "Concurrent Machines". In: *The Computer Journal* 28.5 (1985), pp. 449–465.

[46] M. W. Shields. "Deterministic asynchronous automata". In: *Automata on Infinite Words*. Ed. by Maurice Nivat and Dominique Perrin. Vol. 192. Lecture Notes in Computer Science. Springer, 1985, pp. 89–98. ISBN: 3-540-15641-0.

[47] A. M. Turing. "Computability and $\lambda$-definability". In: *Journal of Symbolic Logic* 2.4 (1937), pp. 153–163. DOI: 10.2307/2268280.

[48] Glynn Winskel. "Events in Computation". PhD thesis. Department of Computer Science, 1980.

[49] Glynn Winskel and Mogens Nielsen. "Models for Concurrency". In: *Handbook of Logic in Computer Science – vol 4 – Semantic Modelling*. Ed. by Samson Abramski, Dov M. Gabbay, and Tom S.E. Maibaum. Oxford University Press, 1995, pp. 1–148.

[50] Krzysztof Ziemianski. "Directed Path Spaces via discrete vector fields". In: (2017). arXiv: 1708.02055. URL: https://arxiv.org/abs/1708.02055.