Cancer Progression:
Model, Therapy & Extraction

BY

Loes Olde Loohuis

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulllment of the requirements for the degree of Doctor of Philosophy, The City University of New York,

2013

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Rohit Parikh

Date	Chair of Examining Committee

Robert Haralick

Date	Executive Officer

Bud Mishra

Nancy Griffeth

Amotz Bar-Noy

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Cancer Progression: Model, Therapy & Extraction

BY

Loes Olde Loohuis

Advisors: Professors Rohit Parikh and Bud Mishra

In this thesis we develop Cancer Hybrid Automata (CHAs), a modeling framework based on hybrid automata, to *model* the progression of cancers through discrete phenotypes. Both transition timing between states as well as the effect of drugs and clinical tests are parameters in the framework, thus allowing for the formalization of temporal statements about the progression as well as timed *therapies*. Using a game theoretical formulation of the problem we show how existing controller synthesis algorithms can be generalized to CHA models, so that (timed) therapies can be automatically generated.

In the second part of this thesis we connect this formal framework to cancer patient data, focusing on copy number variation (CNV) data. The underlying process generating CNV segments is generally assumed to be memory-less, giving rise to an exponential distribution of segment lengths. We provide evidence from TCGA data suggesting that this generative model is too simplistic, and that segment lengths follow a power-law distribution instead. We show how an existing statistical method for detecting genetic regions of relevance for cancer can be improved through more accurate (power-law) null models.

Finally, we develop an algorithm to *extract* CHA-like progression models from

cross-sectional patient data. Based on a performance comparison on synthetic data, we show that our algorithm, which is based on a notion of *probabilistic causality*, outperforms an existing extraction method.

Acknowledgements

This thesis would not exist if it was not for so many great people I worked with in the last few years.

First and foremost I thank my advisors Bud Mishra and Rohit Parikh. I am deeply grateful to Bud for making me into the researcher that I am today. His big ideas directly influenced my way of thinking about cancer, bio-informatics and computer science. By working together and following his advice I learned the skill of doing research. Thank you for your generosity and unwavering support: our talks always gave me confidence and strength.

I am grateful to my advisor Rohit Parikh for helping me develop at the early stage of my doctoral studies. The (epistemic) logic, game theory and theoretical computer science, that I worked on with Rohit Parikh, form the analytical backbone of this thesis. I am grateful for his encouragement and continued support, even as my interests shifted to more applied research on cancer progression. Later on, during our "breakfast discussions" you always helped me understand the bigger picture and connections with other fields.

I also thank my committee members Nancy Griffeth and Amotz Bar-Noy. It was Nancy who introduced me to bio-informatics, and who has been a great role model, as a researcher, and as a person. I would like to thank Amotz for introducing me the word of algorithms, and for his patience with a complete novice.

My thesis work was funded by the The Graduate Center Enhanced Chancellor's Fellowship and by the NSF CMACS grant, for which I am grateful.

I thank my friend and colleague Andreas Witzel for guiding me through the maze of computer science and computer "stuff". Thank you Andi, for all our discussions, your tireless confidence in me, for Bagel Bob's, and your friendship.

Without Giulio Caravagna, Alex Graudenzi, Daniele Ramazzotti and Marco Antoniotti there would be no Chapter 4 of my thesis. Thank you, the 'Italians', for our work in the last year and all the fun, all around the globe.

Also, I would like to thank the other members of the NYU bioinformatics lab: Ilya Korsunsky, Andrew Sundstrom, Justin Jee and Chang Peng for being such a great group, and for growing together. You guys were my second family.

In LA, where I spent the last year of my PhD, I would like to thank Jim Gimzewski and the members of the Nano and Pico Characterization Lab, who took me in and gave me a place to stay and to feel at home.

Finally, I would like to say thank you to: Melvin Fitting and Sergei Artemov, from whom I learned a lot. Especially our discussions during the early years of graduate school were invaluable to me; Lina Garcia, whose advice and support helped me not only to make it to graduation, but also to raise my children and take care of my family; and my logic friends: especially Can Başkent, Çağil Taşdemir, YunQi Xue (QiQi), for being great companions in graduate school, and for all the wine, cheese, and much-needed coffees.

Above all, I am grateful to Hanna and Jakub, my 'littlest', but biggest support system, and my husband and best friend Tomasz, for his love, help and constant support. Thank you for de-stressing me and making concrete plans with, or for me – even if I woke you up in the middle of the night asking for them. Thank you.

Loes Olde Loohuis

September 2013, New York

Contents

List of Figures

List of Tables

Introduction

Cancer is a disease of evolution. Its initiation and progression are caused by dynamic alterations to the genome (such as point mutations, structural alterations of the genome, DNA methylation and histone modification changes) [61]. These genomic alterations are generated by processes that behave randomly, and since individual tumor cells compete for space and resources, the fittest variants are naturally selected for. For example, if through random mutations a cell acquires the ability to ignore anti-growth signals from the body, this cell may thrive and divide, and its progeny may eventually dominate part of the tumor. This *clonal expansion* can be seen as a *discrete state* of the cancer's progression. Cancer progression can then be thought of as a sequence of these discrete progression steps, where the tumor acquires certain distinct properties at each state. Different progression sequences are possible, but some are much more common than others, and not every order is viable [60, 89].

In the last few decades, an extraordinary amount of effort and money have been spent on trying to understand the mechanisms underlying cancer. Many specific genes and genetic mechanisms have been identified that are involved in different types of cancer, and *targeted therapies* that aim to affect the product of these genes are now being developed at a fast pace [89]. Despite its initial promise (such as the success of imatinib in the treatment of chronic myeloid leukemia [41], or vemurafenib as a drug to treat melanoma with a specific mutation [29]), it is becoming more and more apparent that focusing therapy on just a single gene will not very likely result in a

cure for cancer [56, 24].

Because of its evolutionary nature – with the ability to acquire new mutations while natural selection drives progression – cancer is complex adaptive system, for which complex treatments are required. For example, while a specific treatment may remove a dominant clone completely, causing the patient to be practically tumor-free, it may be that through natural selection small drug-resistant sub-clones grow back to full tumor size.

Effective treatments for cancer need to take into account these dynamics of progression and resistance: they need to be *timed* right [35] and likely include *combinations* of various drugs in correct proportions and dosages [56, 86]. Moreover, rather than seeking for an ultimate *cure* for cancer, a more suitable objective may be to turn the disease into a *chronic* one instead.

In this thesis we aim to capture the dynamics of cancer progression into a simple unified framework, with the hope that the barriers to successful treatment just described can be turned into opportunities instead. In doing so, we use tools from computer science, logic, and statistics.

We start by presenting our framework called *Cancer Hybrid Automaton* (CHA) that formally captures cancer progression through discrete states (in Chapter 1). States in CHA models represent states of the progression, and directed edges among pairs of states define possible progression paths. Transitions take certain durations of time, and the effects of drugs are abstractly modeled, thus allowing for the formalization of temporal statements about the progression as well as notions of timed therapies. The framework is also extended to model *partial observability*, and tests are incorporated into the definition of a therapy as actions that reduce uncertainty about the current state of the progression.

We illustrate our approach through a highly simplified running example of a CHA in which states represent *cancer hallmarks* – states common in (early) cancer

progression originally introduced by Hanahan and Weinberg in [60]), and progression paths represent successive hallmark acquisitions. However, the states of the automaton can represent any set of discrete states at varying levels of abstraction. Examples include phenotypical stages of cancer, a set of affected pathways, and a set of specific genomic aberrations such as genetic mutations at a more mechanistic level.

We then show (in Chapter 2) how the CHA framework not only enables us to formally *describe* cancer progression, but also to *manipulate* its evolution to satisfy certain therapeutic goals. This problem is addressed using a game theoretic formulation of the problem, by building on existing tools and techniques from the controller synthesis literature. We show how algorithms can be designed that generate successful therapies automatically, such that the resulting progression satisfies certain therapeutic goals specified using temporal logic formulas. More precisely, the main result of Chapter 2 (Theorem 9) is that in certain restricted settings CHA control for goals specified in Computation Tree Logic (CTL)[1] is EXPTIME complete. This theorem also extends to CHA models augmented with partial observability.

In the second part of this thesis, we connect our formal framework to actual patient data, by focussing on the problem of *extracting* progression models from available data. In this part, we will not use hallmarks as the states of our progression model, but instead we focus on genes and genetic mutations that play a causal role in cancer progression. These so-called 'driver genes' are the discrete states of our model, and a first step towards progression extraction is that of finding out what the driver genes are. Towards this goal, a lot of precise genetic information has been collected from tumor samples in the last couple of decades. A widely studied type of data, and the one that we focus on in the second part of this thesis, is Copy Number Variation (CNV) data. CNV is structural variation in which relatively large regions of the

[1]See Section 1.2.1 and Section 2.1.1 as well as [33] for details on temporal logic and CTL in particular.

genome are either amplified or deleted, leading to gain- or loss-of-function of the genes contained in the affected regions.

The underlying process generating CNV segments is generally assumed to be memory-less, giving rise to an exponential distribution of segment lengths. In Chapter 3, we provide evidence from TCGA data suggesting that this generative model is too simplistic, and that segment lengths follow a *power-law distribution* instead. This result is important as many tools used to analyze CNV data can be improved by incorporating this power-law distribution hypothesis. Building on the method described in [76], we develop a statistical method for finding driver genes, through more accurate (power-law) null models.

Even though many cancer driver genes and core pathways have been identified in the last few decades (see e.g. [114] for an overview of common cancer genes, and [13, 74] for specific genetic analyses of ovarian carcinoma, and lung adenocarcinoma respectively), relatively little is known about the *dynamics* of cancer progression and the *order* in which these driving events (hallmarks, genetic mutations, etc.) are likely to occur. The main reason for this state of affairs is that information is usually obtained only at one (or a few) points in time, rather than over the course of the disease. Extracting this dynamic information from the available static data is a challenge.

In Chapter 4, we focus on this problem, and we propose a method to extract progression trees and forests using a mathematical notion of *probabilistic causality*. Using synthetic data we show how this method outperforms an existing tree reconstruction algorithm based on correlation.

Finally, we provide some insight into how the various parameters of our model (timing, the effects of drugs), can be estimated from the available cancer data.

Contributions within the Literature

In this section we make the contributions of this thesis explicit.

Automata-based cancer progression model The first and foremost contribution of this thesis is the development of a unified automata-based framework to describe and control cancer progression (in Chapter 1). While many mathematical models of tumor growth and progression have been developed (including stochastic [92], deterministic [105], as well as graph models[39, 11]), and cancer biologists obviously think in terms of discrete disease progression [58], progression timing [51], and combination therapy [24], to the best of our knowledge, no unified framework for describing discrete progression using hybrid automata currently exists. The framework can be used for *diagnostics, prognostics* (using model checking/verification algoritms) and the development of *targeted therapy plans* (using controller synthesis techniques).

Controlling continuous dynamics and a game theoretic model While being hybrid in nature – our framework combines discrete progression with continuous behaviors at a state – CHAs are not standard hybrid automata from the computer science literature (see [62] for an overview). The main difference between CHAs and existing hybrid automata is that in CHAs actions control the dynamics *at a state* by controlling the *rate of the clocks* as opposed to only controlling *transitions* taken directly. However, this difference is mainly conceptual and we provide a way of translating our CHAs into standard hybrid automata in Chapter 2. In this Chapter we show how existing controller synthesis algorithms can be adapted to control CHAs for therapeutic goals described in temporal logic. We focus on a game-theoretic formulation of the control problem as a two-player game played between the controller (the therapist) and a player called 'Nature', in which the therapy is represented as a strategy for the controller.

Even though we are not the first to use a game theoretic framework to aid in cancer therapy design (see e.g.,[53] and [99] for evolutionary game theoretic frameworks), we are the first to use *extensive form games* in which the therapist has many different actions to choose from and a strategy is a timed plan. We treat cancer as a chronic disease, for which therapeutic goals may be complex and described using temporal logic.

Partial observability and control Our CHA framework is extended to capture the therapist's uncertainty by including partial observability and the notion of belief sets and tests to reduce uncertainty (Section 1.4). While we are the first to study uncertainty and belief sets in the context of cancer progression and therapy design, they are at the core of *belief revision*, a topic well-studied by logicians and theoretical computer scientists (see e.g., [52] for a thorough overview, and [27] for an application from the planning literature). In the context of hybrid systems, partial observability has been studied before. The main difference between partial observability in CHAs and in partial observability in existing hybrid systems, is that in CHAs observations are the result of explicit *actions* (performing tests), while in existing frameworks, observations are properties of the *state* and are available 'for free' as soon as a state is reached.

Improved null model of Copy Number Variation in cancer In the second part of the thesis our focus shifts from *describing, verifying* and *manipulating* progression, to *extracting* cancer progression models from static patient data. In Chapter 3 we study copy number variation (CNV) data and we provide evidence from three datasets provided by The Cancer Genome Atlas (TCGA)[2], that in cancer the lengths of the deleted or duplicated segments are not *exponentially*, but *power-law* distributed,

[2]TCGA is a "comprehensive and coordinated effort to accelerate our understanding of the molecular basis of cancer through the application of genome analysis technologies", see `http://cancergenome.nih.gov/`

suggesting a generative mechanism of preferential attachment. This result can be used to improve many tools that analyze CNV data by incorporating this hypothesis. These tools include so-called 'segmenters' that reduce noise in CNV data by combining sets of neighboring data-points in contiguous segments (e.g., GLAD [73], CBS [98], and a method developed by Mishra's group [36]), as well as methods that identify regions containing genes that are relevant for the development of cancer (e.g., methods described in [76, 15]).

As an example, we show in Section 3.2 how null models can be improved to incorporate this finding (Proposition 1), so that one of the subsequent methods can be improved, and genes that drive cancer can be more adequately identified. We also test our model on real patient TCGA CNV data.

Probabilistic causality as a tool to extract progression models In Chapter 4 we develop an algorithm for extracting tree models using *probabilistic causality*, and *probability raising* in particular [107]. Probabilistic notions of causality have been used in biomedical applications before (e.g., to find driver genes from CNV data in [76], and to extract causes from biological time series data in [84]), but never before to infer *progression models* in the *absence* of direct temporal information.

In the literature, several methods have been developed that derive different types of progression models from static data (such as [39, 102, 9, 54, 55]). The method developed by Desper et al. in their seminal paper [39] is (to the best of our knowledge) the only algorithm that also extracts progression *trees* from patient data. Using synthetic data we show that our algorithm outperforms this method. In addition, based on analysis on published real patient data, we show how our method in some cases extracts different progression trees than the ones extracted by Desper's method. We also provide several suggestions for making our method more robust to noise, as well as how other CHA parameters (such as timing and the influence of drugs) can be

extracted and incorporated into the algorithm.

The chapter provides some theoretical results regarding probability raising that are interesting in their own right. The most surprising result is Proposition 2, which relates probability raising between two events with the relative frequency of their occurrence.

Part I

Cancer Progression:

Model and Control

Chapter 1

Cancer Hybrid Automata

The view of cancer as a progressive disease, progressing through discrete states towards a terminal phenotype, bears a striking resemblance to formal models of state-transition machines in computer science.

While many mathematical models of tumor growth and progression have been developed (see e.g., [92] for a stochastic simulation, and [105] for a deterministic model), and cancer biologists think in terms of discrete disease progression (e.g., [58]), to the best of our knowledge, no unified framework for describing discrete progression currently exists.

In this chapter, we aim to fill this gap, by presenting a logical framework called *Cancer Hybrid Automaton* (CHA) that allows us to formally capture cancer progression through accumulation of successive discrete states. States in CHA models represent states of the progression, and directed edges among pairs of states define possible progression paths. Drugs can be thought of as inhibiting or prolonging specific transitions in the automaton. We show how this approach enables us to formally describe cancer progression, automatically verify/model-check its temporal properties, and manipulate its evolution to satisfy certain therapeutic goals.

We illustrate our approach through a highly simplified running example of a cancer

hybrid automaton in which states represent so-called hallmarks, first introduced by Hanahan and Weinberg in [60], and progression paths represent successive hallmark acquisitions.

This chapter is organized as follows. In Section 1.1, we introduce our running example of a CHA based on hallmarks. In Section 1.2, we introduce a basic CHA formalism. In this section, a CHA is modeled as a *finite non-deterministic automaton*. The edges, representing transitions from one progression state (e.g., hallmarks) to the next, are labeled with drugs that can inhibit the transition. A *therapy* is defined as a function that assigns a set of drugs to each finite progression history, or *run*. An execution of a therapy is defined as a run of the CHA that respects the therapy, that is, no transition of the execution is inhibited by the therapy. Our model includes costs by associating a cost vector with each state and each cocktail. Therapies may be selected by comparing costs of possible executions using a notion of Pareto dominance, in addition to the required qualitative properties specified in CTL.

In Section 1.3 we extend the CHA framework to include real time. In this model, transitions take certain durations of time, and drugs can prolong (or stop) the transition process. This is modeled using a hybrid automaton with multiple clocks.[1] Clock constraints on the edges and clock invariants at the states restrict the possible progressions of the system. Multiple clocks are needed to allow for the scenario that a drug affects the transition to possible next states in different ways. Possible runs and therapies of a timed CHA now include the clock values.

In Section 1.4 we introduce uncertainty into the framework. We model the fact that the oncologist may have only partial knowledge about the tumor's internal state, which we model by keeping track of his belief set. Tests are incorporated into the definition of a therapy as actions that reduce uncertainty about the current state. In our framework, tests have costs, but take no time. To integrate the observer's information about

[1] The continuous dynamics of these clocks justify the term *hybrid* in 'cancer hybrid automaton'.

the system, we add epistemic operators to Timed CTL. In Section 1.4.4 we give a translation from therapies for timed CHAs with partial observability into conditional plans.

In Section 1.5 we briefly discuss tumor *heterogeneity*, and how it can be modeled and controlled using our framework.

Finally, in Section 1.6, we present a simple liver automaton as an example of a system of the host organism that may be affected by the therapy. These systems can be combined with CHAs using parallel composition.

Section 1.7 concludes with a discussion of several possible extensions of our model. Most of the ideas and results presented in this chapter and the next have been presented in [94] and [95].

1.1 An example: the hallmarks of cancer

Among other theories, the view of cancer as a disease progressing through discrete states, is reflected in the so-called *hallmarks of cancer* proposed by Hanahan and Weinberg [60], and this theory has become one of the predominant ways of thinking about cancer, solidified through many further publications and experiments.[2] A more recent article by the same authors [61] reviews and consolidates the new insights of the last decade.

According to the model proposed by Hanahan and Weinberg, tumors must necessarily acquire certain "intermediate" phenotypical progression states (which they call hallmarks) culminating in the "final" hallmarks of tissue invasion and metastasis. As the authors write,

Simply depicted, certain mutant genotypes confer selective advantage on

[2] As March 2011, their original paper [60] was Cell's most cited article `http://www.sciencedaily.com/releases/2011/03/110316113057.htm`, and as of August 19 2013 it has received more than 17,000 citations.

> *subclones of cells, enabling their outgrowth and eventual dominance in a local tissue environment. Accordingly, multistep tumor progression can be portrayed as a succession of clonal expansions, each of which is triggered by the chance acquisition of an enabling mutant genotype. [61, p. 658]*

The current list of cancer hallmarks includes the abilities to reproduce autonomously, to ignore anti-growth signals, to signal for formation of new blood vessels, as well as handful of other phenotypes. Hallmarks can be obtained in various different orders, but not every order is viable. Intuitively, a hallmark can be acquired by a dominant sub-population of cells if it conveys a selective advantage compared to the other phenotypes acquired in that population. For example, in a wildly growing cluster of cells, the ability to signal for new blood supply, and thus nutrients, oxygen, and waste disposal, will allow the respective sub-population to outgrow the others.

Most hallmarks are acquired through mutations of very specific sets genes. As a result, many of the targeted drugs, administered individually or combinatorially in a cocktail, which have been developed in recent years, aim to influence the function of the products of these genes [89] and thus cancer's evolution from specific hallmarks. For example, the vascular endothelial growth factor (VEGF) signals for creation of new blood vessels (*angiogenesis*), and the drug Avastin inhibits the associated signaling pathway, thus preventing growing tumors from obtaining the needed blood supply. Current therapies usually target only the observed hallmark at any instant, and rarely take into account the potential hallmarks that may evolve in the future and the temporal structure of the underlying evolution.

A simple, intuitive exemplary CHA is shown in Figure 1.1. It comprises the following hallmarks (see [60] for more details):

SSG: Self-sufficiency in growth signals. Roughly speaking, cells no longer depend on external growth-promoting signals, but grow autonomously. Usually, such a state is associated with a gain of function of an oncogene or a loss of function of

a tumor suppressor gene.

IAG: Insensitivity to anti-growth signals. Cells with this hallmark continue to grow even in the presence of inhibiting signals. Usually, certain cell-cycle checkpoints are no longer properly regulated.

Ang: Sustained angiogenesis. This state enables a cancer cell to signal for the construction of blood vessels.

LRP: Limitless replicative potential. While most normal cells can only divide a certain number of times, cells with this hallmark can divide without limits. In this state, a cancer cell may upregulate telomerase to restore telomere lengths.

EvAp: Evading apoptosis. Normally, cells have a program for controlled cell-death, which is used to remove damaged or otherwise unwanted cells. This program is disabled in this hallmark, which allows cells with highly corrupted DNA to survive – thus facilitating cancer progression further.

M: Metastasis. This state enables cancer cells to spread from their original location to other parts of the body.

Various possible progressions through these hallmarks can be seen as transitions as in Figure 1.1 (note that this is a simplified and incomplete model). For example, Ang can be acquired after SSG and IAG. Moreover, if a growing tumor fails to acquire Ang, it may starve; in this case, a solid tumor is unable to grow further and attain the later hallmarks. For simplicity, it may be modeled as a transition to the normal state. [3]

In this example, the therapy "give the drug Avastin whenever a state leading up to Ang is reached" will prevent the cancer from reaching M.

[3]This is a very simplistic example, as cancer cells that are oxygen-deprived for a sustained period of time may in fact become more invasive and dangerous [113].

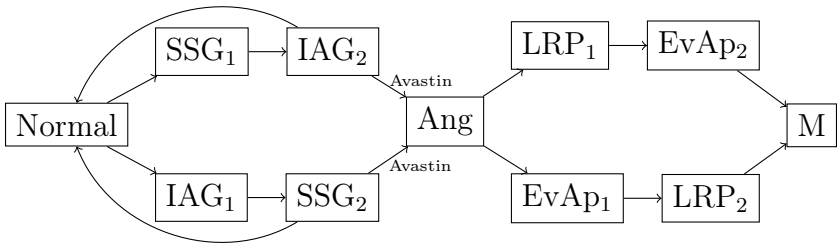

Figure 1.1: A simple CHA whose progression can be stalled by a VEGF-inhibitor such as Avastin.

1.2 Formal model

In the following, we start with a preliminary and simple formalization of the notions described above. We will successively extend the formal model in the later sections.

We assume a global set $\mathcal{D}$ of **drugs**.

Definition 1. *A **Cancer Hybrid Automaton (CHA)** is a tuple*

$$H = (V, E, v_0) \ ,$$

where

- V *is a set of states,*[4]
- $E \subseteq V \times 2^{\mathcal{D}} \times V$ *is a set of directed edges labeled with sets of drugs, and*
- $v_0 \in V$ *is the initial state.*

We usually omit v_0 *and write just* (V, E).

Intuitively, an edge (v, D, v') represents a transition from state v to state v' that can be inhibited by any drug from the set $D \subseteq \mathcal{D}$. We allow several drugs to be given simultaneously and refer to such sets $C \subseteq \mathcal{D}$ of drugs as **cocktails**. Given a cocktail

[4]Strictly speaking, in the case of hallmarks, a state corresponds to a subset of hallmarks that have been acquired, fot this reason we include subscripts $_1$ and $_2$.

C, the edge $(v, D, v') \in E$ is *inhibited* by C if $C \cap D \neq \emptyset$. Given a state v and a cocktail C, v can transition to v' under C, in symbols $v \xrightarrow{C} v'$, if there is an edge (v, D, v') that is not inhibited by C. Note that we allow multiple edges (with different labels) between the same two states. To prevent a transition between two states, all edges connecting them need to be inhibited, which is one reason for considering cocktails rather than just single drugs. We assume that for every state v and every cocktail C there exists some state v' such that $v \xrightarrow{C} v'$ (possibly $v' = v$, these edges were omitted in Figure 1.1).

A **run** of a CHA $H = (V, E, v_0)$ is a sequence of transitions in E. Let $Runs(v, H)$ denote the set of runs that start in v. We write $Runs(H)$ for $Runs(v_0, H)$, and by $Runs_{\mathsf{f}}(v, H)$ we denote the set of finite runs from $Runs(v, H)$.

We now formalize how it is possible to *interfere* with the progression of the system.

Definition 2. *A **therapy** is a function $\pi : Runs_{\mathsf{f}}(H) \to 2^{\mathcal{D}}$. A **possible execution** of π in H is a run*

$$S = v_0 v_1 v_2 \dots \ ,$$

such that for each $i \geq 0$, $v_i \xrightarrow{\pi(S_i)} v_{i+1}$, where S_i denotes the initial segment of S up to step i.

To illustrate these definitions, consider the following example based on our toy example automaton of Figure 1.1.

Example 1. *Given the example CHA H of Figure 1.1, a possible* run *of H (in case no drugs are administered) is the sequence*

$$Normal \xrightarrow{\emptyset} SSG_1 \xrightarrow{\emptyset} IAG_2 \xrightarrow{\emptyset} ANG \xrightarrow{\emptyset} EvAp_1 \xrightarrow{\emptyset} LRP_2 \xrightarrow{\emptyset} M.$$

We can define a successful therapy *π for H as follows: Given a run S, let $\pi(S) = \{Avastin\}$ if the last state of S is IAG_2 or SSG_2, and $\pi(S) = \emptyset$ otherwise. Every*

possible execution of this therapy will halt the cancer progression before angiogenesis is reached.

Definition 3. ***Costs*** *are given by the following (overloaded) function, for some finite dimension n:*

- $c : V \to \mathbb{R}^n_{\geq 0}$ *specifying costs of states,*
- $c : 2^{\mathcal{D}} \to \mathbb{R}^n_{\geq 0}$ *specifying costs of cocktails.*

Thus, both states and cocktails have costs assigned to them, represented as n-dimensional vectors. Dimensions may include toxicity of the drugs, monetary cost of the drugs, discomfort for the patient, etc.

The cost of a possible execution $S = v_0 v_1 v_2 \ldots$ of a therapy π with discount factor $0 < d \leq 1$ *is*

$$c(S, \pi, H) = \sum_{i \geq 0} d^i \big(c(v_i) + c(\pi(S_i))\big) \ .$$

The set of possible costs of π for a CHA H is

$$c(\pi, H) = \{c(S, \pi, H) \mid S \text{ is possible execution of } \pi \text{ in } H\}.$$

Now that we have a definition of the set of possible costs of a therapy, we can compare different therapies with respect to their costs.

Definition 4. *A cost vector $x \in \mathbb{R}^n$* ***Pareto-dominates*** *another vector $x' \in \mathbb{R}^n$, in symbols $x \prec x'$, iff for each $1 \leq \ell \leq n$ we have $x_\ell \leq x'_\ell$ and for some $1 \leq \ell \leq n$ we have $x_\ell < x'_\ell$.*

A therapy π ***Pareto-dominates*** *another therapy π' in a CHA H if for each $x \in c(\pi, H)$ and $x' \in c(\pi', H)$ we have $x \prec x'$. The set of* ***candidate therapies*** *for*

H is

$$\Theta(H) = \{\pi \mid \pi \textit{ is not Pareto-dominated in } H\} \ .$$

For the special case of 1-dimensional costs (or if there is a function to aggregate cost vectors into single scalars), the set of candidate therapies is the set of therapies whose best-case cost is not higher than some other therapy's worst-case cost.

This definition of a set of candidate therapies is a very conservative one, in that it includes any therapy that is not overtly worse than some other therapy. There are different possibilities for defining the set of candidate therapies, or for pruning the set further. Examples of such strategies for pruning the set further include, i.e., choosing those strategies that lead to the best worst-case outcome, or *maximax*, i.e., choosing those strategies that lead to the best best-case outcome. However, making these decisions depends on the risk attitude of patient and doctor which may not be fully formalizable. Therefore we include all the potentially relevant therapies in the set of candidate therapies. [5]

In order to be clinically applicable, a CHA model may need to be *personalized* for any given patient or cancer type. The reason for this being that different cancer (sub-)types have different progression paths/timings. This personalization will result in families of CHAs, with different sets of candidate therapies for each type. One possible application that can be used to analyze a set of such richer models is the notion of *universal therapies*. That is, for families of automata, we can ask whether there are any therapies that are successful for all of the included automata. Such therapies can result in faster and cheaper treatments.

To be able to apply therapies across different automata, their domain must be the same. This requirement can be satisfied, for example, by considering CHAs that contain the same set of hallmarks, and therapies that either depend only on the

[5] In [100], a formal game theoretic framework is introduced that allows modeling different agents (players) using different types.

current state, or that have the set of all sequences of states as domain. The following definition applies to therapies on such unified domains.

Definition 5. *Given a family $\mathcal{H}$ of CHAs, the set of* ***(universal) candidate therapies*** *for $\mathcal{H}$ is*

$$\Theta(\mathcal{H}) = \bigcap_{H \in \mathcal{H}} \Theta(H) \ .$$

A set θ of therapies ***covers*** *$\mathcal{H}$ if*

$$\theta \cap \Theta(H) \neq \emptyset \text{ for all } H \in \mathcal{H} \ .$$

Note that if $\Theta(\mathcal{H}) \neq \emptyset$ then for each $\pi \in \Theta(\mathcal{H})$, $\{\pi\}$ covers $\mathcal{H}$.

1.2.1 Temporally extended goals: CTL

We have seen in the previous section that therapies can be compared according to their costs. Thus, the problem of finding the right therapy can be viewed as an optimization problem. It can, however, be necessary to have more detailed control over the therapeutic objectives. Simple reachability properties can be used as goals, such as "metastasis must never be reached". For more expressivity we can use Computation Tree Logic (CTL) [33] to specify goals.[6]

Example 2. *The goal $\mathsf{AG}\neg M$ states that metastasis is never reached. Another possible goal could be*

$$\mathsf{AG}(Ang \rightarrow \mathsf{AG}\neg EvAp) \ .$$

This sentence means that whenever sustained angiogenesis is acquired, then at no point in the future the capability of evading apoptosis will be obtained.

Note that the example CHA of Figure 1.1 controlled by the therapy defined in Example 1 satisfies both of these goals.

[6] For more details on temporal logic and CTL, see Section 2.1.1.

One may be interested in checking properties of the CHA itself, without application of a therapy. This goal can be achieved by using CTL model checking (see, e.g., [34]). CTL properties can also be checked on the possible executions of a given pair of therapy and untimed CHA. Supervisory control for finite automata with CTL goals is known to be EXPTIME-complete, and controller synthesis algorithms exist [79].

The above representation of a cancer automaton is intuitive, but it does not include *timing*. It fails to model the fact that some transitions could be very short while others may take many years. In the next section we introduce timed CHAs, which are automata equipped with a set of real-valued variables, denoted as *clocks*, and constraints on the edges and states restrict the progression of the system. As hinted earlier, this model will be a special kind of hybrid automaton, justifying the word *hybrid* in 'cancer hybrid automata'.

1.3 Timed CHAs

The framework we built so far is somewhat idealized in that transitions occur spontaneously and drugs can switch off transitions completely. More realistically, transitions would take certain durations of time. For example, in pancreatic cancer, it takes about a year for K-*ras* mutations in a cell to lead to neoplasms (so-called PanINs) [71]. Also, it has been estimated that it takes 17 years for a large benign tumor to evolve into an advanced cancer, but < 2 years for cells within that cancer to acquire the ability to metastasize [80]. To model durations, we will now add a notion of *time* to our CHA framework, as well as how drugs *slow down* (or stop) the progression.

We start with the assumption that the acquisition of a hallmark requires a certain minimum amount of time.[7] Only after that time a given transition will be possible, and as mentioned, drugs can be used to prolong this time. Further, we allow states

[7]Because currently most cancer patient data is static in nature, the problem of determining how long a transition takes is not an easy one. We will discuss this problem and several of its possible solutions in Section 4.7.

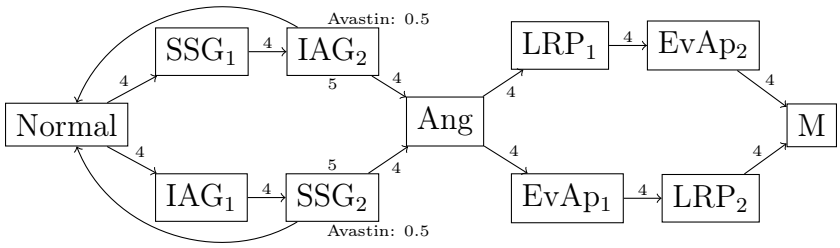

Figure 1.2: A simple timed CHA. The edges are labeled with the minimum times needed to make the respective transitions. In the two states that lead up to Angiogenesis, Avastin can be given to slow down the progress by half. Those states are labeled with invariants, and depending on the precise timing, these invariants can force the system back to Normal before the transition to Angiogenesis is possible.

to have *invariants*, specifying the maximum time that the system can remain in the respective state. For example, a tumor may only be able to remain in a state of unbounded growth without angiogenesis for a certain number of months.

Figure 1.2 shows the automaton from Figure 1.1 with timing information added, illustrating this intuition. We formalize the extension in the following.

We assume a finite set X of real-valued variables called *clocks*, over which the set of *constraints* $\mathcal{C}(X)$ is generated according to the grammar

$$\phi ::= x \geq k \mid \phi \wedge \phi \ ,$$

where $k \in \mathbb{N}$ and $x \in X$. A *valuation* of the variables in X is a mapping $\mathsf{val} : X \to \mathbb{R}_{\geq 0}$. We denote the null valuation $x \mapsto 0$ by 0. By $\mathsf{val} \models \phi$ we denote that val satisfies ϕ.

Definition 6. *A **timed CHA** is a tuple $H = (V, E, v_0, \ell, \rho)$ where*

- *V is a set of states,*
- *$E \subseteq V \times \mathcal{C}(X) \times V$ is a set of directed edges each labeled with a clock constraint,*
- *$v_0 \in V$ is the initial state,*
- *$\ell : V \times X \to \mathbb{N}$ is a partial function specifying the time limit (if any) for each*

clock that the system can remain in a given state (this is also called the invariant*), and*

- $\rho : V \times \mathcal{D} \times X \to \mathbb{R}_{\geq 0}$ *yields a function specifying how a given drug influences the clocks at a given state.*

Intuitively, at a given state v, the drug d modifies the clock rate, by slowing down or speeding up the clock x as specified by a multiplicative factor $\rho(v, d, x)$. When the factor is 1, the drug has no effect on that clock, and when it is 0, it effectively stops the clock from progressing [8].

If several drugs have an effect on a clock, we assume that their factors are multiplied. We extend ρ to cocktails by setting $\rho(v, C, x) = \prod_{d \in C} \rho(v, d, x)$ for any cocktail $C \neq \emptyset$, and by convention, $\rho(w, \emptyset, x) = 1$.

A directed edge (v, ϕ, v') represents a transition from v to v' that can take place once the time constraint ϕ is satisfied.

We assume that for each state v that has a time limit for a clock x, there is an outgoing edge (v, ϕ, v') such that $\mathsf{val} \models \phi$ for all val with $\mathsf{val}(x) = \ell(v, x)$.[9] This edge specifies the behavior of the system if the respective clock reaches its time limit.

The cost functions in the context of timed CHAs are the same as those for the untimed version, but with a timed interpretation: $c(v)$ is the cost of staying at state v per time unit (days/weeks/months/years), and $c(C)$ is the cost of administering a drug cocktail C per time unit.

We next see how to adapt the definitions related to runs of a CHA to the timed version, starting with the notion of a *timed state*.

Definition 7. *A* ***timed state*** *of a timed CHA (V, E) is a tuple $(v, \mathsf{val}) \in V \times \mathbb{R}^X$,*

[8] The effect of drugs on the clocks can be estimated by comparing slopes of Kaplan Meier survival curves of patients treated with different drugs [44]. We will provide more details on this in Section 4.7.

[9] Note that we may require $\mathsf{val} \models \phi$ even for valuations that exceed some other clock's invariant; however, this condition does not have an effect since we only allow $\geq$ constraints on the edges.

where v is a state and val *a clock valuation. There are two types of* ***transitions*** *between timed states:*

1. Delay *transitions, in symbols* $(v, \mathsf{val}) \xrightarrow{\delta, C} (v, \mathsf{val}')$, *where*

   - $\delta \in \mathbb{R}_{>0}$ *represents the (real) time delay,*
   - C *denotes the cocktail active during that time,*
   - $\mathsf{val}'(x) = \mathsf{val}(x) + \delta\rho(v, C, x)$ *for all* x, *and*
   - $\mathsf{val}'(x) \leq \ell(v, x)$ *for all* x *with* $\ell(v, x)$ *defined.*

2. State *transitions, in symbols* $(v, \mathsf{val}) \to (v', 0)$, *where*

   - *there is an edge* $(v, \phi, v') \in E$ *with* $\mathsf{val} \models \phi$.

Note that whenever a state transition takes place, the clocks are reset. This strategy simplifies our presentation and could be replaced by explicit clock resets as common in the literature.

This setup includes the special case where there is one clock unaffected by any drug, representing real time. Invariants over that clock can be used to specify, for example, the duration over which the tumor can remain in a certain state. [10]

This timed setup can also emulate the concept of edges labeled with drugs that inhibit them. This model can be constructed as follows: Suppose we want to model an edge between two states v, v' that can be inhibited by a drug d. Then we can introduce a clock variable $x_{d,v'}$ with $\rho(v, d, x_{d,v'}) = 0$, and add a constraint $x_{d,v'} \geq z$ to the edge between v and v', for some $z > 0$. As long as drug d is given before the constraint is satisfied, the transition will be inhibited. However, once the constraint is satisfied, the tumor has advanced too far and it is no longer possible to inhibit the transition.

[10] This clock is implicitly included in our example CHA and is used to measure the invariant 5 of IAG_2 and SSG_2 in Figure 1.2

A *run* in the case of a timed CHA H is a non-Zeno[11] sequence of delay and state transitions. Similar as before, let $Runs((v, \mathsf{val}), H)$ denote the set of runs that start in (v, val). We write $Runs(H)$ for the set $Runs((v_0, 0), H)$, and $Runs_{\mathsf{f}}((v, \mathsf{val}), H)$ for the set of finite runs from $Runs((v, \mathsf{val}), H)$.

Definition 8. *A **therapy** is a function $\pi : Runs_{\mathsf{f}}(H) \to 2^{\mathcal{D}}$. A **possible execution** of π in H is a run*

$$S = (v_0, 0)(v_1, \mathsf{val}_1)(v_2, \mathsf{val}_2) \cdots$$

such that for all i with delay transitions $(v_i, \mathsf{val}_i) \xrightarrow{\delta, C} (v_{i+1}, \mathsf{val}_{i+1})$,[12] for every $0 \leq \delta' < \delta$

$$\pi((v_0, 0) \ldots (v_i, \mathsf{val}_i)(v_i, \mathsf{val}_i + \delta' \rho(v_i, C))) = C,$$

where $\rho(v_i, C)$ denotes the partial evaluation of ρ, i.e., the function $x \mapsto \rho(v_i, C, x)$.

This last condition ensures that the therapy does not change during a transition, or, put differently, that a change in therapy is always reflected by starting a new transition.

Example 3. *Given the example CHA H of Figure 1.2, a possible* run *of H (in case no drugs are administered) is the sequence*[13]

$$(Normal, 0) \xrightarrow{4, \emptyset} (Normal, 4) \to (SSG_1, 0) \xrightarrow{5, \emptyset} (\ SSG_1, 5) \to (IAG_2, 0) \ldots$$
$$\xrightarrow{4.5, \emptyset} (IAG_2, 4.5) \to (Ang, 0) \xrightarrow{4.5, \emptyset} (Ang,\ 4.5) \to (EvAp_1, 0) \ldots$$
$$\xrightarrow{7, \emptyset} (EvAp_1,\ 7) \to (LRP_2, 0) \xrightarrow{4, \emptyset} (LRP_2, 4) \to (M, 0).$$

[11] These are sequences not containing an infinite chain of timed transitions with convergent total duration.

[12] Note that $v_i = v_{i+1}$.

[13] For simplicity of exposition, only one clock is displayed, but technically the automaton has two clocks and Avastin only inhibits one of them.

We can define a successful therapy π *for* H *as follows: Let*

$$\pi(S'(IAG_2,\ n)\ ) = \pi(S'(SSG_2,\ n)) = \{Avastin\},$$

For every finite run S'*, and* $\pi(S) = \emptyset$ *otherwise. Similar to the untimed case, the therapy states that the drug Avastin should be given as soon as any state directly prior to reaching angiogenesis is reached, and nothing should be done otherwise.*[14] *Every possible execution of this therapy will halt the cancer progression, and force the system back to Normal before angiogenesis is reached*[15]*. A possible execution of this therapy is the following run:*

$$(Normal, 0) \xrightarrow{4,\emptyset} (Normal, 4) \to (SSG_1, 0) \xrightarrow{5,\emptyset} (\ SSG_1, 5) \dots$$
$$\to (IAG_2, 0) \xrightarrow{5,\{Avastin\}} (IAG_2, 2.5) \to (Normal, 0) \dots$$

For any finite run $r \in Runs_{\mathfrak{f}}(H)$, we denote its *duration* as

$$\tau(r) = \sum_{0 \le j < len(r)} \begin{cases} \delta & \text{if } r_j \xrightarrow{\delta, C} r_{j+1} \text{ for some } \delta, C \\ 0 & \text{otherwise,} \end{cases}$$

where $len(r)$ denotes the length of the state sequence in r and r_i its initial segment of length i.

Definition 9. *Given a CHA* H *and a possible execution* S *of a therapy* π*, the* ***cost***

[14] Technically, one could start giving Avastin as late as when the clock reaches a value of (almost) 3, because in this case due to the slowing down effect of Avastin before the value reaches (almost) 4, and a transition to Ang becomes possible, the invariant of 5 will already be reached.

[15] assuming, of course, that the treatment starts before Ang is reached.

of S given π with discount factor $0 < d \leq 1$ is

$$c(S, \pi, H) = \sum_{i \geq 0} \frac{1}{d} \left(e^{-d\tau(S_i)} - e^{-d\tau(S_{i+1})} \right) (c(v_i) + c(\pi(S_i)))$$

(as before, by S_i we denote the initial segment of S up to step i). This simple discounting function does not necessarily capture a real patient's preferences, but any convergent function will work in its stead. We will consider more realistic functions in the future, which can potentially be designed on a case-by-case basis depending on the patient's valuation.

The set of possible costs of π in a timed CHA H is the set of costs of possible executions of π,

$$c(\pi, H) = \{c(S, \pi, H) \mid S \text{ is possible execution of } \pi \text{ in } H\}.$$

The notions of Pareto dominance and universal therapies carry over from untimed CHAs.

1.3.1 Timed CTL

We can extend the CTL goals of the previous section to include time [2]. For example, the goal $\mathsf{AG}_{\leq 20}\neg\mathrm{M}$ says that metastasis is not reached within 20 time units (e.g., 20 years). This kind of goal represents the approach of turning cancer into a chronic disease, rather than trying to cure it completely. For example, the above formula may be appropriate for a patient of sixty years of age, who may then be able to get a less strenuous therapy, while for a younger patient the time requirements may be more extensive.

Out of all the therapies satisfying a CTL goal, the best ones may be chosen either by a separate cost optimization, or by incorporating cost requirements into the

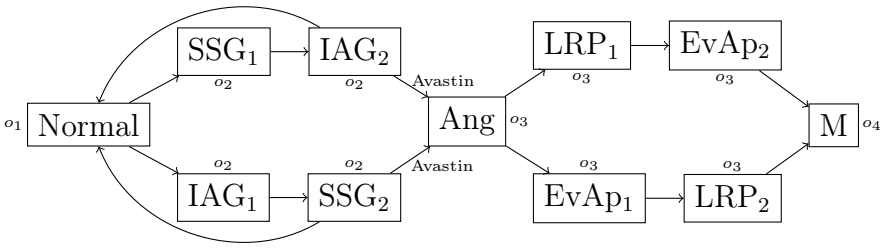

Figure 1.3: A simple hallmark automaton with test observations $o_1, \ldots, o_4$.

formulas using a weighted version of CTL [19].

1.4 Partial observability and tests

So far, we assumed perfect information about the state of the system. In reality however, a clinician will only have partial observations of the tumor's internal state. To reduce uncertainty about the current state of the cancer progression, tests can be performed. In this section, we extend our formal framework to include partial observability and tests, both for untimed and timed CHAs.

1.4.1 Tests in untimed CHAs

We view tests as functions mapping states to observations.[16] See Figure 1.3 for an example of such a test with 4 possible observations. When the test yields observation o_2, we know that the system is in a state prior to acquiring sustained angiogenesis, and that we can give Avastin to inhibit the progression to a hallmark promoting construction of new blood vessels to the tumor. A more fine-grained test, or another test with intersecting observations, would have to be performed to determine the state more

[16]The test we describe here are deterministic, i.e., for any given state, a certain test always leads to the same observation. In the literature, non-deterministic tests are common, where a test may lead to one of a set of possible observations. Our framework can be extended in the same way, but from the biological perspective, that would mean that there are different mechanistic causes for the system being in that state. In that case, we recommend refining the model to have different states representing these different causes.

precisely, e.g., whether it is in the upper or in the lower branch of the automaton, and thus whether other potential drugs should be preferred.

Formally, for a CHA (V, E) we assume a set T of **tests** and a set O of **observations**. Each test $t \in T$ is a function $t : V \to O$, inducing a partition on the set of states. When performing test t while the system is in state v, the resulting observation allows the conclusion that the system is in one of the states in the equivalence class of v with respect to that partition.

We now extend the notion of therapy to include tests. We assume that tests only acquire information, without affecting the state of the system. That is, given a test t and a state v the system can only *transition* to v itself: $v \xrightarrow{t} v$.

We can keep track of the information that results from tests by adding belief sets to runs. A belief set is a subset of states that the system may be in at a given moment. We augment states with belief sets to obtain *belief states*.

Definition 10. *A **belief state** of a CHA (V, E) is a tuple (v, b), where*

- *$v \in V$ a state,*
- *$b \subseteq V$ with $v \in b$ is a belief set.*[17]

There is a transition *from belief state (v, b) to (v', b') under $C \in 2^{\mathcal{D}} \cup T$ if*

- *$v \xrightarrow{C} v'$ and*
- $$b' = \begin{cases} [b]_{\xrightarrow{C}} & \text{if } C = C \in 2^{\mathcal{D}} \\ \{v' \in b \mid t(v') = t(v)\} & \text{if } C = t \in T \end{cases}$$

where $[X]_R$ denotes the image of set X under relation R, i.e., $[X]_R = \{x' \mid (x, x') \in R \text{ with } x \in X\}$.

[17] Note that belief states correspond to *pointed models* in epistemic logic, in the sense that they consist of a set of *possible* states with a distinguished *actual* one.

In symbols, we write $(v, b) \xrightarrow{C} (v', b')$. In addition to an initial state v_0, we now also have an initial belief set b_0. So a CHA is now a tuple (V, E, v_0, b_0), and a **run** of a CHA H is now a sequence of transitions over belief states. As before, $Runs((v, b), H)$ denotes the set of runs that start in (v, b). We write $Runs(H)$ for $Runs((v_0, b_0), H)$, and by $Runs_{\mathsf{f}}((v, b), H)$ we denote the set of finite runs from $Runs((v, b), H)$.

We now extend the notions of therapies and their execution to include tests and belief sets.

Definition 11. *A **therapy** is a function $\pi : Runs_{\mathsf{f}}(H) \to 2^{\mathcal{D}} \cup T$. It is **uniform** if it only depends on the belief sets.*[18] *We only consider uniform therapies, without explicitly mentioning it.*

*A **possible execution** of π in H starting with (v_0, b_0) is a run*

$$S = (v_0, b_0)(v_1, b_1)(v_2, b_2) \dots \quad ,$$

such that for each $i \geq 0$, $(v_i, b_i) \xrightarrow{\pi(S_i)} (v_{i+1}, b_{i+1})$.

Example 4. *Given the example CHA H of Figure 1.3 with one test t, a possible therapy π for H would be to perform a test immediately upon starting the treatment, administering Avastin whenever SSG_2 and IAG_2 are in the resulting belief set, and performing a new test whenever something can be learned from it (that is, whenever the partition induced by t refines the current belief set).*

A possible execution of this therapy is the following run (in which the initial belief set is $\{Normal, SSG_1, IAG_1, SSG_2, IAG_2, Ang\}$, the first test yields observation o_2, and the second o_1):

[18] More precisely, if for any two runs $r = (v_0, b_0)(v_1, b_1) \dots (v_k, b_k)$ and $r' = (v'_0, b_0)(v'_1, b_1) \dots (v'_k, b_k)$ which agree on their belief sets, we have $\pi(r) = \pi(r')$.

$$(\{Normal, SSG_1, IAG_1, SSG_2, IAG_2, Ang\}) \xrightarrow{t} (\{SSG_2, IAG_2\}) \xrightarrow{\{Avastin\}}$$
$$(\{Normal, SSG_2, IAG_2\}) \xrightarrow{t} (\{Normal\}) \xrightarrow{\emptyset} (\{Normal, SSG_1, IAG_1\}) \dots$$

Note that this particular therapy does guarantee that cancer progression will be halted and metastasis will not be reached, assuming of course that the initial state is before Ang.

We also extend the definition of costs, using $c : T \to \mathbb{R}^n_{\geq 0}$ to specify costs of tests. The definition of cost of an execution then is the same as in definition 3, and we can proceed with the notion of possible costs.

Definition 12. *The set of possible costs of π for a CHA H is*

$$c(\pi, H) = \{c(S, \pi, H) \mid S \text{ is a possible execution of } \pi \text{ in } H \text{ starting with } (v, b_0) \text{ for any } v \in b_0\}.$$

The remaining notions such as Pareto dominance, candidate therapies, and universal therapies remain unchanged.

1.4.2 Epistemic and temporally extended goals

Given that we now have a framework that captures not only the actual behavior of the system but also the observer's (e.g., oncologist's) information about it, we need to reflect this additional aspect in the formal language that defines goals. This goal can be achieved by adding an epistemic modality K to the logic, which intuitively means "it is known that".

Instead of the previously mentioned goal $\mathsf{AG}\neg\mathrm{M}$, we can now express that it is known that metastasis is never reached by stating $K\mathsf{AG}\neg\mathrm{M}$.

Another, somewhat more complex, goal is

$$\mathsf{AG}\big(\mathrm{Ang} \rightarrow \big((\neg \mathrm{M} \wedge \mathsf{AX}\neg \mathrm{M})\,\mathsf{U}\,K\,\mathrm{Ang}\big)\big)$$

which intuitively says that whenever the tumor acquires angiogenesis, this will be known (strictly) before the tumor reaches metastasis.[19] Any such goal formula should implicitly be put inside an enclosing K operator to ensure that it holds in all starting states initially considered possible.

Model checking tools for epistemic CTL can be devised by combining CTL model checking with a subset based construction common in epistemic logics. An example of a logic that includes both temporal logic and epistemic component (as well as a notion of cooperation) is Alternating-time Temporal Epistemic Logic [112]. For this logic, model checking tools exist.

1.4.3 Tests in timed CHAs

Analogously to untimed CHAs, we also extend the timed CHA framework to include belief sets and tests. A *belief set* b now is not just a set of states v, but a set of timed states (v, val). A *belief state* is a tuple (v, val, b) such that $(v, \mathsf{val}) \in b$. As before, we assume some initial belief set b_0 that is used when no other belief set is given.

Before we generalize the notions of transitions and executions of a therapy we need to introduce a new relation. It addresses the following issue: With full observability, we can identify the individual delay or state transitions; however, with partial observability, a sequence of several transitions might look like just one transition to the outside observer. We denote such multi-step transitions using $\overset{\delta,C}{\dashrightarrow}$, which relates any two states that are related by any number of transitions under C taking a total time of δ.

[19] More precisely, the statement is that at any point in the future where Ang holds, M will not hold at the current or the next step until Ang is known (where Ang is the Angiogenesis hallmark and M the Metastasis hallmark).

Formally, for two timed states (v, val) and (v', val'), we have $(v, \mathsf{val}) \overset{\delta,C}{\dashrightarrow} (v', \mathsf{val}')$ if there exists a sequence

$$S = (v, \mathsf{val})(v_1, \mathsf{val}_1) \ldots (v_k, \mathsf{val}_k)(v', \mathsf{val}')$$

of state or delay transitions under C, with $\tau(S) = \delta$. (Recall that $\tau(S)$ denotes the total duration of execution S.)

Definition 13. *In timed CHAs with partial observability, there are three types of* ***transitions*** *between belief states:*

1. Delay *transitions, in symbols* $(v, \mathsf{val}, b) \xrightarrow{\delta,C} (v, \mathsf{val}', b')$*, where*
   - $(v, \mathsf{val}) \xrightarrow{\delta,C} (v, \mathsf{val}')$ *and*
   - $b' = [b]_{\overset{\delta,C}{\dashrightarrow}}$
2. State *transitions, in symbols* $(v, \mathsf{val}, b) \to (v', 0, b')$*, where*
   - $(v, \mathsf{val}) \to (v', 0)$ *and*
   - $b' = [b]_{\overset{0,C}{\dashrightarrow}}$*, that is, all state transitions under* C
3. Test *transitions, in symbols* $(v, \mathsf{val}, b) \xrightarrow{t} (v, \mathsf{val}, b')$*, where*
   - $b' = \{(v', \mathsf{val}') \in b \mid v' \in t(v)\}$.

Note that tests in this formulation only give information about the current state, and not about the current clock values. If deemed biologically plausible, this formulation can be extended appropriately.

Note also that test transitions are assumed to be instantaneous. We make this assumption because receiving the result of a test usually takes hours or days, whereas tumors usually progress on a larger time scale (months or years).

As before, a *run* of a timed CHA H with tests is a non-Zeno sequence of delay, state and test transitions.

Definition 14. *A **therapy** is a function $\pi : Runs_{\mathsf{f}}(H) \to 2^{\mathcal{D}} \cup T$. Again, a therapy is **uniform** if it only depends on the belief sets, and we only consider uniform therapies, without explicitly mentioning it. A **possible execution** of π in H is a run*

$$S = (v_0, 0, b_0)(v_1, \mathsf{val}_1, b_1)(v_2, \mathsf{val}_2, b_2) \dots$$

such that

- *for all i with delay transition $(v_i, \mathsf{val}_i, b_i) \xrightarrow{\delta, C} (v_{i+1}, \mathsf{val}_{i+1}, b_{i+1})$ and for every $0 \leq \delta' < \delta$,*

$$\pi((v_0, 0, b_0) \dots (v_i, \mathsf{val}_i, b_i)(v_i, \mathsf{val}_i + \delta' \rho(v_i, C), [b_i]_{\dashrightarrow}^{\delta', C})) = C \ ,$$

  where $\rho(v_i, C)$ denotes the partial evaluation of ρ, i.e., the function $x \mapsto \rho(v_i, C, x)$, and

- *for all i with test transition $(v_i, \mathsf{val}_i, b_i) \xrightarrow{t} (v_{i+1}, \mathsf{val}_{i+1}, b_{i+1})$,*

$$\pi((v_0, 0, b_0) \dots (v_i, \mathsf{val}_i, b_i)) = t \ .$$

Example 4 can straightforwardly be generalized to include timing.

The definition of costs is analogous to definition 9, except that tests have to be treated separately since they take no time (and would thus add no costs). The formula can straightforwardly be modified to count the costs of tests at some constant rate (still discounting the future).

Again, the notions of cost of executions, Pareto dominance, universal therapies, non-Zeno-ness and null therapies are the same or very similar to those with untimed

CHAs. Also, the result of Theorem 9 can be extended to include partial observability and tests, using a similar construction as in the proof of the theorem.

1.4.4 Therapies as conditional plans

In this section, we show how a therapy can be interpreted as a conditional plan instead of a function from runs to actions. Intuitively, a conditional therapy plan is a sequence of therapeutic actions, which branches after each test action into distinct sub-cases according to the possible observations of the test. We give the formal translation of a therapy π into a conditional plan π_c below.

Before we proceed, we note that, due to uniformity, a therapy can be regarded as a function assigning actions to sequences of belief sets (rather than executions). We write b_S for the sequence of belief sets in S. When S is clear from the context, we drop the subscript and simply write b. By $\mathrm{b} \circ b$ we denote the sequence b with belief set b appended.

Definition 15. *Given a sequence of belief sets* $\mathrm{b} = b_0 \dots b_n$*, a time* τ *and a therapy* π *we define a conditional plan* π_c *as follows:*

- *If* $\pi(\mathrm{b}) = C \in 2^{\mathcal{D}}$*, then*

$$\pi_c(\mathrm{b}, \tau, \pi) = (C, \tau); \pi_c(\mathrm{b} \circ [b_n]_{\dashrightarrow}^{\delta, C}, \tau + \delta, \pi) \ ,$$

  where δ *is the minimum value such that*

  - $\pi(\mathrm{b} \circ [b_n]_{\dashrightarrow}^{\delta, C}) \neq C$*, and*
  - $\pi(\mathrm{b} \circ [b_n]_{\dashrightarrow}^{\delta', C}) = C$ *for all* δ' *such that* $0 \leq \delta' < \delta$*.*

- *If* $\pi(\mathrm{b}) = t \in T$ *with possible observations* $o_1, \ldots, o_k$*, then*

$$\pi_c(\mathrm{b}, \tau, \pi) = (t, \tau); \texttt{case} \begin{bmatrix} o_1 : \pi_c(\mathrm{b} \circ (b_n \cap O_1), \tau, \pi) \\ \ldots \\ o_k : \pi_c(\mathrm{b} \circ (b_n \cap O_k), \tau, \pi) \end{bmatrix}$$

  where $O_i = \{(v, \mathsf{val}) \in V \times \mathbb{R}^X_{\geq 0} \mid t(v) = o_i\}$*, and the* `case` *statement has the intuitive meaning, as explained below.*

Given the initial belief set b_0*, the conditional plan that corresponds to the therapy* π *is defined as* $\pi_c(b_0, 0, \pi)$*.*

The intuition behind this translation is as follows. Since a therapy only depends on the sequence of belief sets, and the evolution of belief sets under any cocktail C is predetermined, we can compute when the therapy will change. For example, starting at the initial belief set b_0 with initial cocktail C, the therapy changes at the smallest δ such that $\pi([b_0]_{\overset{\delta,C}{\dashrightarrow}}) = C'$ for some $C' \neq C$. The new belief set at this moment is $b_1 = [b_0]_{\overset{\delta,C}{\dashrightarrow}}$, and the conditional plan up to this point is $(C, 0); (C', \delta)$. We can continue this procedure with the sequence $b_0 b_1$. When a test is performed, the next move depends on the observation o_i, which is reflected in the branching `case` statement. The execution of such a therapy plan would then continue at the branch labeled with the observation.

1.5 Tumor heterogeneity

So far we have not discussed an important property of cancer, namely, that most large tumors are *heterogeneous*. That is, they consist of several sub-populations of tumor cells [93], each with a distinct dominant phenotype [49, 68]. Tumor heterogeneity forms a large obstacle in therapy design, as it is often the cause of *therapeutic resistance*.

Namely, for any given treatment, there is likely to be a small population of cells within the tumor that is resistant to the effects of the drug. When the drug is given to a patient, even though the bulk of the tumor disappears, these cells will survive and multiply which ultimately lead to recurrence [24].

CHA models, even though presented in this chapter as models for *inter*-tumor heterogeneity (that is, different tumors progress using different paths), can also model this type of *intra*-tumor heterogeneity where different sub-clones of the same tumor are assumed to follow different progression paths. In this interpretation, the *time* it takes to make a transition does not only include the *stopping time* of acquiring a certain mutation, but rather the time it takes for a small clone of cells to grow to full tumor size. Similarly, a transition to a next state does not need to indicate that the tumor *acquires* a certain mutation or property, but rather, it can also become available once the dominant clone of the tumor has been removed e.g., by therapeutic intervention.

To control heterogeneous tumors, several possible progression paths need to be controlled at once. This can be represented within our current framework by exploiting our notion of *partial observability*, as it was just introduced. Rather than interpreting this notion as representing that the therapist does not know *the* current state of the system, it can be used to model that the therapist does can not know exactly *which sub-clones* the tumor consists of, but needs to design a therapy based on all the possibilities. The resulting therapeutic regimens will in most cases need to make use of *combination therapy*.

In our framework, there is no need to make an explicit distinction between the two types of inter-tumor versus intra-tumor genetic diversity, as they can be included in the same model, and will result in similar therapy plans. In fact, when extracting progression models from cross-sectional patient data, there is no way of knowing to which degree the observed variability results from the different types of diversity. We

will return to the problem of progression extraction in Chapter 4.

1.6 Liver and product automata

In a patient, cancer itself is not the only system of relevance. Other systems interact with the tumor's development, and especially during a therapeutic intervention, they need to be monitored. For example, the immune system and its role throughout carcinogenesis are receiving more and more attention [37], and the liver needs to be monitored to avoid damage due to excess toxicity.

In principle, other subsystems of an organism could be modeled as hybrid automata in the same way as our CHA, which could then be composed as an overall model for which therapies with goals spanning all subsystems could be generated. Here we do not provide a general framework, but we sketch a simple toxicity-based liver model that can be "attached" to a CHA. It has only one clock, modeling one type of toxicity level, and a very simple discrete dynamics governed by a sequence of thresholds. Simple as it is, this kind of model can still capture effects that are discussed in the literature, such as the dynamics of the toxicity level in the liver caused by Taxol [103], a drug used in breast cancer treatment.

Definition 16. *A **liver automaton** is a tuple $L = (W, F, w_0, \ell, \rho)$, where*

- *W is a set of states,*
- *$F \subseteq W \times W$ is a set of directed edges,*
- *$\ell : W \to \mathbb{R}$ gives the toxicity threshold for each state, and*
- *$\rho : W \times \mathcal{D} \to \mathbb{R}_{\geq 1}$ gives the toxicity factor for each pair of state and drug.*

For simplicity, we restrict attention to *linear* liver automata, i.e., each state has at most one successor. For this reason, we do not need explicit constraints on the

edges and can instead assume that a state's outgoing edge is enabled exactly when its toxicity threshold is reached.

We can then define the overall toxicity factor of a given cocktail in a given state as a function $\rho : W \times 2^{\mathcal{D}} \to \mathbb{R}$ as follows:

$$\rho(w, C) = \begin{cases} \prod_{d \in C} \rho(w, d) & \text{if } C \neq \emptyset \\ -1 & \text{if } C = \emptyset \end{cases}$$

Note that $\rho(w, \emptyset) = -1$, while for any $C \neq \emptyset$, we have $\rho(w, C) \geq 1$. That is, we assume that drugs cumulatively increase the toxicity level, and that the liver regenerates only when no drugs are given. The model can easily be extended to include some drugs that have no effect on the liver, or to allow for other interactions between cocktails.

Definition 17. *A **timed state** of a timed liver automaton $L = (W, F, w_0, \ell, \rho)$ is a tuple (w, c), where $w \in W$ is a current state and $c \in \mathbb{R}$ is a current clock value for w.*

There are three types of transitions between timed states in a liver automaton:

1. Delay *transitions, in symbols* $(w, c) \xrightarrow{\delta, C} (w, c')$, *where*

   - $\delta \in \mathbb{R}_{>0}$ *represents the (real) time delay,*
   - C *denotes the cocktail active during that time,*
   - $c' = \begin{cases} \max\{0, c + \delta\rho(w, C)\} & \text{if } w = w_0, \text{ and} \\ c + \delta\rho(w, C) & \text{otherwise} \end{cases}$
   - $-1 \leq c' \leq \ell(w)$.

2. State *transitions, in symbols* $(w, c) \to (w', 0)$, *where*

   - $c = \ell(w)$,
   - $(w, w') \in F$.

3. Regenerating *transitions, in symbols* $(w, -1) \to (w', c')$, *where*

- $c' = 0$,
- $(w', w) \in F$.

The exact thresholds for regenerating transitions can be modeled in more detail where required.

A liver automaton can be combined with a CHA using standard parallel composition methods as the one defined in [62]. Informally, states of the new model are combined states $((v, \mathsf{val}), (w, c))$ from the CHA and Liver model respectively. A delay transition between two states $((v, \mathsf{val}), (w, c))$ and $((v', \mathsf{val}'), (w', c'))$ is possible if the delay transition is possible between both (v, val) and (v', val') as well as (w, c) and (w', c') under the current therapeutic regimen. A state transition is possible if: a state (or regeneration in case no drug is being applied) transition is possible in one model, and in the other model the states stay the same; or, if both models allow for a state transition at the same time. This method can be extended to include belief sets as wel.

For the resulting product model we can formulate combined goals involving both the CHA and the liver models. To illustrate this point, consider the following simple example:

Example 5. *Let L be a liver model with two states: L_1 low toxicity, L_2 high toxicity, such that the threshold for going to L_2 is 10. Assume furthermore that the toxicity level of Avastin is 4. Thus, if we administer Avastin for a duration of more than 2.5 time units, the liver automaton will move to the high toxic state.*

A simple therapeutic goal in the combined model of L with the example CHA from Figure 1.2 might be to avoid a high level of toxicity (L_2) while avoiding reaching metastasis:

$$\mathsf{AG}(\neg M \wedge \neg L_2) \ .$$

This goal can be achieved by a therapy in which one administers Avastin in the states

prior to Ang as before, but does this as late as possible, so as not to reach a high toxicity level. This kind of control can be achieved by starting drug therapy when the clock reaches a value of (almost) 3*, such that due to the slowing down effect of Avastin in the CHA the invariant of* 5 *will be reached before the clock value reaches a transition to Ang becomes possible. At the same time toxicity levels below* 10 *(in fact, below* 8*) are maintained* [20]*. The system will be forced back to Normal, and before Avastin is administered again, the liver has time to regenerate.*

1.7 Conclusion and two other phenomena

In this chapter we established a general formalism for describing cancer progression. Our goal was to design a conceptually clear framework based on realistic biological foundations. As a case study, we have used this model to describe cancer hallmarks and their dynamics. Before turning the problem of CHA *control* in the next chapter, we conclude by mentioning two model phenomena, not yet discussed so far, that can be modeled using our framework as i.

More general clocks: Thus far, we have referred to the clocks in CHAs as measuring *time*. However, they could be measuring different properties like *tumor size*, the number of *stem* and *differentiated* cells, *motility* or *spatial properties*. For example, in the case of tumor size, the growth rate of the tumor may depend on the current discrete states of the progression and drugs can influence this rate. With this model one can describe tumor growth dynamics as described in [105], by introducing two clocks: one measuring the number of stem cells and the other the number of differentiated cells. The various mutations can be modeled as transitions to a next state with different

[20]Note that the therapy defined in example 3 does not satisfy this goal, as high toxicity levels will be reached.

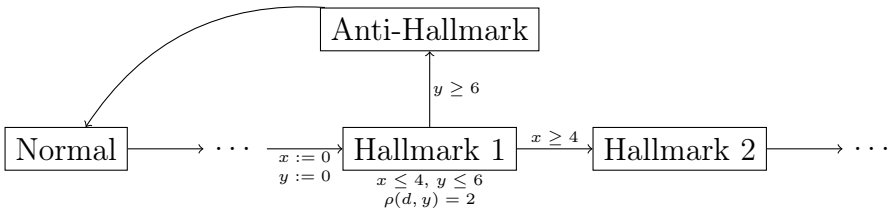

Figure 1.4: Illustrating how to model an anti-hallmark using two clocks x and y and a drug d that speeds up clock y at Hallmark 1 by a factor of 2.

growth dynamics depending on the mutations already acquired[21].

Anti-hallmarks Instead of trying to *slow down* cancer progression, there has recently been growing interest in approaches to *speed up* the process to a degree which will make the tumor nonviable and "push it over the edge" towards collapse (see e.g., [82]). We refer to such nonviable states as *anti-hallmarks*. They can be modeled by putting constraints on the transitions leading to them that will never be satisfied, unless a drug is given which speeds up a certain clock. For example, consider the CHA in Figure 1.4. At Hallmark 1, without interference (both clocks increase with rate 1), the transition to Hallmark 2 will be taken after 4 time units. A drug that speeds up clock y by a factor of 2 will instead push the tumor to the Anti-Hallmark state, if given starting at most 1 time unit after entering Hallmark 1.

[21]One should note, however, that even though the progression can be described, the controller synthesis results from the next chapter assume that clock behavior is linear at each state, and do not carry over to a system in which clock dynamics is described by differential equations.

Chapter 2

Controller Synthesis and Automatic Therapy Design

Given the complexity of (timed) cancer progression and the influence of various drugs, the task of finding near-optimal therapy plans is (soon to be) beyond manual planning, and automated computational tools are very desirable. In this chapter we turn to the question if, and how, algorithms can be designed that generate successful therapies automatically.

To answer this question we turn to the area of engineering called *control theory*. In control theory, the objective is to manipulate a system in such a way that the controlled system satisfies a certain desired specification. With the ultimate goal of controlling CHAs (introduced in the previous chapter) in mind, an overview of some of the relevant concepts and results are presented in Sections 2.1 and 2.2. In this short overview, we discuss the problem of control under two types of uncertainty: uncertainty of the outcome of an action (non-determinism), and uncertainty of the state of the system (partial observability) – both properties are important building blocks of our CHA framework. We focus on a decision- or game-theoretic formulation of the control problem as a two-player game in which the controller is represented as

a strategy for one of the players. A strategy is winning if the goal is guaranteed to be satisfied for every possible play of the game that conforms to that strategy. For several formal frameworks, including finite, timed and finally hybrid automata, we study the problem of generating winning strategies for various goals like reachability and safety, as well as more general temporally extended goals, specified using temporal logic such as CTL. Even though many different notational standards exist in the literature, we aim to present the various framework in a unified way. The most important definitions and results are presented in the body of this chapter, while more details can be found in the appendix, or in the references provided. While the tools and results in this short overview (Sections 2.1 and 2.2) provide our basis for designing CHA control algorithms, they are presented in an independent fashion, and can thus be read separately, or may be skipped all together.

In Section 2.3 we turn our focus to CHAs specifically. Untimed CHAs are a special kind of discrete automata for which game theoretic controller synthesis algorithms exist that can be applied to automatically design therapy-plans. Timed CHAs, on the other hand, are a special class of hybrid automata, and for this type of frameworks, control is not as straightforward. Unfortunately, in the general case, even simple verification and control problems like reachability and safety are undecidable for hybrid systems [66]. However, several decidable subclasses of hybrid automata exist for which algorithms have been devised, and we will see in Section 2.3, how existing results can be extended to CHAs. Our main result is Theorem 9 which states that in a certain restricted setting (where the players can only move at discrete moments in time), CHA control for CTL goals is EXPTIME complete. This result can also be extended to include partial observability (Theorem 10).

2.1 Acting and control under uncertainty

Uncertainty interferes with decision making in two ways [87]:

1. Uncertainty of the **outcome of an action**. Because of uncertainties, it may not be known what will happen when certain actions are performed, and future states are thus not fully predictable. This kind of uncertainty is sometimes called *predictability uncertainty* or simply *non-determinism*.

2. Uncertainty of the **state of the system**. As a result of uncertainties, the current state of the system may not be not known. Information about the state can be obtained from initial conditions, memory of the history of play, or observations. This kind of uncertainty is sometimes called *sensing uncertainty, partial observability* or *imperfect information*.

These two types of uncertainty are largely independent and can be studied separately. Just as we have done when introducing CHAs, in this overview, we will start by studying ways to model and control the first type of uncertainty and will subsequently incorporate the second type.

Non-determinism can be modeled by introducing a decision maker called *Nature*, who influences the outcome of an action. The task for the decision maker is to make a good, or the best, decision, despite the interfering effect of Nature's actions.

Nature's decision making process can be characterized using either qualitative non-deterministic (henceforth: non-deterministic) models or probabilistic models. Models that include probabilities are often easier to deal with since expected utility, or expected outcome can be used to guide the decision making process. A *best* action can be determined as the action that maximizes expectation (expected payoff, expected chance of winning, etc). However, it is not always reasonable to assume that the

agent has enough information about the situation to be able to adequately assign probabilities to Nature's actions, in which case the use of non-deterministic models is required. This is one of the main reasons why we have not included probabilities in our CHA framework.

2.1.1 Control of discrete automata: a decision theoretic perspective

Both fields of computer science and control theory study these discrete event systems. Traditionally, computer science has focused on system verification and the question 'What will the system do?', while control theory centers around synthesis questions 'How can we make the system behave the way we want?' [7]. In the latter case, the objective is to manipulate a system ("plant") in such a way that the controlled system ("plant + controller") satisfies a certain desired specification. The interaction between a controller and a plant can be seen as an antagonistic infinite game between two players (the controller and Nature) [8]. A *strategy* (or a *therapy* in case of CHAs) for a given game is a set of rules that tells the controller what action to take in any possible situation of the game. A strategy is *winning* if no matter what actions Nature chooses during the game, the controller is guaranteed to win. Properties that can be used to evaluate the play include utilities or costs of outcomes, as well as more general properties about the states visited: these can be specific properties like non-reachability of bad states (so-called *safety properties*) as well as more general properties, expressed in temporal logics such as CTL. The problem of designing a controller for a plant now becomes that of finding a winning strategy for the controller. For untimed systems this problem is relatively straightforward and has been studied for a variety of winning conditions. In the following, we discuss the simplest of winning conditions, of *safety* and *reachability*, as well as as well as more general goals specified using temporal logic. We refer the reader to [108] or [91] for more general treatments

of winning conditions.

In Appendix A.1, we describe extensions in which outcomes of actions have associated numerical costs that need to be minimized while deciding on a strategy.

Discrete games

Assume two players 1 (the controller) and 2 (Nature). The players play on a finite graph with states Q. At each moment in the game, both players choose an action available to them, and the game progresses along an edge consistent with the chosen actions. Formally, a two-player automaton-game is defined as follows (along the lines of [8] and [7]):

Definition 18 (Game Automaton)**.** *A* Game Automaton A *is a tuple* $\langle Q, Q_0, \Sigma^1, \Sigma^2, E\rangle$*, where*

- Q *is a finite set of states,*
- $Q_0 \subseteq Q$ *a set of initial states,*
- Σ^i *a set of actions for player* $i \in \{1, 2\}$*,*
- $E \subseteq Q \times \Sigma^1 \times \Sigma^2 \times Q$*, is a set of transitions.*

With $T^i(q)$ denote the set of actions from Σ^i that are enabled at q is denoted. That is, $T^1(q) = \{a \in \Sigma^1 \mid (q, a, b, q') \in E \text{ for some } b\}$. It is commonly assumed that for each $q \in Q$ and $i \in \{1, 2\}$, $T^i(q) \neq \emptyset$ and that whenever $(q, a) \in T^1$ and $(q, b) \in T^2$ there exists a transition $(q, a, b, q') \in E$ for some $q' \in Q$. That is, there can be no deadlock. We write $q \xrightarrow{(a,b)} q'$ if $(q, a, b, q') \in E$. Intuitively, this notation means that the system can move from q to q' if the agent plays a and Nature plays b. A *run* of A is a (possibly infinite) sequence ζ of the form

$$\zeta = q_0 \xrightarrow{(a_0,b_0)} q_1 \xrightarrow{(a_1,b_1)} q_2 \dots$$

We let $L(A)$ denote the set of runs of A starting at a state $q_0 \in Q_0$, and $L(A)_f \subseteq L(A)$ the set of finite runs. [1]

Definition 19 (Strategy)**.** *A strategy π_i for player i is a function $\pi_i : Q \to \Sigma^i$, such that $\pi_i(q) \in T^i(q)$ for each player i. Given a strategy π for player 1, a run $\zeta = q_0 \xrightarrow{(a_0,b_0)} q_1 \xrightarrow{(a_1,b_1)} q_2 \ldots$ is said to* conform *π if for every j, $\pi(q_j) = a_j$. We let $L_\pi(A) \subseteq L(A)$ denote the set of runs of A that conform π.*

A strategy is sometimes referred to as a *controller* in control theory. In the planning literature, a strategy is referred to as a *feedback policy*, a *conditional plan* or simply a *plan*. And, in the context of CHAs a strategy is referred to as a *therapy*.

Note that strategies in the above definition are assumed to be *memory-free*. That is, they do not depend on the entire run of the game but only on the current state. If we allow the system to possess memory, as we do in CHAs, a controller can be defined as $\pi_i : L(A)_f \to \Sigma^i$. It should be noted however, that if winning conditions are simple and only depend on which states are visited during a run, like safety or reachability, allowing for memory does not improve winning conditions. More complicated winning conditions, like those specified using temporal logic, sometimes require more information about the history of the game. For safety and reachability games discussed next, considering only memory-less strategies is thus appropriate.

Control of safety and reachability games

As mentioned before, in a safety game, the goal of the controller is to keep the game outside a set of 'bad states' B. In a reachability game, the goal of the controller is to eventually reach a 'good state' from a set F. The winning states are all the states from which the controller is able to accomplish this. Thus, the winning states are those such that no matter what actions Nature chooses, the controller is able to keep

[1] In CHAs we use the less general notiation $Runs_f(H)$ to denote the set of runs of a CHA H.

the game from going outside the set of safe states $F = \bar{B}$ for safety games, and to eventually reach one of the good states for reachability games.

Now, how do we determine if a set of states F is safe? The basic idea is to iteratively search the space for a set of winning states F^*. The iterative search starts with $F_0 = F$, the set of safe states, and at each round i the set of states from which the controller cannot force the game to stay in F_i are deleted. This procedure converges to F^*. If $Q_0 \subseteq F^*$, the controller has a winning strategy, which can be described as 'never move out of the set of safe states'. For reachability the argument is very similar. Instead of iteratively deleting states that may lead to a state *outside* of F, now states are iteratively added that can lead to new states *inside* F. The complexity of these control algorithms is linear in the number of states.

The formal details of the controller synthesis problem for these games, as well as well as extensions that include costs and cost-optimization to the framework are provided in Appendix A.1.

The probabilistic counterpart of game automata are Markov Decision Processes (MDPs). An MDP is a discrete time stochastic automaton in which a next state is reached with a certain probability depending on the actions chosen. Just like in the game automaton, in MDPs, the probability that a next state is reached only depends on the current state and action, and not on states previously visited. In the context of probabilistic models, this property of being memory-less is sometimes referred to as being 'Markov'.

More general goals: temporal logic

The control problem for reachability and safety goals can be generalized to include more general goals specified using temporal logic. The two most commonly used temporal logics are *linear temporal logic* (LTL) and *computation tree logic* (CTL) [31].

In LTL, a basic propositional language is extended with the temporal operators X (Next), F (Future), G (Global) and U (Until). Intuitively the formula $X\phi$ can be read as 'in the next state ϕ will hold', $F\phi$ as 'at some future state ϕ will hold', $G\phi$ as 'at every future state ϕ will hold', and $\psi U\phi$ as 'ψ will hold until ϕ becomes true'. LTL satisfaction presupposes that there is only one possible future path. That is, future time is assumed to be linear. In the case of plays on game automata, however, this is not necessarily the case. CTL was already introduced in Chapter 1, and is a branching time temporal logic that has two additional operators A (for all paths) and E (there exists a path), with the formula $AF\phi$ meaning 'At every possible path, at some point in the future ϕ becomes true'.

CTL^* is an extension of CTL that allows for arbitrary nesting of temporal operators.

In the case of temporally extended goals, controllability no longer ensures the existence of memory free strategies, but rather (parts of) the history of the game need to be taken into account when deciding on a suitable move [108]. This complicates the controller synthesis problem, but does not make it noncomputable. In fact, much work has been done on automatically generating controllers for game automata with LTL and CTL specifications. In [79], for example, Kumar et al. develop an algorithm using CTL specifications, that runs in exponential time in the size of the formula to be satisfied. This complexity bound is sharp, as the problem is shown to be EXPTIME complete:

Theorem 1 (Jiang and Kumar [79])**.** *The controller synthesis problem for game automata with CTL goals is EXPTIME complete in the size of the formula.*

Adding partial observability

Up until now we have assumed that the current state of the system is always known. But what if this is not the case? In this section we introduce the second type of uncertainty: *sensing uncertainty* or *partial observability*. In a case where the state

of the environment is not known, there are still several ways the controller is able to learn information about the current state [87]. Most importantly, *observations* provide measurements regarding the current state of the system. Also, information about the current state of the system can be deduced from *initial conditions* like the set of initial states in an automaton, or from *actions* that were already executed. However, despite all the information available, it may not be possible to deduce the exact state of the system. Luckily, many problems can be solved without requiring that the exact state is ever known. For this purpose, the controller synthesis problem can be described in terms of a *sensor*.

A sensor gives information about the state and consists of two components: 1) an *observation space*, which is the set of possible observations from the sensor, and 2) a *sensor mapping*, which maps situations to the set of observations [87].

Sensor mappings can be *state based*, depending only on the current state, *history based*, depending on the sequence of previously visited states, or *state-Nature based*, depending on both the state and Natures *sensing actions*. Moreover, they can be assumed to be non-deterministic or probabilistic. That is, depending on the current state, history or Nature-state, the observation may not be fully determined.

In the following, we assume that the sensor mapping is state-based and non-deterministic. But these definitions can easily be adapted to incorporate other types of sensor mappings. One way of defining a partially observable game automaton is as follows:

Definition 20 (Partially observable game automaton)**.** *A* Game Automaton A *is a tuple*
$\langle Q, Q_0, \Sigma^1, \Sigma^2, E, O, S\rangle$*, where*

- Q *is a finite set of states,*
- $Q_0 \subseteq Q$ *a set of initial states,*

- Σ^i *a set of actions for player* $i \in \{1, 2\}$,
- $E \subseteq Q \times \Sigma^1 \times \Sigma^2 \times Q$, *is a set of transitions,*
- O *is a set of observations,*
- $S : Q \to 2^O$ *is a sensor mapping, associating to each state a set of possible observations.*

The simplest memory-free strategy in a game automaton with partial observability is a mapping from observations to actions, $\pi : O \to \Sigma$. However, because in this case the agent is likely to obtain information about the current state of the system by keeping track of previous actions and observations, it is more likely that a strategy is a function assigning actions to more general *contexts* [2], $\pi : C \to \Sigma$. These contexts could simply be finite histories of observations, in which case $\pi : O^n \to \Sigma$, but can possibly include previous actions by Nature or sets of states that are considered possible, i.e. *belief sets*.

Algorithms for the control problem of a partially observable automaton usually proceed by translating the partially observable model to one with full observability where states are subsets of the original state space representing possible knowledge states of the agent. An example of such a method can be found in [16], where Bertoli studies the controller synthesis problem (formulated as a planning problem) under partial observability with CTL goals, and proposes a somewhat more sophisticated algorithm. The algorithm based on a forward-chaining approach, that incrementally progresses so-called *belief-desires* structures. These structures relate sets of states, or beliefs about the current state compatible with past actions and observations to sub-goals that have to be achieved from such states. Once a new state is reached the beliefs and the associated goals are updated. The algorithm assumes a single agent

[2] in fact, in [104] Reif shows that even safety games of partial information cannot always be won using memory-less strategies

structure, that is, Nature makes no explicit moves. However, their transition relation is assumed to be non-deterministic, and could be formulated as a two-player game.

The probabilistic counterpart of a game automaton with partial observability is a Partially Observable Markov Decision Processes (POMDP). A POMDP is an extension of a markov decision process in which the agent cannot directly observe the actual state of the system. Instead, it must maintain a probability distribution over the set of possible states, based on a set of observations, the probabilities of these observations, and the underlying MDP.

2.2 Control of timed systems

Thus far, we discussed simple frameworks in which all transitions in the automaton take one 'unit' of time. For many applications however, the exact timing and duration of transitions are crucial to the analysis of the system. In this second part of this short survey, we introduce extensions of discrete (game) automata to timed and hybrid (game) automata, and study their corresponding control problems.

2.2.1 Timed automata

Modeling real time

First we extend the theory of discrete games to timed games. Timed games are based on so-called timed automata that have been introduced by Alur and Dill in their seminal article [3]. Timed automata provide the theoretical foundation for the study of the behavior of real time systems.

A timed automaton is defined as a finite state transition system extended with a finite set of real-valued variables called clocks. Constraints on the clock values are used to restrict the behavior of the system.

Timed game automata

Definition 21 (Clock constraints and clock interpretations)**.** *Given a set X of clocks, the set $\Phi(X)$ is the set of* clock constraints *ϕ defined by*

$$\phi := x \sim c \mid x - y \sim c \mid \phi_1 \wedge \phi_2$$

where x, y are clocks in X, c is a constant in $\mathbb{Z}$ and $\sim \in \{\leq, <, =, >, \geq\}$. A clock interpretation *or* valuation *v is a mapping from X to the reals $\mathbb{R}$.*

For a $\delta \in \mathbb{R}$, $v + \delta$ denotes the clock interpretation that maps every clock x to the value $v(x) + \delta$. For a set $Y \subseteq X$, the function $v[Y := 0]$ assigns 0 to all clocks in Y and agrees with v on the clocks $x \notin Y$.

A timed game automaton is defined as follows (along the lines of [91]).

Assume a finite set of clocks X.

Definition 22 (Timed Game Automaton)**.** *A timed game automaton A is a tuple $\langle Q, V, Q_0, \Sigma^1, \Sigma^2, E, I \rangle$, where*

- *Q is a finite set of states,*
- *$V = \mathbb{R}^X$, the set of clock valuations,*
- *$Q_0 \subseteq Q$ a set of initial states,*
- *Σ^i a set of actions for player $i \in \{1, 2\}$,*
- *$E \subseteq Q \times \Sigma^1 \cup \Sigma^2 \times \Phi(X) \times 2^X \times Q$, is a set of transitions, each of the form (q, c, ϕ, Y, q') where $c \in \Sigma^1 \cup \Sigma^2$, $\phi \in \Phi(X)$ is the clock constraint on the edge and $Y \subseteq X$ is the set of clocks to be reset by the transition.*
- *$I : Q \to \Phi(X)$, associates with each state its invariant.*

As before, we write $q \xrightarrow{c,\phi,Y} q'$ if $(q, c, \phi, Y, q') \in E$. Note that contrary to the discrete case, edges are labeled with only one action. The reason for this modification is that it is no longer necessary for two agents two play at exactly the same time. It may be possible that two edges between states exist indicating that the new state can be reached either by a controlled action by the agent or an uncontrollable action by Nature.

The semantics of a timed automaton is defined as an infinite state transition system where a timed state is a pair (q, v) of the current state of the automaton q and the current clock valuation v.

Just as is the case with timed CHAs, there are two types of transitions between two states: *delay transitions* where just time passes and *state transitions* when the actual state of the automaton changes. Formally, these two transitions are defined as follows.

Definition 23 (Timed Automata: semantics)**.** *Two types of transitions exist.*

- *(Delay transition)* $(q, v) \xrightarrow{\delta} (q, v + \delta)$*, with* $\delta \geq 0$ *if for all* $0 \leq \delta' \leq \delta$*,* $v + \delta'$ *satisfies the invariant* $I(q)$*.*
- *(State transition)* $(q, v) \xrightarrow{c} (q', v[Y := 0])$ *if there is an edge* $q \xrightarrow{\phi,c,Y} q'$ *such that* v *satisfies* ϕ *and* $v[Y := 0]$ *satisfies* $I(q')$*.*

A run *is a (possibly infinite) sequence of alternating Delay and State transitions. As before,* $L(A)$ *denotes the set of runs of* A *starting at a state* (q_0, v_0) *with* $q_0 \in Q_0$ *and* $v_0(x) = 0$ *for all* $x \in X$*.*

With $T^i(q, v)$ we denote the set of actions from Σ^i that are enabled at (q, v). That is, $T^1(q, v) = \{a \in \Sigma^1 \mid (q, v) \xrightarrow{a} (q', v[Y := 0]) \text{ for some } q'\}$.

Intuitively, the system can stay in a state q as long as the local state invariant $I(q)$ remains satisfied. A transition to a next state using action c can be taken when

the clock constraint, or guard, ϕ on the edge is satisfied. The transition resets all the clocks on the edge to 0, while the values of other clocks remain the same. It is now up to the controller not only to choose an appropriate action at each state, but also the exact time at which this action is best to be preformed. While making this choice, the controller must remain cognizant of the fact that at any time in the interval it chooses to delay its next action, Nature is free to make any available moves.

Control of timed safety and reachability games

The backward search algorithm for safety and reachability is very similar to that of the untimed case, except that now, not only the possible consequences of the agent's actions are important to keep track of, but also the moments when the agent is not acting.

To compute the set of winning states for safety and reachability we can apply the same fixed point algorithm as for the discrete case. However, because we are now iterating over an infinite set, it remains to be shown that the algorithm actually converges. In the discrete case, this was immediate since the iteration was over a finite domain. In the timed case, so-called *regions* and *zones* guarantee convergence. Regions and zones provide a finite partition of the state space such that within a given region, the behavior of the system is indistinguishable. Stated differently, the states within a given region are *bisimilar*. Using this bisimilar graph, there are actually a finite number of states that the algorithm iterates over. For a more detailed description of control of timed automata, including regions and zones, as well as extensions involving optimizing time and cost, see Appendix A.2.

In [26] an efficient algorithm for solving the controller reachability and safety problem for timed games has been proposed that makes use of the zone graph. Based on the on-the-fly algorithm proposed in [88] for linear-time model-checking of finite-state systems, this algorithm combines the backward propagation of winning states

with simultaneous *forward computation* of states to explore. The main idea underlying the algorithm is the same as for the untimed case: to keep the game in a set of 'safe' states.

The reachability and safety control problems in timed game automata have been shown to be EXPTIME complete [65]:

Theorem 2. *The controller synthesis problem for timed game automata with safety and reachability goals is EXPTIME complete in the size of the game.*

More general goals and control problems

The control problem of timed games (without weights) has also been studied for specifications given as temporal formulas. For LTL goals the problem is 2EXPTIME complete, for CTL goals EXPTIME complete [47], and for MTL - a timed extension of LTL - the control problem is only shown to be decidable [21]:

Theorem 3 (Faella et al. [47])**.** *The controller synthesis problem for timed game automata with LTL (CTL) goals is 2EXPTIME (EXPTIME) complete in the size of the formula.*

Adding partial observability

Partial observability of the state space has also been studied in the context of timed automata. Contrary to discrete event systems, where adding partial observability does not affect decidability of the control problem, timed control under partial observability is generally undecidable (while the analogous problem under complete observability is decidable) [20]:

Theorem 4 (Bouyer et al. [20])**.** *Timed control of timed game automata with partial observability is undecidable for a general class of goals (including safety and reachability).*

However, in the same paper it is shown that fixing the resources of the controller (i.e. a maximum number of clocks and maximum allowed constants in guards) restores decidability.

Cassez et al. have developed efficient on-the-fly algorithms for timed games with partial information and reachability goals using zone-based approaches [27, 25, 20]. In their model, sensors map timed states (that is, states of the form (q, v)) to a finite set of observations. Strategies are assumed to be observation based and *stuttering free*. That is, the controller decides based on a sequence of observations corresponding to a run, such that successive identical observations are collapsed to one. In this case, when an agent decides to play a controllable or a delay an action, he cannot play until the observation changes. That is, once the controller makes a move, he cannot make another move until the system reaches a new observation state. In the meantime, Nature is allowed to make any move to influence the behavior of the system.

Similar in spirit to subset-based construction for the untimed case, Cassez et al. solve the control problem by reducing it to a game of full observability. Their reduction makes use of a *knowledge based subset construction* introduced in [109, 30]. States in the reduction are sets of symbolic states that represent the knowledge of the controller.

The reduced graph corresponds to a timed game and efficient algorithms can be used to solve the reachability problem.

Note that this algorithm assumes that the controller cannot gain any information about the current state by keeping track of the time himself, nor by keeping track of the history of play.

A tool support for these and other algorithms is available in Uppaal-Tiga (see e.g. [12])

2.2.2 Hybrid automata

Hybrid automata extend timed automata to allow for non-synchronous continuous evolution. More precisely, while in timed automata clocks increase synchronously at the same rate, clocks in so-called hybrid automata can run at different rates, which can change independently with the transition to another state. Thus, hybrid automata combine the discrete dynamics of a finite automata with the continuous dynamics of a dynamical system. Just like in timed games, when playing in a hybrid game, the controller may choose an action that results in a discrete move in the automaton. The controller is not able to influence the rate of the clocks.

In hybrid games, even simple verification and control problems like reachability are undecidable [66]:

Theorem 5 (Henzinger et al. [66])**.** *The verification of reachability of hybrid automata is undecidable.*

From this theorem it follows that the controller synthesis problem is undecidable as well. However, several decidable subclasses of hybrid automata exist for which algorithms have been devised. Here we will focus on rectangular hybrid automata and discretized hybrid automata.

Rectangular hybrid game automata

An important class of hybrid game automata is the class of *rectangular* hybrid automata. A rectangular automaton is an automaton in which the clock constraint on each edge is a rectangular region of continuous states, and the clock speed at each state is bounded from below and above.

Rectangular hybrid systems are important for various reasons [67]. First, They form a natural system in between timed and hybrid systems. Second, they can approximate with arbitrary precision the behavior of full hybrid games, as long as

all clock rate function satisfy a strong form of uniform continuity called Lipschitz continuity [64]. Third, they form a most general class of hybrid automata for which even the reachability model checking problem is decidable [66, 67]. Since the controller synthesis problem for a class of hybrid systems is at least as hard as model checking for the same class, it follows from this that the control problem for more general hybrid games cannot be decidable.

Formally, a *rectangle* for the set X with dimension n is a subset of $\mathbb{R}^n$ that is the cartesian product of n possibly unbounded intervals, all of whose finite endpoints are integers. We let R^X be the set of all rectangles for X.

Remark 1. *Note that rectangles for a set X are very much like zones. In fact, they are restricted type of 'rectangular' zones of X: they do not allow for comparing the values of different clocks e.g $x - y < 10$.*

A rectangular hybrid game is defined as follows (along the lines of [67]).

Definition 24 (Rectangular Hybrid Game Automaton)**.** *A timed game automaton A is a tuple*
$\langle Q, V, Q_0, \Sigma^1, \Sigma^2, E, I, flow \rangle$, where

- *Q is a finite set of states,*
- *$V = \mathbb{R}^X$, the set of clock valuations,*
- *$Q_0 \subseteq Q$ a set of initial states,*
- *Σ^i a set of actions for player $i \in \{1, 2\}$,*
- *$E \subseteq Q \times \Sigma^1 \cup \Sigma^2 \times R^X \times 2^X \times R^X \times Q$, is a set of transitions, each of the form $(q, c, \phi, Y, \psi, q')$ where $c \in \Sigma^1 \cup \Sigma^2$, ϕ is the clock constraint on the edge, Y is the set of clocks whose values may change when the discrete transition takes place, and ψ is the rectangle restricting the new state of the variables at the arriving state.*

- $I : Q \to \Phi(X)$, *associates with each state its invariant.*
- $flow : Q \to R^X$, *maps to each state a* bounded *rectangle that constraints the clock speeds at this state.*

Given a rectangle ψ, the interval constraint for x imposed by ψ is denoted by $\psi(x)$.

An important requirement on rectangular hybrid games is that of *initialization* or *constant reset*. Initialization is the property that for every edge $(q, c, \phi, Y, \psi, q')$ if $flow(q)_k \neq flow(q')_k$ then $k \in Y$. That is, whenever the speed of a clock changes, the value of the variable is reinitialized. This property cannot be relaxed, as it would make the control problem undecidable [66]:

Theorem 6 (Follows from Henzinger et al. [66])**.** *The reachability controller synthesis problem for hybdrid games* not satisfying *initialization is undecidable.*

Just like in the timed game, if the controller decides to make a move $a \in \Sigma^1$ while at a state (q, v), the game will evolve to a next state (q', v') along an available edge $(q, c, \phi, Y, \psi, q')$. The new continuous state will be as follows. For each clock x, such that $x \in Y$, the value of x is non-deterministically assigned to a new value in the interval $\psi(x)$. For each $x \notin Y$, $v'(x) = v(x)$, and will move at a speed non-deterministically determined to be in the interval $flow(q')(x)$.

In [67],Henzinger et al. show that the control problem with LTL specifications is decidable:

Theorem 7 (Henzinger et al. [67])**.** *The controller synthesis problem of rectangular hybrid automata with LTL goals is EXPTIME-complete in the size of the game, and 2EXPTIME-complete in the size of the formula.*

The decidability result hinges on the fact that a rectangular hybrid automaton can be translated into a *singular game automaton*. A singular game automaton is a restricted versions of a rectangular game, in which the rate of the clock is unique at

each state, and after each transition the new value of the clock is uniquely determined. For this class of games, which is very similar to the class of timed games[3], the same useful properties of zones that guaranteed decidability in the timed case, now guarantee decidability in the hybrid case.

Discretizing the control problem

The above result is very nice but depends heavily on a set of restricting assumptions on the behavior of the system. If any of the assumptions like *initialization* are relaxed, decidability cannot be retained [66]. The simplest way around the undecidability of the hybrid automata control problem is to allow for control moves only at integer times. Henzinger and Kopke [65] give an exponential-time algorithm for discrete-time safety control of rectangular hybrid automata with bounded and non-decreasing variables. Their algorithm is based on a finite reduction to a bisimilar model, and does not assume initialization of the automata.

Theorem 8 (Henzinger and Kopke [65])**.** *The discrete time safety controller synthesis problem for rectangular hyrbid automata (without initialization) is decidable.*

However, the algorithm as presented in [65] only applies to hybrid automata and not to hybrid games. That is, it assumes that all discrete moves are made by the controller and not by Nature.

This result has been extended by De Wulf et al. by using a fixed-point theory to include partial observability of the state in [38]. Just as in the case of timed automata, observations are associated with timed states (in fact, they are unions of timed states) and become available automatically when the system enters a state of that observation.

[3]the main difference between the two being that in the case of timed games the clock rate is always 1, while in singular games this can be any number.

Other decidable classes of hybrid automata

Another decidable subclass of hybrid automata are so-called *O-minimal hybrid automata*. These are hybrid systems with a very rich continuous dynamics (polynomial and exponential functions can be used to describe clock behavior), but have limited discrete behavior (all variables are reset after each discrete transition). For safety and reachability goals the control problem of o-minimal hybrid systems is decidable [17].

Weighted timed automata as discussed in Section A.2, form an intermediate class between timed automata and hybrid automata for which decidability results are retained [23]. In the case of weighted timed automata, the undecidability of hybrid automata is avoided because information about costs cannot be used as constraints on the edges. Rather, costs are available to the observer, but do not influence the discrete behavior of the system.

Algorithms to control timed and hybrid games have been applied to many real-life systems. Examples include synthesis of online schedulers [1], synthesis of climate control for pig stables [77], automatic synthesis of robust and near-optimal controllers for industrial hydraulic pumps [28], and conflict resolution for air traffic control [110]. In the next section we will see whether/how CHAs can be controlled using hybrid games as well.

2.3 Automatic therapy design for CHAs

Untimed CHAs are a special kind of discrete automata for which controller synthesis algorithms exist and can be applied to automatically design therapy-plans (see e.g. [91] for control using safety goals and [79] for an algorithm that uses CTL specifications).

In this section we investigate how automatic therapy design can be solved for timed CHAs using game theoretical notions, by extending existing controller synthesis results for hybrid systems. The idea to use game theoretic notions to describe cancer

progression and to automatically design therapeutic regimens is not a new one, and has for example been studied in [99].

From timed CHAs to rectangular hybrid game automata Timed CHAs bear a striking resemblance to rectangular hybrid automata, and it is thus worth exploring whether some of the controller synthesis results and algorithms can be applied to CHA models as well. Unfortunately, existing decidability results do not carry over directly because of some important differences between CHAs and (rectangular) hybrid automata.

First, in the hybrid automata literature, the rates of the clocks are generally assumed to be constant at any given state[4] and what is controllable are (some of) the transitions between states. In the CHA framework, in contrast, the rates of the clocks are what can be affected by control actions (drugs), while the transitions (tumor progression) cannot be directly manipulated. However, this difference is mainly conceptual as a timed CHA can be translated to a hybrid automaton as follows:

Given a set of drugs $\mathcal{D}$ and a CHA H with states V, we construct a hybrid automaton RH in the following way: For each state $v \in V$ and each cocktail $C \in 2^{\mathcal{D}}$, RH contains a state (v, C) with the same clock invariants as v. For any edge between two states $v, v' \in V$, RH contains an uncontrollable edge between (v, C) and (v', C), for each cocktail C, with the same clock constraints and resets as on the CHA edge. In addition to the uncontrollable edges, there are controllable directed edges from (v, C) to (v, C') for each v, C and C'. These edges represent changes of therapies, and have no clock constraints or resets. At a state (v, C), the rate of each clock $x \in X$ is fixed, given by $\rho(v, C, x)$. This translation yields an automaton of size exponential in the number of drugs, but linear in the number of CHA states. We can model tumor progression and therapy as a *game* in which the therapist chooses therapeutic regimens

[4]One exception occurs in the context of so-called *differential games* [90], but their theory has not been well developed.

and the cancer chooses which state to progress to next. For this purpose, we specify that the controller is only allowed to make moves that include a change of therapy: from C to C' at state v by moving from (v, C) to (v, C'), and cancer is only allowed to pick an accessible new CHA state from the available $((v, C)(v', C))$ transition. The result is a *rectangular hybrid game automaton* with two players: nature and controller (the therapist).

For a detailed definition of the translation as well as correctness arguments, we refer to the appendix Section A.3.

To illustrate this translation, consider our example timed CHA of Figure 1.2. Translating this CHA into a hybrid automaton using the above construction, would result in two copies of the original CHA, with two copies of each state from the original automaton: one in which the VEGF inhibitor drug Avastin is administered, and one in which it is not. For example, we have a state (Ang, $\emptyset$), and a state (Ang, $\{Avastin\}$). Within each copy of the CHA (drug-controlled versus non-drug-controlled) the original transitions and timing constraints apply, and the rate of the clocks is now fixed: 1 in the no-drug states of the form (Hallmark, $\emptyset$), 0.5 in the Avastin-states (Hallmark, $\{Avastin\}$). The original transitions of cancer progression can not be influenced by the therapist and are controlled by nature. But, in addition, there exists a bi-directional edge connecting all states (Hallmark, $\emptyset$) and (Hallmark, $\{Avastin\}$), reflecting stopping, or starting drug administration. These transitions are controlled by the therapist. Contrary to uncontrolled CHA edges, which reset the clocks, when transitions of this type are taken, the clock values (measuring the degree of cancer progression at a state) are kept intact, and only their rate changes.

In the case of only one drug, the result is an automaton twice the size of the original one. With n drugs, however, the resulting automaton will be 2^n times the sizes of the original CHA.

A timed CHA can thus be reduced to a rectangular hybrid game automaton. Given

this reduction, it should be noted that a CHA could, in principle, be *defined* as a special sub-class of hybrid game automata, using a state of each possible pair of progression state and cocktail, rather than having drugs influence the rate of progression directly. We chose to represent CHAs using the current framework, because we believe that clock control is a very natural way of describing how therapy affects progression, and provides and powerful intuition. Moreover, the current CHA formulation is concise and (we believe) easier to understand.

Initialization and discretized control A problem with the above translation is that the translated CHA does not satisfy an important property that the positive results from [67] rely on: *initialization* or *constant reset*. Initialization states that whenever the speed of a clock changes after a transition, the value of the variable is reinitialized to a fixed value (or a value in a fixed interval). This property cannot be relaxed without making the control problem undecidable [66]. Thus, the results of Henzinger et al. do not apply.

To see why initialization is not satisfied, recall that the clock values (indicating progression time) are kept along controllable (change of cocktail) transitions while changing the rates of the clock, for example, when starting Avastin administration at a state SSG_2 (or, making a controlled move from (SSG_2, $\emptyset$), to (SSG_2, $\{Avastin\}$)).

Luckily, there is a rather simple way around the undecidability of the control problem for rectangular hybrid automata that do not satisfy initialization. It is achieved by allowing for control moves only at discrete instants of time. Henzinger and Kopke [65] give an exponential-time algorithm for discrete-time safety control of rectangular hybrid automata with bounded and non-decreasing variables. They also show the problem to be EXPTIME-hard and discrete-time verification of rectangular hybrid automata to be solvable in PSPACE.

Even though our definition of timed CHAs does not require clocks to be bounded,

such a restriction would not impose a severe limitation. By bounding the clocks by some value that even the healthiest patient will never reach, we can thus aim for decidability without forfeiting any meaningful therapy. We let m denote this upper bound on the clocks. The algorithms from [65] do not directly apply to CHAs as their framework requires all discrete transitions to be controllable, whereas our cancer progression transitions are uncontrollable. However, they can be extended to include our framework via the following theorem.

Theorem 9 (Discrete control of bounded CHAs)**.** *The controller synthesis problem of bounded discretized CHAs for CTL formulas can be solved in EXPTIME.*

The proof of this theorem can be found in Appendix A.3. The (bisimilarity quotient of the) translated discretized automaton used to prove this result has at most $(|V| \times 2^{|\mathcal{D}|}) \times (2m+2)^{(n+1)}$ states, where $|V|$ is the size of the original state space, $|\mathcal{D}|$ is the total number of drugs, m is the bound on the clocks, and n is the number of clocks.

The controller synthesis problem is also EXPTIME-hard. This result follows immediately from the fact that for (the sub-class of) simple discrete event systems the controller synthesis problem for CTL goals is shown to be EXPTIME-complete[79].

Automatic therapy design Using the construction from the above result, therapies can be automatically generated as follows. Given a timed CHA H and a therapeutic goal ϕ specified using CTL, a therapy can be generated such that the controlled CHA H satisfies ϕ, by the following two steps:

1. Translate H into a discretized hybrid game (using the construction of Theorem 9).

2. Use existing controller synthesis algorithms for discrete event systems, such as the one developed in [78], on the resulting (bisimilarity quotient of) discrete

automaton to generate a therapy for ϕ (or show that this is impossible).

It should be noted that even though the problem of automatic therapy design is solvable, it is very complex: both the translation and discretization steps lead to state explosion, and even for discrete event systems controller synthesis problems are EXPTIME complete. For example, for our simple sample timed CHA from Figure 1.2 with one drug, two clocks and (for example) a clock bound of 10 the translation would give rise to a discrete automaton with thousands of states.

However, we believe that at this point the complexity should not pose an actual problem for the applicability of automatic therapy design. Partly, this is because current progression models are small and require few and granular clocks. Also, the control algorithm is not required to work very quickly or 'on-the-fly', but rather therapies can be computed using powerful computational machines, and may take a while.

Finally, it should be noted that Theorem 9 extends to partially observable CHAs as well.

Theorem 10 (Discrete control of bounded CHAs)**.** *The controller synthesis problem of bounded discretized CHAs with partial observability for epistemic CTL formulas can be solved in EXPTIME.*

The proof uses standard subset-based techniques such as the one in [27] common in epsitemic logic and is omitted. Basically, a new state is created for every set of states in the discretized hybrid automata that are indistinguishable to the therapist. This construction leads to another state explosion, exponential in the number of states of the discretized CHA. In this translation, tests are treated as actions, and are the only way observations can be obtained.

Part II

Cancer Progression:

Extraction

Chapter 3

CNV Data Analysis and Driver Gene Detection

Up to now, we have developed a formal model to describe cancer progression, and we have shown how, given a progression model, therapies can be automatically generated. The second part of this thesis revolves around a specific, perhaps more fundamental, problem of *extracting* progression models from available patient data.

With the initiation of the The Cancer Genome Atlas (TCGA) project[1], a lot of precise genetic information has been collected from tumor cells from many different patients with different kinds of cancers. However, this information is usually obtained only at one (or a few) points in time, and there is not a lot of (direct) *dynamic* data about the cancer's progression. The main reason underlying this situation is that obtaining timed data over the course of the disease is expensive compared to collecting data from the current pool of patients, which is thus much less common. It is thus an important challenge to extract this dynamic information from the available static data.

A first step towards cancer progression extraction is finding which genes are driving

[1] `http://cancergenome.nih.gov/`

cancer progression, the so-called 'driver genes'.

In the next section, we provide genetic evidence that CNV segment lengths are not exponentially distributed, but most likely power-law distributed instead. Based on this evidence, many tools that are used to analyze this type of data can be improved. In this chapter, we focus on one such statistical method developed in [76] for finding the driver genes through more accurate statistical null-models in Section 3.2.

Using these selected driver genes, we will develop (and test) algorithms to extract cancer progression models from static data in Chapter 4 . Most of the ideas and results presented in this chapter are presented in [97].

3.1 Copy Number Variation data: improved null model

One type of genomic cancer-patient data that has become commonly available is Copy Number Variation (CNV) data. Recall that CNV is structural variation in which relatively large regions of the genome are either amplified or deleted, leading to gain-of-function or loss-of-function of the genes contained in the affected regions.

CNV data consists of copy-number values of thousands of markers corresponding to different locations in the genome. To reduce the noise in this data, sets of neighboring markers are often combined resulting in contiguous segments of equal copy number, classified into *normal*, *amplified*, or *deleted* segments. Examples of such tools, usually called 'segmenters,' include GLAD [73], CBS [98], and a method developed by Mishra's group [36]. The abnormal segments correspond to duplication or deletion events and are used as input data to identify regions containing genes that are relevant for the development of cancer. (e.g., methods described in [76, 15]).

The underlying process generating these CNV segments is generally assumed to be memory-less, giving rise to an exponential distribution of segment lengths. In this chapter, we provide evidence from cancer patient data, which suggests that

this generative model is too simplistic, and that *segment lengths follow a power-law distribution instead*. We conjecture a simple preferential attachment generative model that provides the basis for the observed power-law distribution.

From a thorough understanding of the statistical properties of genomic copy-number data in cancer, one expects to discover (either directly or indirectly) improved oncogenomics features, using statistical inference tools which build upon more accurate null-models (examples of these tools include [73, 98, 36, 76, 15]). In Section 3.2, we provide one such improved estimator to an existing statistical method (due to Ionita et al. [76]) for detecting genetic regions relevant to cancer, which we achieve by incorporating the power-law distribution in the null. We use it to analyze three TCGA CNV data sets.

3.1.1 Evidence and fitting

We analyzed three CNV data sets from The Cancer Genome Atlas (TCGA): Lung Squamous Cell Carcinoma (LUSC 201 patients), Glioblastoma (GBM 299 patients), and Ovarian Serous Cystadenocarcinoma (OV 337 patients)[2]. The level 2 data was segmented using the segmentation algorithm of Daruwala et al. [36] and the empirical segment-length distributions of amplifications and deletions were fit to both power-law ($cx^{-\alpha}$) and exponential ($ce^{-\lambda x}$) distributions.

Figure 3.1 shows the segment length distribution and fitted functions for the deleted segments of the OV dataset, and Table 3.1 lists the numerical values of all fits, as well as their R^2 goodness of fit. Plots for the remaining data sets can be found in figure B.1 of Appendix B.1.

To determine threshold values for amplifications and deletions, we suitably modify the method described in [75], which implies that a segment is treated as an amplification (or resp. a deletion) if its value greater (or reps. smaller) than the mean plus (or

[2] `http://cancergenome.nih.gov/` The datasets used are: LUSC HMS_HG-CGH-415K_G4124A, GBM HMS_HG-CGH-244A, and OV HMS_HG-CGH-415K_G4124A.

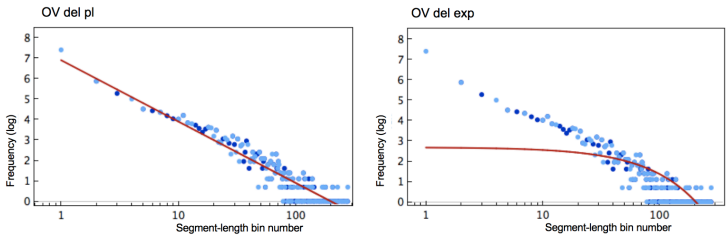

Figure 3.1: Segment length distribution and fitted functions of deleted segments from the OV dataset. The best power-law fit is shown on the left and the best exponential fit on the right. See Appendix B.1 Figure B.1 for the images showing the fits for all other data sets.

	best exponential fit		best power-law fit	
	function	R^2	function	R^2
LUSC Amp	$e^{-0.014x}$	0.65	$x^{-1.27}$	0.86
LUSC Del	$e^{-0.008x}$	0.45	$x^{-0.89}$	0.79
OV Amp	$e^{-0.014x}$	0.67	$x^{-1.39}$	0.91
OV Del	$e^{-0.013x}$	0.64	$x^{-1.30}$	0.91
GBM Amp	$e^{-0.015x}$	0.39	$x^{-1.01}$	0.71
GBM Del	$e^{-0.012x}$	0.60	$x^{-1.20}$	0.78

Table 3.1: for three TCGA data sets: LUSC, OV, and GBM.

resp. minus) twice the standard distribution ($AVG \pm 2STD$). The fit was estimated by collecting all the segment-lengths of segments above the amplification threshold value or below the deletion threshold value and taking a histogram of the segment lengths. To make the fit particularly sensitive to the tail of the distribution, we chose to fit the log of the data against the log of the exponential and power-law distributions.

As shown in Table 3.1, in all three datasets, the power-law fits the segment-length distributions better than the exponential one.

Several remarks about this result are due at this point. First, the remaining segments that are not considered amplifications or deletions (the 'Normals'), are not clearly power-law (nor exponentially) distributed (see Appendix B.1 Table B.1 for the actual fits, and Figure B.2 for an illustrative figure). The power-law distribution only appears to fit segments above (or below) a certain threshold. In Appendix B.1, we provide some analysis of the fits relative to a selected threshold. Second, taking the logarithm of the data is a way to magnify the difference between the power-law and

exponential fit, which occurs mostly in the tail. It should be noted, however, that it does not affect the relative goodness of the exponential and power-law fit, as can be verified by the results listed in Table B.3 in Appendix B.1.

Also, it should be remarked that the segmenter that was used to reduce noise in the data uses a non-informative prior, consistent with the assumption of an exponential distribution of segment-lengths [36]. Thus, the result presented here cannot be an artifact of the segmenter. To the contrary, the fact that we observe this distribution *despite* its use is quite striking, and we expect that if the null model of this method is improved, the results would be even more apparent (see Section 3.3, for some more discussions and other suggestions for future work).

3.1.2 Generative model

The observed power-law distributions for amplifications and deletions can be explained by a mechanism of preferential attachment. That is, once a region has large aberrations, it is more likely to acquire even more numerous large aberrations. One straightforward reason that could underlie this mechanism is that large amplifications or deletions lead to genomic instability and hence allow for subsequent large copy number aberrations.

3.2 Improving tools through more accurate statistical null-models

Most of the tools that are developed to analyze genomic data assume a non-informative exponential null-model for segment length distribution (e.g., segmenters [36] and tools for detecting cancer genes [76]). Knowledge of the fact that segment lengths are not exponentially distributed allows us to improve our null models and hence our tools. This resulting prior is especially important when there is not sufficient data available to accurately predict null-models from the data. In the next section we show how an

existing tool for detecting cancer genes can be improved.

3.2.1 Statistical method for detecting cancer genes

In this section we adopt a method described in [76] for finding cancer driver genes from copy number variation data by building upon the assumption that segment lengths are power-law distributed.

Cancer genes are generally divided into two types: tumor suppressor genes (TSGs) and oncogenes (OGs). TSGs prevent tumor development by regulating cell growth. A loss or reduction in its function (for example by a deletion), can lead to uncontrolled cell division and allows the cancer to progress. Oncogenes, on the other hand, are genes whose function promote proliferation. Gain-of-function mutations (like amplifications), or overexpression, promote tumor progression. In the case of TSGs a deletion of a part of the gene will cause a loss-of function, while for OGs the gene needs to be amplified as a whole to cause a gain-of-function.

The algorithm for finding TSGs and OGs enumerates all possible intervals and assigns to them a score function that measures the likelihood of this being a driver gene. This score function can be described as follows:

For any interval I the strength of the association between deletions in I or amplifications of I and the disease is quantified by analyzing the genomic data for many individuals with a specific type of cancer. For this purpose, a metric called *Relative Risk* ($RR_{\text{event } I}$) assigns a numerical value to any event, a deletion or amplification of an interval, which thus compares the probability of the disease occurring with or without the event. Informally, $RR_{\text{event } I}$ measures the degree to which the occurrence of event I *raises the probability* of the disease incidence. *Probability raising* is an important ingredient of probabilistic causality developed by Suppes in [107]. In Chapter 4 we will develop a progression extraction method based on this notion, and we will examine its properties in great detail.

Formally, $RR_{\text{event } I}$ is defined as follows:

$$\begin{aligned} RR_{\text{event } I} &= ln \frac{\mathcal{P}(\text{ disease } \mid \text{ event } I\ )}{P(\text{ disease } \mid \text{ NOT event } I\ )} \\ &= ln\left[\frac{\mathcal{P}(\text{ event } I \mid \text{ disease })}{\mathcal{P}(\text{ NOT event } I \mid \text{ disease })} \times \frac{\mathcal{P}(\text{ NOT event } I\ )}{\mathcal{P}(\text{ event } I\ )}\right] \\ &= ln\left[\frac{\mathcal{P}(\text{ event } I \mid \text{ disease })}{\mathcal{P}(\text{ NOT event } I \mid \text{ disease })}\right] + \left\{-ln\left[\frac{\mathcal{P}(\text{ event } I\ )}{\mathcal{P}(\text{ NOT event } I\ )}\right]\right\}, \quad (1) \end{aligned}$$

where, in case of a deletion, "event I" denotes the event that *at least part* of I is deleted. We call this event 'I lost'. In case of an amplification "event I" denotes the event that there exists an amplified interval that *fully includes* I. We call this event 'I gained'.

The first term in equation (1) can be computed from the available tumor samples:

$$\frac{\mathcal{P}(\text{ event } I \mid \text{ disease })}{\mathcal{P}(\text{ NOT event } I \mid \text{ disease })} = \frac{n_{\text{ event } I}}{n_{\text{ NOT event } I}},$$

where $n_{\text{ event } I}$ (or $n_{\text{ NOT event } I}$) is the number of patients in whose tumor genomes the event I occurs (or does not occur). Note that becasue of the intrinsic differences between TSGs and OGs in case of deletions, the longer the segment the larger $\frac{n_{\text{ event } I}}{n_{\text{ NOT event } I}}$ whereas in case of amplifications the situation is reversed: longer segments have smaller $\frac{n_{\text{ event } I}}{n_{\text{ NOT event } I}}$. This imbalance is corrected for by the second part of (1),

$$-ln\left[\frac{\mathcal{P}(\text{ event } I\ )}{\mathcal{P}(\text{ NOT event } I\ )}\right],$$

which incorporates prior information inherent in the statistical distribution of amplifications and deletions.

To compute the prior score, we assume that, at any genomic location, a breakpoint (starting point) may occur as a Poisson process at a rate of $\mu \geq 0$. We consider two different μ's: one for amplifications μ_{AMP} and the other for deletions μ_{DEL}, but we drop the subscript when no confusion arises. Segments are modeled as vectors. Starting at a breakpoint and moving left (or right) with probability $\frac{1}{2}$. The length

t of each segment is distributed according to a power-law distribution: $t^{-\alpha}$, with $1 \leq \alpha \leq 2$. Let ϵ be the constant that represents the shortest length an interval could possibly have.

Given these assumptions we can derive the prior probability that an interval I is amplified or deleted.

Proposition 1. *Assuming that segment lengths are power-law distributed :*

1. *The probability that an interval $I = [a, b]$ is lost is as follows:*

$$\begin{aligned}\mathcal{P}([a,b]\ lost) &= 1 - e^{-\mu(b-a)} \times \\ & e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]} \times \\ & e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-b)^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]};\end{aligned}$$

2. *The probability that an interval $I = [a, b]$ is gained is as follows:*

$$\begin{aligned}\mathcal{P}([a,b]\ gained) &= 1 - e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{b^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right]} \times \\ & e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-a)^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right]};\end{aligned}$$

where $[0, G]$ represents the region of interest (e.g. a chromosome) and $[a, b]$ is an interval within this region. It is assumed that $\epsilon \ll G$.

□

The proof of this proposition can be found in Appendix 1.

The parameter α can be estimated from the data as described in Section 3.1.1. The values of the μ_{DEL} and μ_{AMP} parameters are the mean number of amplifications and deletions per unit length respectively and can be computed directly from the segmented data as well.

The constant ϵ can take any value. If we assume the value of ϵ is 1 unit (corresponding to a single probe in microarray data or a single base in sequencing data) the

probability that a segment is lost approaches:

$$\begin{aligned}\mathcal{P}([a,b] \text{ lost}) &= 1 - e^{-\mu(b-a)} \times \\ & e^{-\mu\frac{1}{2}\left[\frac{a^{2-\alpha}}{2-\alpha}\right]} \times \\ & e^{-\mu\frac{1}{2}\left[\frac{(G-b)^{2-\alpha}}{2-\alpha}\right]};\end{aligned}$$

Similarly for amplifications:

$$\begin{aligned}\mathcal{P}([a,b] \text{ gained}) &= 1 - e^{-\mu\frac{1}{2}\left[\frac{b^{2-\alpha}-(b-a)^{2-\alpha}}{2-\alpha}\right]} \times \\ & e^{-\mu\frac{1}{2}\left[\frac{(G-a)^{2-\alpha}-(b-a)^{2-\alpha}}{2-\alpha}\right]}.\end{aligned}$$

The *RR* score can be used to estimate the location of tumor suppressor genes and oncogenes. The simplest algorithm first computes the score for all intervals with value in a range determined by lower and upper bounds, and then picks the highest scoring interval on each chromosome. Many other algorithms can be imagined. For example, one can use two scoring functions to compute the left and right boundaries of the interval separately. The final step of the algorithm is significance testing of the obtained intervals. The methods as described in [76] for tumor suppressor genes, and in [75] for oncogenes can be directly applied. Both methods assign a p-value for every putative TSG or oncogene using tools from *scan statistics* [115].

We have implemented the algorithm by computing the RR score for each interval while keeping track of the highest scoring interval. Because each interval needs to be visited only once the time complexity is linear in the number of intervals.

Instead of finding only the interval with maximum score on each chromosome we can let the algorithm pick higher scoring intervals. One straightforward way is to pick the n non-overlapping significantly amplified/deleted intervals with the highest score, by keeping track of a list of results while going through the set of all intervals. One shortcoming of this method if that if we want to go through the list of intervals only once (or a linear number of times), the order in which the intervals are scored

and stored may influence the result. More refined methods, that optimize total score and/or use multiple hypothesis testing, can be imagined.

3.2.2 Performance comparison

To be able to test the influence of the improved null model, we have applied the previously described algorithm with both the original exponential and the power-law null models to the three TCGA datasets: OV, LUSC and GMB.

To compare the two models we asked which of the commonly amplified or deleted genes in the three cancer types were found by the respective algorithms. The results are summarized in table 3.2. Consistent with our expectation, the power-law based model performs (slightly) better than the exponential model.

Cancer	Gene	Power-law	Exponential
OV	BRCA1	no	no
	BRCA2	no	no
	ERBB2	no	no
	K-ras	**yes**	**yes**
	AKT2	no	no
	PIK3CA	no	no
	c-MYC	**next**	**no**
	p53	no	no
LUSC	SOX2	no	no
	PDGFRA	no	no
	FGFR1	**yes**	**no**
	WHSC1L1	**next**	**no**
	CDKN2A	**yes**	**yes**
GBM	EGFR	**next**	**next**
	MDM2	no	no
	PDGFR	no	no
	CDK4	no	no
	Rb	no	no
	CDKN2A	**yes**	**yes**

Table 3.2: List of genes that are commonly altered in OV, LUSC and GBM cancer cells, and whether or not they were found by the power-law and exponential methods using the three highest scoring non-overlapping intervals. A more detailed version of this table can be found in Table B.6 in Appendix B.3.

Note that despite the (slightly) better performance of the algorithm with the power-law null model over the exponential model, the difference between the two performances is comparable and both algorithms appear to miss many cancer genes. These false negatives are indicative of the inadequacy of both genomic resolution in the data and sample sizes, currently available. Both methods can be further improved by including additional information (e.g., gene-ontologies, gene-networks or pathways). In such settings, as well as when regions for many more genes are checked, the contribution from more accurate null model is expected to be more pronounced.

We offer several explanations for the missing genes. For example, the algorithm only picks out a few (in this case three) high scoring intervals per chromosome. Often, these intervals are in the same region close to a single gene, which causes other regions of interest to be overlooked. For example, in the OV dataset, all three deleted intervals that were found on chromosome 17 were close to (but not exactly overlapping with) BRCA1. It became therefore impossible to find P53, which also lies on chromosome 17, as well. This problem can be resolved by adopting more sophisticated statistical methods for selecting high-scoring intervals.

In addition, regions either right next to actual genes or close to the centromere were often identified as likely cancer genes. We expect this type of error to disappear as methods for CNV data collection become more precise. In the next section, we briefly mention several other possible ways to improve the method for finding driver genes.

3.3 Conclusion

In this chapter, we have provided evidence suggesting that the segment lengths of CNV amplifications and deletions in cancer cells follow a power-law distribution instead of the commonly assumed exponential distribution. This evidence suggests a generative

mechanism of preferential attachment: many long amplifications and deletions lead to even more long amplifications and deletions. Even though our data analysis rules out exponentially distributed segment lengths, and the evidence for power-law distribution is compelling, other distributions (such as log-normal or stretched exponential, see Table B.4 in Appendix B.1) cannot be completely excluded on the basis of this evidence.

Especially in cases where only a small sample of data is available to estimate the prior distribution from the data, knowledge about the statistics of CNV data allows us to improve our analytic tools. As an example, we have demonstrated how the technique for finding cancer driver genes described in [76] can be modified to incorporate the power-law distribution and, as our preliminary results indicate, how the power-law-based scan-statistics algorithm outperforms the exponential one. Once inferred, the set of cancer driver genes can be used as input to cancer progression extraction algorithms to derive progression models from static cancer patient data (see e.g., [39, 40, 55, 54]), leading to improved diagnostics, prognostics, and targeted therapies.

The tool to find cancer driver genes can be further improved in several ways. For example, we will need to incorporate a preferential attachment model to the segmenter that analyzes the genomic data from each cell-type; use more accurate priors of the *distribution of breakpoints* that are known to occur in different cell-types; apply more sophisticated statistical tools for picking high-scoring intervals by incorporating prior biological knowledge; and include such information (i.e., how pathways affect the cell-states) in combination with precise correction for multiple hypothesis testing in order to make the final results more meaningful.

In the next chapter we will build progression models, or CHAs, form the output of driver gene detection algorithms as the one discussed in this chapter. Given a set of states (such as mutated genes, or hallmarks), we will aim to answer questions such

as what are the likely orders in which these states occur, and whether more can be inferred about the *causality* governing them.

Chapter 4

Progression Extraction

We have arrived at the final chapter of this thesis, where we *extract* progression models from cross-sectional data.

As we have mentioned before, a lot of precise and comprehensive genetic information has been collected in the last couple of decades from tumor samples, and sets of specific 'driver events' (e.g., genetic mutations, CNVs, expression profiles, hallmarks, etc.) have been identified that drive each cancer's progression, using a variety of methods (the method described in Chapter 3 being one of them. Another well-known method using CNV data is [15]).

However, despite this flood of information, relatively little is known about the *dynamics* of cancer progression and the *order* in which these driving events are likely to occur. The main reason for this situation is that information is usually obtained only at one (or a few) points in time, rather than over the course of the disease. Because obtaining precise timed information from patient data still lies far ahead in the future, it is an important challenge to extract this dynamic information from the available static data.

In this chapter, we propose a method to extract progression *trees and forests* using mathematical notions of probabilistic causality. Using synthetic data we show how

this method outperforms an existing tree reconstruction algorithm.

The problem of reconstructing progression models from cross-sectional patient data is not a new one, and several progression extraction methods have been proposed. Most notable among them, for our discussion, is the seminal paper by Desper et al. in which they developed a tree reconstruction algorithm, called 'oncotrees' [39]. In their framework, which we will discuss in detail in Section 4.2, nodes represent amplified or deleted driver genes, and edges correspond to possible progressions from one genetic event to the next. The choice of which edges to include in a branching tree is based on a functional that assigns a weight $w_{a,b}$ to each pair of events a, b based on how often each event occurs, and how often they occur together in the same tumor – thus measuring *correlation*. The oncotree is constructed as the rooted tree whose total weight (sum of all the weights of the edges) is maximized. The oncotree method has been used to derive progression trees for various types of cancers such as breast cancer [83], head and neck carcinoma [72] and melanoma [102].

In this chapter we present a new tree reconstruction algorithm that improves upon Desper's algorithm. In Section 4.1 and Section 4.2 we formally introduce the problem and the method introduced by Desper et al. In Section 4.3 we will discuss our main tool, namely *probabilistic causality* and *probability raising* in particular. We show several theoretical results regarding this notion, the most unexpected one being Proposition 2, which connects probability raising between two events with the relative frequency of their occurrence. In Section 4.4 we present the reconstruction technique, and in Section 4.5 we compare our method against Desper's method, using both simulated and synthetic data, concluding that our method outperforms Desper's one. In Section 4.6 we discuss and implement several techniques to increase robustness against noise. Finally, in Section 4.7 we connect our progression extraction algorithm back to CHAs, and we discuss how our algorithm can be extended to reconstruct complete CHAs including some of the missing CHA parameters such as *timing* and

the effects of *drugs*.

Many of the ideas and results presented in this chapter are presented in [96].

4.1 The problem

The set-up of the extraction problem may be described as follows. Assuming that we have a set G of n events (tumor genes, CNVs, etc.) and m samples, we represent a dataset as a $m \times n$ boolean matrix. In this matrix, an entry $(k, l) = 1$ if the event l was observed in sample k, and 0 otherwise. The problem we aim to solve in this chapter is that of trying to extract a progression *tree* (G, E) structure from this matrix. More precisely, we aim to reconstruct proper *rooted trees* that satisfy the basic requirements: (i) each node has at most one incoming edge, (ii) there is no incoming edge to the root, and finally (iii) there are no *cycles*. The root of the tree can be modeled using a special node, not included in the set of samples. In this case, the root is used to connect several disconnected components, in which event the reconstruction problem generalizes to that of reconstructing *forests*.

We assume that each progression tree labeled with $\alpha(e)$, denoting the probability of each edge, generates a distribution where the probability of observing a sample with set of alterations $G^* \subseteq G$ is

$$\prod_{e \in E'} \alpha(e) \cdot \prod_{\substack{(u,v) \in E \\ u \in G^*, v \notin G}} \Big[1 - \alpha(u, v)\Big] \ ,$$

where $E' \subseteq E$ is the set of edges connecting the root, denoted $\diamond$ to the events in G^*. [1]

[1] Note that in this chapter we assume that edges are independent events. This implies that at a state in the progression with two outgoing edges, it is possible to acquire either one of the successors next, *or both*. In CHA models, on the contrary, each unit can progress along *one* progression path at a time. We will come back to this difference in greater detail in Section 4.7.1, where we also suggest how to consolidate this difference in future work.

4.2 Oncotrees

In [39, 40] Desper et al. developed an algorithm to extract progression trees, called 'oncotrees', from static CNV data. In these trees nodes represent amplified or deleted driver genes, and edges correspond to possible progressions from one CNV event to the next. The problem is exactly as described above, and each tree is rooted in the special root $\diamond$, not among the set of events.

The choice of which edges to include in a tree is based on a functional that assigns a weight $w_{a,b}$ to each pair of events a, b based on how often each event occurs, and how often they occur together in the same tumor – thus measuring *correlation*. Formally, the weight functional is defined as follows:

$$w_{a,b} = \log\left[\frac{\mathcal{P}(a)}{\mathcal{P}(a)+\mathcal{P}(b)} \cdot \frac{\mathcal{P}(a,b)}{\mathcal{P}(a)\mathcal{P}(b)}\right] . \tag{4.1}$$

Weights also include the fake root $\diamond$, added to each sample of the dataset. In this definition the rightmost term is the (symmetric) *likelihood ratio* for a and b occurring together. The leftmost term guarantees asymmetry and reflects *temporal priority*, measured by rate of occurrence. That is, if a occurs *more often* than b, then it likely occurs *earlier*, and the inequality

$$\frac{\mathcal{P}(a)}{\mathcal{P}(a)+\mathcal{P}(b)} > \frac{\mathcal{P}(b)}{\mathcal{P}(a)+\mathcal{P}(b)}$$

holds.

The oncotree tree is constructed as the rooted tree whose total weight (sum of all the weights of the edges) is maximized. If n is the total number of genetic events, the optimal tree is reconstructed in $O(n^2)$ steps using Edmond's seminal result [42].

By construction, the resulting graph is a proper tree rooted in $\diamond$, such that each event occurs only once in the oncotree, and *confluences* are absent, i.e. any event is

caused by at most one other event.

The oncotree method has been used to derive progression trees for various cancer data sets (e.g.,[83, 72, 102]), and even though several extensions of the method exist that derive different graph models, to this day, it is the only reconstruction method that reconstructs trees and forests as described in Section 4.1.

4.3 Probabilistic causality

We will develop our reconstruction algorithm based on precise notion of *probabilistic causality*. But, before we present this method, we review the notion of probabilistic causality and its properties in a bit more detail.

In his seminal work [107], Suppes proposed a notion of causality which can be summarized as follows. For any two events c and e, under the mild assumptions that $0 < \mathcal{P}(c), \mathcal{P}(e) < 1$, the event c *causes* e if

(i) the cause happens *before* the effect:

$$(\text{TP}) \quad t_c < t_e \ , \tag{4.2}$$

where t_c (resp. t_e) is the time at which c (resp. e) happens. TPis commonly referred to as temporal priority property for causality,

(ii) the cause *raises the probability* of the effect:

$$(\text{PR}) \quad \mathcal{P}(e \mid c) > \mathcal{P}(e \mid \overline{c}) \ , \tag{4.3}$$

PR is commonly referred to as the *probability raising* property of causality.

The following is an important property underlying PR.

Property 1 (Dependency)**.** *Whenever the* PR *holds between a and b, then the events are* statistically dependent *in a positive sense, that is*

$$\mathcal{P}(b \mid a) > \mathcal{P}(b \mid \overline{a}) \Leftrightarrow \mathcal{P}(a, b) > \mathcal{P}(a)\mathcal{P}(b) \ . \tag{4.4}$$

This property, as well as Property 2, is a well-known fact of probability raising, and its derivation is straightforward. For a good discussion of probabilistic causality, including probability raising, its properties and its problems we refer to [70].

We would like to use PR criteria to discriminate whether there exist a causality relation between two events a and b, and when a should be placed ahead of b in the progression tree.

Unfortunately, we cannot simply state that a causes b when

$$[\text{PR } a \rightarrow b] \quad \frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > 1 \ ,$$

since PR is known to be symmetric in the following sense:

Property 2 (Mutual PR)**.** $\mathcal{P}(b \mid a) > \mathcal{P}(b \mid \overline{a}) \Leftrightarrow \mathcal{P}(a \mid b) > \mathcal{P}(a \mid \overline{b})$.

That is, if a raises the probability of b, then b raises the probability of a as well.

Instead, to determine causes and effects among the genetic events, we can use *degree* of probability raising to decide the direction of causality between two events. That is, if a raises the probability of b *more* than the other way around, then a is a more likely cause of b than b of a. It turns out that indeed, PR is not fully symmetric, and the *direction* of probability raising depends on the relative frequencies of events. We make this asymmetry precise in the following proposition.

Proposition 2 (Probability raising and probabilities)**.** *For any two events a and b*

such that the probability raising $\mathcal{P}(a \mid b) > \mathcal{P}(a \mid \overline{b})$ *holds, we have*

$$\mathcal{P}(a) > \mathcal{P}(b) \iff \frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \ .$$

That is, given that probability raising holds between two events, a raises the probability of b *more* than b raises the probability of a, if and only if a occurs more frequently than b. The proof of this proposition is not completely straightforward and can be found in Appendix C (as well as other results in this chapter). From this result it follows that if we measure the timing of an event by the rate of its occurrence (that is, $\mathcal{P}(a) > \mathcal{P}(b)$ implies that a happens *before* b), then this notion of PR subsumes the same notion of *temporal priority* found in trees, analogue to Desper's method.

Given this result, we can define PR $a \to b$ as follows: we state that a causes b whenever

$$[\text{PR'}\ a \to b] \quad \mathcal{P}(b \mid a) > \mathcal{P}(b \mid \overline{a}) \quad \text{and} \quad \mathcal{P}(a) > \mathcal{P}(b) \ ,$$

That is, a causes b if a *raises the probability* of b, and a occurs more frequently than b. We will use this notion of PR to determine causality relation to extract progression trees in the following section.

4.4 Using causality to derive progression trees

Our reconstruction method is described in Algorithm 1. The algorithm is very similar in spirit to Desper's algorithm, with the main difference being an alternative weight functional based on probability raising. We define the weight as follows

$$m_{a,b} = \frac{\mathcal{P}(b \mid a) - \mathcal{P}(b \mid \overline{a})}{\mathcal{P}(b \mid a) + \mathcal{P}(b \mid \overline{a})} \ .$$

and we include an edge (a, b) in the tree if 1) a raises the probability of b, 2) $m_{a,b} > m_{b,a}$, and 3) compared to all other possible incoming edges of b, a raises the

Algorithm 1 Tree-like reconstruction with probability raising

1: consider a set of genetic events $G = \{g_1, \ldots, g_n\}$ plus a fake event $\diamond$, added to each sample of the dataset;
2: define a $n \times n$ matrix M where each entry contains

$$m_{i,j} = \frac{\mathcal{P}(j \mid i) - \mathcal{P}(j \mid \bar{i})}{\mathcal{P}(j \mid i) + \mathcal{P}(j \mid \bar{i})}$$

according to the observed probability of i and j;
3: [PR causality] define a tree $\mathcal{T} = (G \cup \{\diamond\}, E, \diamond)$ where $(i, j) \in E$ for $i, j \in G$ if and only if:

$$\frac{\mathcal{P}(j \mid i)}{\mathcal{P}(j \mid \bar{i})} > 1 \quad \text{and} \quad m_{i,j} \geq m_{j,i} \quad \text{and} \quad \forall i' \in G . m_{i,j} > m_{i',j} \, .$$

4: [Correlation filter] define $G_j = \{g_i \mid \mathcal{P}(i) > \mathcal{P}(j)\}$, replace edge (i, j) with edge $(\diamond, j)$ if, for all $g_w \in G_j$, it holds

$$\frac{1}{1 + \mathcal{P}(j)} > \frac{\mathcal{P}(w)}{\mathcal{P}(w) + \mathcal{P}(j)} \frac{\mathcal{P}(w, j)}{\mathcal{P}(w)\mathcal{P}(j)} \, .$$

probability of b the most. In the algorithm, we use a normalized version of the PR ratio to gain a higher tolerance to samples slightly diverging from the tree-induced distribution (see also Section 4.6). Notice that the normalization we used is monotonic with respect to the PR ratio.

Proposition 3 (Monotonic normalization)**.** *For any two events a and b we have*

$$\frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \iff m_{b,a} > m_{a,b} \, .$$

Also, $-1 \leq m_{a,b} \leq 1$. When $m_{a,b}$ tends to -1 the events appear disjointly (anti-causality), when it tends to 0 no causality/anti-causality can be inferred and when it tends to 1 the causality from a to b is robust.

Along the lines of Desper we augment the set of events with a fake root $\diamond$ with $\mathcal{P}(\diamond) = 1$ to separate trees within the forest. Algorithm 1 initially builds a single big tree using only the new weight m based on PR criterion. Then, we connect all the

nodes that are sufficiently correlated with it, to the fake root. More precisely, for any node j, if the correlation of j with the fake root

$$\frac{\mathcal{P}(\diamond)}{\mathcal{P}(\diamond)+\mathcal{P}(j)}\frac{\mathcal{P}(\diamond,j)}{\mathcal{P}(\diamond)\mathcal{P}(j)}=\frac{1}{1+\mathcal{P}(j)}$$

is larger than its correlation with each elements higher up in the tree (with a higher probability), then we remove the edge coming into j and add an edge $(\diamond, j)$ to the forest instead. While performing this update, we keep all the children of j intact.

Note that the weight used to discriminate whether causality is present between nodes or the fake root $\diamond$ is the same as Desper's. The reason for using correlation in this case, and not probability raising, is that in case of an event (the root) with probability 1, probability raising is not well-defined. This method allows us to reconstruct forests consisting of trees rooted in the children of the fake root.

This algorithm reconstructs well-defined trees in the sense that no transitive connections and no cycles can appear.

Theorem 11 (Algorithm correctness)**.** *Algorithm 1 reconstructs a well defined tree $\mathcal{T}$ without disconnected components, transitive connections and cycles.*

The proof of this Proposition follows immediately from Proposition 2 and can be found in Appendix C.

4.5 Performance comparison of both methods

4.5.1 Performance comparison using synthetic data.

To compare our Algorithm 1 with the oncotree approach by Desper, we constructed synthetic datasets for various random trees/forests, and applied both methods to data-sets generated from these trees. The trees reconstructed by both methods are

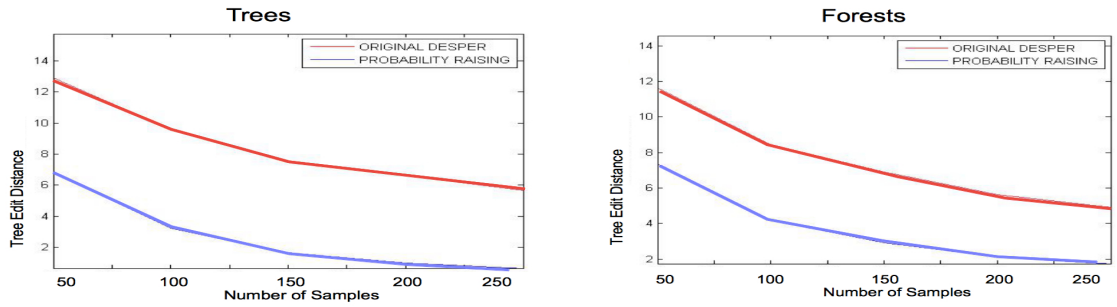

Figure 4.1: **Tree-alike reconstruction: comparison on synthetic data.** Performance of both our (blue line) and the Desper's correlation based (red line) algorithms measure by the average Tree Edit Distance between the reconstructed tree and the tree (left)/forest (right) used to generate the data. A lower valued TED indicates a better performance. We used synthetic datasets consisting of 250 random trees/forests (20 genes for each tree/forest) and 50/100/150/200/250 samples for each topology. For each configuration 25 different datasets are generated.

compared to the original trees, and the differences between the trees provide a measure of the respective quality of the algorithms.

Because widely branching trees are harder to reconstruct than slim trees, we chose to generate trees that are logarithmic in depth (with respect to the number of genetic events considered), thus representing a hard test-case for both reconstruction algorithms.

Both the algorithms are run on the same synthetic datasets consisting of 250 random trees/forests (20 genes for each tree/forest), and 50/100/150/200/250 samples for each topology. Also, for each of these configuration we generate 25 different datasets and average the performance results. The performance of each algorithm is measured by adopting the well-known *Tree Edit Distance* (TED, [118]). Intuitively, the TED is defined as the minimum-cost sequence of node edit operations that transform one tree into another.[2] We evaluated the TED between the reconstructed tree and the tree/forest used to generate the data, the lower the distance the better is the reconstructed tree. In Figure **??** (left) we show the result on data generated by a

[2]Three operations are allowed: *relabeling* a node, *deleting* a node, and *inserting* a node.

single tree, and in (right) we generalize it to forests.

We observe that for our Algorithm 1 TED is, on average, lower than that of Desper's algorithm, indicating that our algorithm outperforms the correlation-based technique. On trees, Algorithm 1 has an average TED of almost 7 based on a reconstruction from 50 samples, while the correlation-based approach has an average TED of almost 13; Algorithm 1 improves significantly when the size of the dataset increases, as the TED drops down to 0 for 250 samples, while it remains at a high level of almost 6 for Desper's approach. Our algorithm performs worse on forests, but it still performs much better than just the correlation based approach. These results suggest that the PR approach is more suitable to infer causality relation.

4.5.2 Performance comparison on real data

The results in the previous section indicate that our method outperforms Desper's Oncotree algorithm. In this section, we investigate whether and how these theoretical differences affect progression extraction when real cancer patient datasets are analyzed.

To test our reconstruction approach on a real dataset we applied it to the *ovarian cancer* dataset made available within the oncotree package from [39]. The data is collected through the public platform SKY/M-FISH [85], used to allow investigators to share and compare molecular cytogenetic data. The data is obtained by using the *Comparative Genomic Hybridization* technique (CGH) on samples from *papillary serous cystadenocarcinoma* of the ovary. This technique uses fluorescent staining to detect CNV data at the resolution of chromosome arms. Presently this kind of analysis can be done at a higher resolution, making this dataset rather outdated. Nevertheless, it can still serve as a perfectly good test-case for our approach. The seven most commonly occurring events are selected from the 87 samples, and the set of events are the following gains and losses on chromosomes arms $G = \{8q+, 3q+, 1q+, 5q-, 4q-, 8p-, Xp-\}$ (where e.g., $4q-$ denotes a deletion of the q arm of the 4^{th} chromosome).

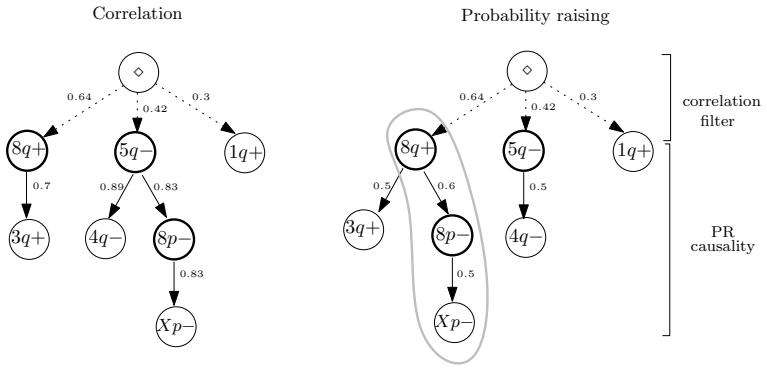

Figure 4.2: **Tree-like reconstruction of ovarian cancer progression.** Trees reconstructed with the correlation-based method by Desper and with Algorithm 1. The set of CGH events considered are gains on $8q$, $3q$ and $1q$ and losses on $5q$, $4q$, $8p$ and Xp. Events on chromosomes arms containing the key genes for ovarian cancer are in bolded circles. In the left tree all edge weights are the observed probabilities of events. In the right the full edges are the causality inferred with the PR and the weights represent the normalized coefficients of Algorithm 1. Weights on dashed lines are as in the left tree.

In Figure 4.2 we compare the trees reconstructed with correlation and PR. [3] The PR approach differs from the correlation-based predicting the causal sequence of alterations

$$8q+ \ \rightarrow \ 8p- \ \rightarrow \ Xp- \ .$$

At this point, we do not have a biological interpretation for this result. However, we do know that common cancer genes reside in these regions. (i.e.., the tumor suppressor gene PDGFR on $5q$, and the oncogene MYC on $8q$), and loss of heterozygosity on the short arm of chromosome 8 is very common[4]. Recently, it has been reported that this locations contains many *cooperating* cancer genes $8p$ [116].

In order to assign a confidence level to these inferences we applied a both parametric and non-parametric *bootstrapping methods* to our results [43]. Essentially, this tests consists of using the reconstructed trees (in the parametric case), or the probability observed in the dataset (in the non-parametric case) to generate new (synthetic)

[3] This Figure, as well as Figures 4.4 and 4.6 were drawn by Giulio Caravagna, and used with permission.

[4] See e.g., `http://www.genome.jp/kegg/`

Desper's algorithm (overall confidence 8.3%)

$\rightarrow$	$8q+$	$3q+$	$5q-$	$4q-$	$8p-$	$1q+$	$Xp-$
$\diamond$	**.99**	.06	**.51**	.22	.004	**.8**	.06
$8q+$	0	**.092**	.08	0.16	0.4	.02	.007
$3q+$	.002	0	.04	0	0	.09	.04
$5q-$	.001	.002	0	**.52**	**.39**	.009	.16
$4q-$	0	0	.27	0	.14	.05	.11
$8p-$	0	0	.07	.08	0	.004	**.59**
$1q+$	0	0	0	.004	0	0	0
$Xp-$	0	0	.003	.003	.04	.01	0

Algorithm 1 (overall confidence 8.6%)

$\rightarrow$	$8q+$	$3q+$	$5q-$	$4q-$	$8p-$	$1q+$	$Xp-$
$\diamond$	**.99**	.06	**.51**	.22	.004	**.8**	.06
$8q+$	0	**.92**	.06	.16	**.62**	.01	.008
$3q+$	.002	0	.03	.002	0	.09	.04
$5q-$	.001	.002	0	**.5**	.26	.009	.17
$4q-$	0	0	.29	0	.09	.05	.12
$8p-$	0	0	.07	.08	0	.004	**.59**
$1q+$	0	0	0	.004	0	0	0
$Xp-$	0	.001	.003	.004	.01	.01	0

Figure 4.3: **Estimated confidence for ovarian progression.** Frequency of edge occurrences in the non-parametric bootstrap test, for the trees shown in Figure 4.2. Colors represent confidence: light gray is $< .4\%$, mid gray is $.4\% \div .8\%$ and dark gray is $> .8\%$. Bold entries are the edges recovered by the algorithms.

datasets, and then reconstruct progressions again. (See e.g., [46] for an overview of these methods). The confidence is given by the number of times the trees in Figure 4.2 are reconstructed from the generated data. A similar approach can be used to estimate the confidence of every edge seperately. For Desper's algorithm the *exact tree* is obtained 83 times out of 1000 non-parametric resamples, so its estimated confidence is 8.3%. For our algorithm the confidence is 8.6%. In the non- parametric case, Desper's confidence is 17% while our is much higher: 32%. For the non-parametric case, edges confidence is shown inTable 4.3. Most notably, our algorithm reconstructs the inference $8q+ \rightarrow 8p-$ with high confidence (confidence 62%, and 26% for $5q- \rightarrow 8p-$), while the confidence of the edge $8q+ \rightarrow 5q-$ is only 39%, almost the same as $8p- \rightarrow 8q+$ (confidence 40%).

Analysis of other datasets We report differences between the reconstructed trees also based on datasets of gastrointestinal and oral cancer ([57, 101] respectively). In the case of gastrointestinal stromal cancer, among the 13 CGH events considered in [57] (gains on $5p$, $5q$ and $8q$, losses on $14q$, $1p$, $15q$, $13q$, $21q$, $22q$, $9p$, $9q$, $10q$ and $6q$), the correlation identifies the path progression

$$1p- \rightarrow 15q- \rightarrow 13q- \rightarrow 21q-$$

while probability raising reconstructs the branch

$$1p- \rightarrow 15q- \qquad\qquad 1p- \rightarrow 13q- \rightarrow 21q- \ .$$

In the case of oral cancer, among the 12 CGH events considered in [101] (gains on $8q$, $9q$, $11q$, $20q$, $17p$, $7p$, $5p$, $20p$ and $18p$, losses on $3p$, $8p$ and $18q$), the reconstructed trees differ since correlation identifies the path

$$8q+ \rightarrow 20q+ \rightarrow 20p+$$

while probability raising reconstructs the path

$$3p- \rightarrow 7p+ \rightarrow 20q+ \rightarrow 20p+ \ .$$

4.6 Increasing robustness

The results in the previous sections are very promising, and conceptually our algorithm is very powerful. However, as is shown in figure 4.4, once we introduce noise is into the system, the performance of our algorithm quickly degrades to the level of Desper's correlation based approach. In this section we suggest a simple adaptation to the

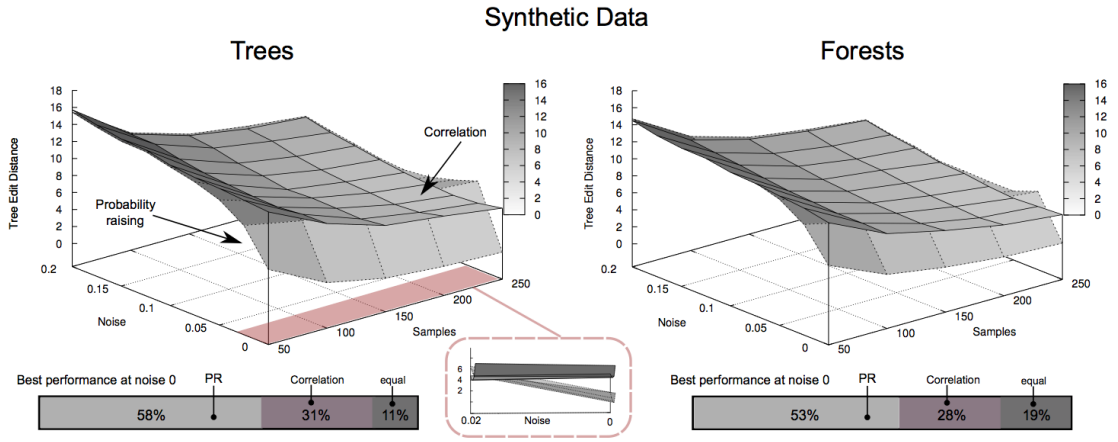

Figure 4.4: **Tree and forest reconstruction: comparison on synthetic data in the presence of noise.** Specifications are exactly as in Figure 4.1, with a noise parameter $\nu \in [0, 0.2]$ (discretized with step 0.05).

algorithm that will increase its robustness to noise.

Noise can be modeled as follows. We introduce a *noise* parameter $0 \leq \nu < 1$, that measures the probability that a measurement is flipped (thus introducing both false negatives and false positives), such that we have on average $|G|\nu$ errors in each sample. Noise is not uncommon in this type of data and can be caused by experimental/measuring errors as well as the presence of heterogeneity. [5]

Intuitively, the reason that noise affects the performance of our algorithm a lot, is due to the fact that rather than just using correlation, probability raising includes the probability of an event in the *absence* of its cause. Because this probability if rather small, this information is very precise and a slight divergence due to noise can have large effects on the weight functional.

To improve robustness to noise, we introduce correlation into our weight functional, while keeping the beneficial effects of probability raising. The basic idea is to combine probability raising with a correlation coefficient resembling a normalized version of

[5]Note that the assumption that noise is uniformly distributed among the events may be too simplistic: some events may be more robust, or easy to measure then others. In future work one could model more sophisticated noise distributions.

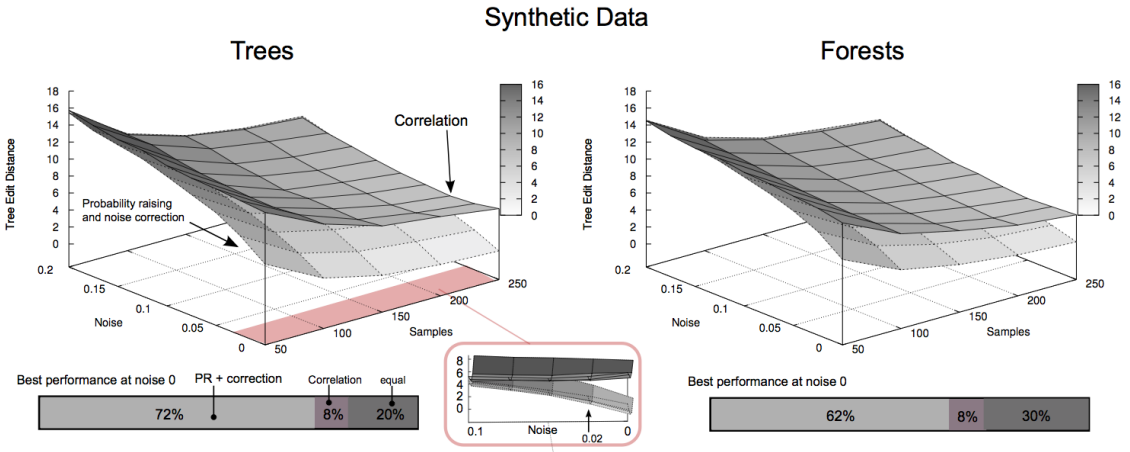

Figure 4.5: **Tree and forest reconstruction: comparison on synthetic data with noise correction.** Performance of the algorithm with noise correction. The specifications are as in Figure 4.4. As can be seen from the picture, the algorithm is much more robust to noise.

Desper's weight. The definition is as follows:

$$m_{i,j} = \frac{\dfrac{\mathcal{P}(j \mid i) - \mathcal{P}(j \mid \bar{i})}{\mathcal{P}(j \mid i) + \mathcal{P}(j \mid \bar{i})} + \dfrac{\mathcal{P}(i,j) - \mathcal{P}(i)\mathcal{P}(j)}{\mathcal{P}(i,j) + \mathcal{P}(i)\mathcal{P}(j)}}{2} \ .$$

Note that we include division by a factor of 2 merely to keep our weight functional between -1 and 1. With the new weight, the resistance to noise of our reconstruction algorithm improves significantly, as can be seen in Figure 4.6.

Resampling Another possible way of increasing robustness to noise is by using *bootstrapping*. Rather than using bootstrapping *after* the extraction to determine the confidence of the reconstructed tree as in Section 4.5.2, we can use bootstrapping *to determine* the progression model. Instead of extracting the tree structure from the whole dataset, we can use a simple resampling method, and determine the tree based on the outcomes of the sample estimates.

A simple algorithm that uses resampling would be almost exactly as Algorithm

1 but with weights $m_{i,j}$ and probabilities based on the average of N (say, a 1000) sample datasets obtained by bootstrapping [46]. These weights, and thus the resulting reconstructed trees, are less affected by small fluctuations due to noise.

A few points should be noted about this method. First, because of the resampling, the correctness of the algorithm no longer follows from Proposition 2. To guarantee the absence of cycles, we have to exclude them explicitly by requiring that at reconstruction phase, that only nodes with higher (average or actual) probability are considered as possible causes of each node. Second, the algorithm just described is the simplest method using resampling. Many straightforward extensions of this method can be imagined that do not use the *averages* of the determined weights alone, but also includes the information from *distributions* of the obtained weights. We plan to investigate these extensions in future work.

4.7 Back to CHAs

So far we have developed a method that can derive the underlying structure (in this case a tree or forest) of the progression from patient data. We conclude this chapter by briefly returning to our CHA models and discussing how close we have come to extracting CHAs and how some of the missing CHA parameters such as timing can be extracted from the data as well.

4.7.1 More general topologies

In this chapter, we have introduced basic ideas of how to reconstruct progression models using notions of causality. We have used trees as our basic topology, but one might argue that more general topologies are needed to more adequately describe disease progression. In particular, a major drawback of trees (and forests) is that they do not allow for one event to have several possible predecessors. In fact, the

CHA models introduced in Chapter 1, are of a more general type and allow for these *confluences*, and can be represented as *direct acyclic graphs* (DAGs).

The method by Desper et al. has been extended to include different topologies in various ways: Most notably, mixture models have been developed to include the possibility of extracting several trees that are accessed with a certain probability [9], and conjunctive Bayesian networks were developed by Beerenwinkel et al. [11, 54, 55]. In the latter, *constraints* (rather than possibilities) on the sequence of genetic events along the progression are extracted in the form of conjunctive Bayesian networks using maximum likelihood methods.

In future research we plan to extract more complex models using probability raising. In particular, we reconstruct direct acyclic graphs. While building on ideas presented in this chapter, this method will make use of more sophisticated statistical methods such as bootstrapping [43] and false discovery rate control [45] to *determine the edges of network*. We also plan to make use of common techniques from probabilistic causality to filter certain edges.

Different types of branchings It should be noted that in this chapter we assumed that transitions are *independent* events. That is, imagine a state a in the progression, with two outgoing edges to states b and c, then it is possible to acquire b *and/or* c after a. The type of branching thus reflects the interpretation of an inclusive or connective (OR). To the contrary, in CHA models, edges are not assumed to be independent, and each unit (patient, cancer cell, tumor subclone, etc.) progresses along a single path. That is, the branching represents an exculsive or connective (XOR). Both type of branchings are valuable for cancer progression models as they can be used to model different situations. For example, a strict XOR is necessary in case two events cannot occur together (e.g., due to *synthetic lethality* [81]), and an OR is useful to model independent events, that may occur individually or together. A similar distinction

can be made at nodes with two *incoming* edges: it may be that both predecessors are necessary for the event to occur (*AND*, this type of confluency is modeled by Beerenwinkel's conjunctive boolean networks [11]), or only one of the predecessors is necessary for the event to occur (*OR*).

In future work, when we extract DAGs, we plan to make use of common techniques from probability theory to classify edges, nodes, as well as branching and confluency points, by assigning them different connectives/logical formulas.

4.7.2 Timing

Timing of progression is of crucial importance to cancer patients, and an important parameter of the CHA framework.

Also, Suppes' notion of probabilistic causality does not only involve probability raising, but includes *timing* as well: The cause happens *before* the effect: $t_C < t_E$. The algorithm we presented above does respect timing measured by the rate of occurence of an event (that is, if $\mathcal{P}(a) > \mathcal{P}(b)$, then a will occur before b in the tree), but more sophisticated ways of incorporating and extracting timing information are possible, and will improve the accuracy and applicability of our CHA models.

As we have already mentioned in Chapter 1, timing parameters can be derived from longitudinal studies (see e.g., [71]). However, most cancer patient data is still static in nature (we have precise genetic information of many patients but at only one point in time). Therefore the problem of determining how long a transition takes is not an easy one. However, efforts have been made to extract timing information from clinical data and using stochastic simulations [80, 10]. As more time-course data on the progression of the disease becomes available, precise measures of timing can be more readily estimated as well.

In this section we will briefly outline several ways of measuring degree of progression, and timings of transitions, *in the absence of* explicit timing information, so that Suppes'

axiom of timing can be incorporated into our algorithm in a more sophisticated way.

There are several ways in which we can estimate how long a cancer has been progressing, (and from this information, how long various transitions take). Given a sample s, a few possible measures of progression timing, $\tau(s)$ of a sample are:

- $\tau(s)$ = (log of) the number of mutations or CNV segments (per MB) in the sample (as in [74]).
- $\tau(s)$ = the number of significant aberrations present that are considered for the progression reconstruction. If we consider n events, progression time ranges from 0 (no events), to n (all events).
- $\tau(s)$ = the stage of the cancer.
- $\tau(s)$ = the size of the tumor.
- $\tau(s)$ = determined by DNA methylation patterns [117].
- $\tau(s)$ = determined by survival data from the the sample.
- $\tau(s)$ = a combination of 1) number of aberrations, 2) stage, 3) degree of abberations, actual copy number 4) length of segments 5) existence of 'fire storms'[69], 6) methylation patterns [106], 7) survival data...
- *etcetera*

Note that instead of estimating the timing of a sample directly, we can also estimate the timing of *events* first, and then the timing of a sample from its events[6]. Each of these methods have their own advantages and disadvantages, and selecting a good measure is not straightforward. However, after a choice has been made, and once we

[6]Note that timing of *events* directly is respected by the algorithm: e.g., the more often an event occurs in the dataset, the earlier it happens in the tree.

have a measure for progression of a sample, we can extend this measure to compare *sets of samples*.

Given a function τ that assigns to each sample s a 'time' $\tau(s)$, say

$$\tau(s) = log(\text{ number of exonic mutations per MB in } s),$$

the measure used in [74], and two sets S, S', when does $\tau(S) \leq \tau(S')$ hold? Two possibilities include:

- $\tau(S) \leq \tau(S')$ if $avg\{\tau(s) \mid s \in S\} \leq avg\{\tau(s') \mid s' \in S'\}$
- $\tau(S) \leq \tau(S')$ if $min\{\tau(s) \mid s \in S\} \leq min\{\tau(s') \mid s' \in S'\}$

For each $i \in G$, let I denote the set of samples that have event i. Given that we have defined a notion of timing on a set of samples, we can now include timing in the definition of our weight, so as to require that Suppes first axiom is satisfied. For example, if we define

$$m^t_{i,j} = \begin{cases} m_{i,j} & \text{if } \tau(I) < \tau(J) \\ 0 & \text{otherwise} \end{cases},$$

then the resulting tree based on Algorithm 1 with weight m^t, will satisfy that an event i happens before j in the graph *only if* $\tau(I) < \tau(J)$.

4.7.3 The influence of drugs

The effect of drugs on the clocks can be estimated by comparing slopes of Kaplan Meier survival curves of patients treated with different drugs [44]. To be more precise, given a treatment option C, the clock delay ρ of the related transition can be measured as the inverse of the difference between slopes of control group and treated group. That is, if β_n and β_C are the slopes of the survival curves of control group and treated

group respectively, then the clock (x) at the relevant state (v) is affected by C at the rate ρ:

$$\rho(v, C, x) = \frac{\beta_C}{\beta_n} \quad .$$

Conclusions and Future Work

In this thesis we established a general formalism for *describing* cancer progression, without relying on any detailed mechanistic model of cancer pathways, or genetic aberrations. Our goal was to design a conceptually clear framework based on a realistic biological foundation. As a case study, we have used this model to describe cancer hallmarks and their dynamics.

Our framework (called CHA) is based on a *hybrid automaton*. Transitions from one discrete state of progression to the next are assumed to take certain durations of time, and the effects of drugs are modeled to *slow down* the progression clocks. Temporal statements about the progression as well as notions of timed therapies are formalized using our framework. We have also extended our model to include partial observability using the notion of belief sets, and tests are incorporated into the definition of a therapy as actions that reduce uncertainty about the current state of the progression.

The CHA framework not only enables us to formally describe cancer progression, but also to *manipulate* its evolution to satisfy certain therapeutic goals. We have studied automatic therapy design in Chapter 2, where we adopted a game theoretic formulation of the problem, and modeled a therapy as a *strategy* in the game against Nature. Building on existing tools and techniques from the controller synthesis literature, we showed that therapies can be automatically generated.

More specifically, we showed that discrete CHA control for CTL goals is EXPTIME complete (Theorem 9). This result has been extended to include partial observability

as well (Theorem 10). The therapy plans generated by our algorithm are *timed* and (are likely to) require *combinations* of drugs, so as to minimize the risk of drug resistance and recurrence due to heterogeneity.

The Theorem provides a complexity result for CHAs that is relevant for therapy design, but it is also meant to give our CHA frameworks a place within the existing literature on hybrid systems, so as to provide a starting point for studying these models and their algorithmic issues further.

In the second part of the thesis our focus shifted from *description* and *manipulation*, to *extraction* of cancer progression models from static real patient data. In Chapter 3 we studied copy number variation data and we presented evidence (from three TCGA datasets) that in cancer the lengths of the deleted or duplicated segments are not exponentially, but *power-law* distributed, and we suggest a generative model for this observation. This finding is important as many tools used to analyze cancer patient data rely on non-informative priors, and can be improved by incorporating this information into their null models. As an example, we have shown how this prior can be incorporated in a method that identifies cancer driver genes [76].

In Chapter 4 we provided a method for extracting tree progression models from cross-sectional patient data using mathematical notions of *probabilistic causality*. Using synthetic data we showed how this method outperforms the existing tree reconstruction algorithms. We have also provided some insight into how the various parameters of our CHA formalism (timing, the effect of drugs), can be estimated from the available cancer data, and incorporated into our algorithm.

Below we summarize several directions for future work.

CHA control As mentioned in Chapters 1 and 2, an important direction for future work is to focus more on to the algorithmic side of verifying cancer hallmark automata, automatically generating therapies (including cost minimization) and finding promising

drug targets. Even though discretized CHA control has been shown to be EXPTIME complete for CTL goals, simpler models (such as direct acyclic graphs), simpler goals (such as only reachability), or simpler strategies (such as only memory-free strategies) may be controlled more efficiently. Also, powerful tools can be imagined that use estimations, simulation-equivalence and abstractions.

This direction of research not only involves designing more efficient algorithms and abstractions, but also the implementation of user friendly software that allows users to both input and modify pregression models, as well as specify therapeutic goals and costs.

Cancer Driver Gene detection When presenting the method to find cancer driver genes in Chapter 3, we focused on the algorithmic/mathematical nature of the problem, and our data analysis remained largely preliminary in nature. As already discussed in Section 3.3, the tool itself can be improved in several ways. Here we mention four directions. First, incorporating a preferential attachment model in to the *segmenter* that analyzes the genomic data from each cell-type, will improve accuracy of the segmented data and thus our results. Second, we can use more accurate priors of the *distribution of breakpoints* that are known to occur in different cell-types. Third, several more sophisticated statistical tools for picking high-scoring intervals can be imagined. In particular, we would like to incorporate prior biological knowledge combined with multiple hypothesis testing, to make the final results more meaningful. Fourth, as we have discussed in Section 1.5, tumors consist of a heterogeneous population of cell-types and the cells of different types interact dynamically going through rapidly-changing cell-states. This heterogeneity requires more sophisticated oncogenomic analysis tools that generalize the mathematics described in Chapter 3 and Chapter 4 much further. In the case of driver gene detection, the null models can be further improved by including a mixture of distributions, with the parameters of the distribution fluctuating as

cancer progresses.

Progression Extraction In Chapter 4, we have presented a first exploration of the problem of extracting progression models, and the use of probabilistic causality as a main tool to do so. In the chapter, we already mentioned several extensions that we wish to develop in future work. First, we wish to extract *more general* CHA-like progression models, that allow for *confluencies*. In particular, we reconstruct direct acyclic graphs (DAGs), whereby we extend not only Desper's and our tree extraction method, but also other existing methods that are used to extract different types of models, such as the one developed in [54]. While building on ideas presented in this paper, to improve robustness to noise this method will make use of more sophisticated statistical methods such as *bootstrapping* [43] and *false discovery rate control* [45] to *derive the edges of network*. Also, we would like to use common techniques from the probabilistic causality literature as well as (modal) logic, to filter and classify edges and nodes (see Section 4.7). Another important extension to the model is to include *timing* into the model in a sophisticated manner. Finally, we plan to use our tool to extract new progression models from real cancer patient data, with the hope that, among others, new therapies can be generated and promising drug targets can be more adequately identified.

Appendices

Appendix A

Automatic Therapy Design for CHAs

A.1 Control of discrete automata

Control of safety and reachability games Formally, the controller synthesis problem for safety and reachability games on discrete game automata is defined as follows (Along the lines of [91]).

Let $L(A, F, \Box) \subseteq L(A)$ be the set of runs that always stay inside F, and $L(A, F, \Diamond) \subseteq L(A)$ the set of runs that eventually reach F.

Definition 25 (Controller synthesis for safety and reachability games)**.** *Given a game automaton A, a set of states F, the synthesis problem $Synth(A, F, \Box)$ (resp $Synth(A, F, \Diamond)$) is: Find a strategy π for the controller such that $L_\pi(A) \subseteq L(A, F, \Box)$ (resp $L_\pi(A) \subseteq L(A, F, \Diamond)$), or otherwise show that such a controller does not exist.*

More generally, if we let Ω be any acceptance condition and $L(A, \Omega)$ the set of runs such that the acceptance condition is satisfied, a more general controller synthesis problem can be defined as follows:

Definition 26 (Controller synthesis problem)**.** *Given a game automaton A, an acceptance condition Ω, the synthesis problem $Synth(A, \Omega)$ is: Find a strategy π for the*

controller such $L_\pi(A) \subseteq L(A, \Omega)$, or otherwise show that such a controller does not exist.

Now, let us focus on finding a strategy that controls the automaton for a set of safe states F. The basic idea is to iteratively search the space for a set of winning states F^*. The iterative search starts with $F_0 = F$, the set of safe states, and at each round i the set of states from which the controller cannot force the game to stay in F_i are deleted. This procedure converges to F^*. If $Q_0 \subseteq F^*$, the controller has a winning strategy. Formally, (adopted from [91])

Definition 27 (Controllable predecessors)**.** *Given a game automaton*

$$A = \langle Q, Q_0, \Sigma^1, \Sigma^2, E \rangle \ ,$$

the function $f : 2^Q \to 2^Q$ assigns a set of states to its controllable predecessors*:*

$$f(P) = \{q \mid \exists a \in \Sigma^1, \forall b \in \Sigma^2, \forall (q, a, b, q') \in E, q' \in P\}$$

Thus, $f(P)$ is the set of states from which the controller can force the automaton into P in one step. In the planning literature, the set of controllable predecessors of P is called the *strong backprojection* of P [87]. The algorithm for calculating the winning states works as a straightforward backward search: The strategy for every

---

$F_0 := F$
for $i = 0, 1, \ldots$ **repeat**
$F_{i+1} := F_i \cap f(F_i)$
until $F_{i+1} = F_i$
$F^* := F_i$

$q \in Q$ can be extracted as follows.

$$\pi(q) = \begin{cases} p(T^1(q)) & \text{if } q \notin F^* \\ p(\{a \in T^1(q) \mid \forall b \in T^2(q), (q, a, b, q') \in E, q' \in F^*\}) & \text{otherwise} \end{cases}$$

where $p(S)$ is a choice-function that picks one element from the set S, for example the first. Note that if $q \notin F^*$, the controller has no way of winning the game and thus it does not matter what action it chooses. It follows immediately by the construction of the algorithm that if a winning strategy exists, the algorithm will output one.

Remark 2. *Note that to improve complexity, the strategy space could be calculated iteratively together with the set F^*.*

For reachability the argument is very similar. Instead of iteratively deleting states that may lead *outside* of F, now states are iteratively added that lead *inside* F. In this case the iterative procedure can be described as:

$F_0 := \emptyset$
for $i = 0, 1, \ldots$ **repeat**
$f_{i+1} := F_i \cup f(F_i)$
until $F_{i+1} = F_i$
$F^* := F_i$

Strategy extraction is the same as for safety games.

Other goals that can be solved using very similar algorithms using only memory-free strategies like 'eventually remaining in a set F' and 'visiting states from F infinitely often', can be found in [91].

Adding costs to the control problem

In this section we extend the control problem to include costs of runs. This problem has mostly been studied in the *planning literature*, where the focus usually lies on

arriving at a goal state (reachability), while minimizing the cost [87]. Even though it is not the most commonly used framework in the planning literature, this planning problem can be expressed using a directed state transition graph, or game automaton, as introduced in the previous section.

Game automata with costs A game automaton A can be made into a weighted game automaton by associating with each edge a cost $C(q, a, b, q')$. It is the cost of taking transition $q \xrightarrow{(a,b)} q'$.

For finite runs, $\zeta = q_0 \xrightarrow{(a_0,b_0)} q_1 \xrightarrow{(a_1,b_1)} \dots \xrightarrow{(a_{n-1},b_{n-1})} q_n$, the cost of the run is simply the cost of the sum of the edges:

$$C(\zeta) = \sum_{i=0}^{n-1} C(q_i, a_i, b_i, q_{i+1}).$$

In the case of infinite runs, a discount factor $0 \leq d \leq 1$ can be applied to avoid infinite costs. However, if the goal is to reach a certain state, never reaching that state can be considered a failed run in which case infinite cost may be reasonable.

As mentioned before in the single-step case, this definition can be generalized to include vectors of costs, in which case a notion of Pareto-optimality can be used to pick out good outcomes.

In the planning literature the *cost of a strategy* (or plan, in their terminology) is defined to be the worst-case cost of all the possible runs of the controlled game. Formally,

$$C(\pi, q_0) = max\{C(\zeta) \mid \zeta \in L_\pi(A) \text{ starting at } q_0\}.$$

Since the strategy is assumed to guarantee reachability, this cost is also referred to as the *cost-to-go* (to a goal state). If the goal state is never reached, the cost-to-go is infinite. Note that this way of defining the cost of a strategy implicitly assumes a

worst-case analysis of the problem. Hence, by optimizing the cost of a strategy, the algorithms discussed in the next section optimize the worst-case outcome of a strategy.

Algorithms for computing strategies Two well-known and generally applicable algorithms for computing strategies for the reachability problem with costs are *Value Iteration* and *Policy Iteration*. Value iteration computes and updates the optimal cost-to-go for each state in the system iteratively. That is, for each state k a plan that minimizes the worst-case cost-to-go is determined based on the already calculated optimal plans of length $k-1$. The algorithm terminates when the worst-case cost-to-go no longer changes for any state.

In policy iteration, the state space of all strategies is searched iteratively. An initial policy (strategy) is picked and then two steps are performed iteratively: *value determination*, which calculates the cost-to-go for each state given that policy, and *policy improvement*, which updates the current strategy if any improvement is possible. The algorithm terminates when the strategy stabilizes. Both value iteration and policy iteration are guaranteed to terminate, but both have their shortcomings: Value iteration is inefficient in trying to compute the cost-to-go of states that already have optimal costs assigned to them, or states for which the goal has not yet been reached. Policy iteration does not have this problem but is only suitable for small state spaces. The reason for this limitation is that large system of linear equations must be solved at each step of the iteration.

In some cases, graph search algorithms like Dijkstra's algorithm can be extended to find optimal strategies efficiently. We refer the reader to [87] (chapter 10) for a detailed discussion of both methods as well as Dijkstra's graph search method.

A.2 Control of timed automata

Control of timed safety and reachability games

Definition 28 (Strategy)**.** *A strategy π_i for player i is a function $\pi_i : (Q \times V) \to \Sigma^i \cup \{e\}$ such that $\pi_i((q, v)) \in T^i(q, v)$ for each player i. The action e is the empty action of doing nothing or letting time pass.*

Given a strategy π for player 1*, a run*

$$\zeta = (q_0, v_0) \xrightarrow{\alpha_0} (q_1, v_1) \xrightarrow{\alpha_1} (q_2, v_2) \xrightarrow{\alpha_2} \dots$$

is said to conform *π if for every j the following holds:*

- *$\alpha_j = \delta$: for every $\delta' \leq \delta$, $\pi(q_i, v_i + \delta') = e$.*
- *$\alpha_j = c$: either $c \in \Sigma^1$ and $\pi(q_i, v_i) = c$ or $c \in \Sigma^2$.*

We let $L_\pi(A) \subseteq L(A)$ denote the set of runs of A that conform π.

Thus, a state transition (q, c, ϕ, Y, q') is taken when either the controller decides to play c, or Nature decides to play c. The transition is only possible when the current clock valuation satisfies ϕ. A delay transition is possible as long as both the controller and Nature decide not to perform an action and the state invariant is still satisfied.

Note that, as before, we can allow for memory by defining a strategy as $\pi_i : L(A)_f \to \Sigma^i \cup \{e\}$.

Definition 29. *Given a timed game automaton A, and a finite union of symbolic states $F \subseteq Q \times V$, the controller synthesis problem $Synth(A, F, \Box)$ (resp $Synth(A, F, \Diamond)$) is: Find a strategy π for the controller such that $L_\pi(A) \subseteq L(A, F, \Box)$ (resp $L_\pi(A) \subseteq L(A, F, \Diamond)$), or otherwise show that such a controller does not exist.*

The backward search algorithm is very similar to the untimed case, except that now, not only the possible consequences of the agent's actions are important to keep track of, but also the moments when the agent is not acting.

To define the notion of controllable predecessor as before, we need to take into account both the controlled and uncontrolled discrete predecessors, as well as *safe*

timed predecessors (from [25]). Given a set P, the set of i-controlled predecessors $f_i(P)$ is defined as follows.

$$f_i(P) = \{(q, v) \mid \exists c \in \Sigma^i, (q', v') \in P \text{ s.t. } (q, v) \xrightarrow{c} (q', v')\}.$$

These are the set of timed states from which player i can reach P by performing one action. Intuitively, the set of *safe* timed predecessors $f_\delta(P, U)$ of a set P with respect to a set U is the set of states (q, v) such that a state $(q', v') \in P$ can be reached by letting time elapse, and while waiting states from U are avoided. Formally,

$$f_\delta(P, U) = \{(q, v) \mid \exists \delta s.t. (q, v) \xrightarrow{\delta} (q, v + \delta), (q, v + \delta) \in P \text{ and } p_{[0,\delta]}(q, v) \subseteq \bar{U}\}$$

where $p_{[0,\delta]}(q, v) = \{(q, v + \delta') \mid \delta' \in [0, \delta]\}$.

Given these definitions, the *controllable predecessors* operator can be defined as follows:

$$f(P) = f_\delta(P \cup f_1(P), f_2(\bar{P}))$$

Remark 3. *Note that according to this definition, a situation where Nature is forced to make an action that helps the agent to win will not happen. A discrete action must be performed by the agent to reach a winning state, and uncontrollable actions can only spoil the game. This constraint has tacitly been assumed in most works on timed control e.g.[7, 8], but has been made explicit by Cassez in [26].*

To compute the set of winning states for safety and reachability we can apply the same fixed point algorithm as for the discrete case. However, because we are now iterating over an infinite set, it remains to be shown that the algorithm actually converges. In the discrete case, this was immediate since the iteration was over a finite

domain. In the timed case, so-called *regions* and *zones* guarantee convergence. The main idea is that the timed automaton can be translated into a *bisimilar* finite region or zone graph, and thus that there are actually a finite number of states that the algorithm iterates over. To be more explicit, if F is a finite union of states, then $f(F)$ is again a finite union of states, and effectively computable. The iterative process of finding F^*, will converge after finitely many steps, which guarantees decidability. Correctness follows from construction, and a detailed proof can be found in [91].

In the next section we will elaborate on the concept of regions and zones, as they form the heart of the analysis and decidability of verification and control of timed automata.

Regions, zones and symbolic states

The region automaton Because the clocks in timed automata take real values, the semantics of a timed automaton is an infinite transition system. However, this infinity can be dealt with by reasoning symbolically. The main ingredient for the analysis of timed (game) automata is the notion of *regions* [3]. Regions provide a finite partition of the state space such that within a given region, the behavior of the system is indistinguishable. That is, the states within a given region are *bisimilar*.

The definition of a region is such that all the states inside a region agree on the integral parts of all the clock values, and also on the ordering on the the fractional parts of all clock values. The integral parts of the clock values are needed to determine whether or not a clock constraint will be satisfied, and the ordering of the fractional parts is needed to decide which clock will change its integral value first [4].

For each $x \in X$, let c_x be the largest integer such that c_x appears as a clock constraint of x in the automaton. Then, two clock valuations v, v' are region equivalent $v \sim v'$ if the following conditions hold:

- For all $x \in X$, $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x) > c_x$ and $v'(x) > c_x$
- For all $x, y \in X$, with $v(x) \leq c_x$ and $v(y) \leq c_x$, $fract(v(x)) \leq fract(v(y))$ iff $fract(v'(x)) \leq fract(v'(y))$
- For all $x \in X$, with $v(x) \leq c_x$, $fract(v(x)) = 0$ iff $fract(v'(x)) = 0$

A *clock region* for an automaton A is an equivalence class of clock interpretations induced by $\sim$. Finiteness of the set of clock regions is guaranteed by the maximal clock constraints c_x. The region graph of A is a set of states (q, R) such that $q \in Q$ and R is a region. Transitions extend the transition of the original graph: that is, (q', R') can be reached from (q, R) by action a if (q', v') can be reached by action a from (q, v) for $v \in R$ and $v' \in R'$. Note that this definition for timed automata extends to timed game automata as well.

The region automaton preserves many properties such as reachability, safety and temporal formula and can thus be used as an abstraction to study the behavior of the automaton.

Zone graphs Even though the region automata provides a powerful tool for proving decidability of many verification and controller synthesis algorithms, it is impractical, as the number of regions grows exponentially with the number of clocks, and hence all algorithms are exponential time.

Luckily, another abstraction has been proposed that allows for polynomial-time algorithms [63]. This abstraction partitions the set of clock valuations into *zones*.

A *zone* is a subset of valuations $\mathbb{R}^X$ that is the solution set of a clock constraint. With $[[\phi]]$ we denote the zone of clock constraint ϕ. A *zone graph* of A is a set of states (q, Z) such that $q \in Q$ and Z is a zone. States of type are called *symbolic states* and the zone graph is sometimes referred to as the symbolic graph. Transitions extend the transition of the original graph: that is, (q', Z') can be reached from (q, Z)

by action a if $(q, c, \phi, Y, q') \in E$ and Z' is such that it is a set of timed successors of $(Z \cap [[\phi]])[Y := 0]$ satisfying the invariant of q'. Finiteness of the zone graph can be achieved using a technique called normalization [18], but the number of zones is still much larger than the number of regions. However, the reason for zone-based algorithms to be efficient in practice, is that algorithms don't need to explore the entire state space, but rather work *on-the-fly*. Moreover, zones can be represented using difference-bound matrices (DBMs) which allows for efficient operations on zones [22]. We refer the reader to [14] for a more thorough overview of regions, zones and their implementations.

Optimal timed, and weighted timed games

The algorithms discussed in Chapter 2 provide winning strategies for both safety and reachability games, by assigning at each moment of the game any action that keeps the game in the set of winning states. In the case of reachability games, this implies that no preference is given to actions that will guarantee to move the game to a goal state sooner rather than later. It seems natural, however, to require that the goal state is reached *as soon as possible*, or within a certain time frame. Games with such goals are called *optimal timed games*. In [6], optimal timed games are shown to be decidable, and a quantified generalization of the backward search algorithm is presented that guarantees that the goal set F is reached in minimal time. The approach that is used is similar in spirit to value and policy iteration and iterates over the value function which gives an upper bound on arrival times to the set F. As always in the case of timed systems, the use of zones are sufficient to ensure decidability. For details, see [6].

Of course, time is not the only measure of the cost of a run. It may be that some states or trajectories are undesirable or expensive for other reasons. Timed automata have been extended to include general cost information both on locations and edges

of the automata. This framework presents a generalization of optimal timed games, which are now weighted timed games where cost represent time elapsing. Weighted timed games have been and are extensively studied and have many applications (see [23] for an overview). To name an example, in [111], optimal timed games with general costs are studied and a 2EXPTIME algorithm is designed for synthesizing optimal controllers for acyclic games. This algorithm has further been improved (to single EXPTIME) in [5].

Theorem 12 (Alur et al. [5])**.** *The controller synthesis problem for optimal timed game automata can be solved in EXPTIME in the size of the game.*

The problem of optimal runs has also been studied on infinite runs using discount factors (see e.g. [48]). Also for stochastic timed games the expected time to reach a target has been studied (see e.g. [50]).

A.3 Proof of Theorem 9 (Discrete control of bounded CHAs)

Theorem (Discrete control of bounded CHAs)**.** *The controller synthesis problem of bounded discretized CHAs for CTL formulae can be solved in EXPTIME.*

Proof. First, we translate the bounded CHA H into an equivalent rectangular hybrid game automaton RG_H. Then we translate the RG_H into a *sampling control game* DRG_H in which the players can only make one move every time unit. Since this game can be finitely represented using a two-player bisimulation it follows from [79] that controller synthesis of discretized bounded CHAs with CTL goals is solvable in EXPTIME. □

We will start with a recalling a few necessary definitions

Definition 30 (CTL satisfaction on timed runs)**.** *Given a timed run*

$$S = (l_0, \mathsf{val}_0)(l_1, \mathsf{val}_0)(l_2, \mathsf{val}_2) \dots \ ,$$

and ϕ a CTL formula, S satisfies ϕ if the underlying path of discrete states $v_0 v_1 v_2 \dots$ satisfies ϕ (for more details on CTL satisfaction in discrete systems see e.g., [32]).

Definition 31 (CTL control)**.** *Given a timed system H, a strategy/therapy f and ϕ a CTL formula, f* controls *H for ϕ, if every run conform f satisfies ϕ.*

CHA H to rectangular hybrid game automaton RG_H

Definition 32. *(from [65]) Let X be a set on n real-valued variables. A* rectangular inequality *over X is a formula of the form $x_i \sim d$, where d is an integer and $\sim \in \{<, \leq, >, \geq\}$. A* rectangular predicate *over X is a conjunction of rectangular inequalities. The set of all rectangular predicates over X is denoted $Rect(X)$. Given a valuation $val : X \to \mathbb{R}$, and a rectangular predicate ϕ, we say that $val \vdash \phi$ if $\phi[X := val(X)]$ is true. Similarly $x_i \vdash \phi_i$ if $\phi_i[x_i := val(x_i)]$ is true.*

Note that both the clock constraints on edges and the invariants on states of the CHA are rectangular predicates.

In the following we will show that a CHA H can be translated into an equivalent rectangular game automaton RG_H. First, we recall the definition of a rectangular hybrid game. [1]

Definition 33. *(from [67]) A* Rectangular Hybrid Game *is a tuple*

$$R = (X, L, M_1, M_2, enabled_1, enabled_2, flow, E, jump, post) \ ,$$

[1] Actually, this definition is slightly different from the one presented in the body of Chapter 2. Both definitions are equivalent and used interchangeably in the literature. Here we chose to use this formulation, as it makes the translation from CHAs easier.

where

- X *is a set of* n *variables,*
- L *is a set of discrete states,*
- M_i *is the set of moves for player* i*.* $M_i^{time} = M_i \cup \{time\}$ *where time denotes a move that permits the passage of time.*
- $enabled_i : M_i^{time} \times L \to Rect(X)$*, which specifies for each move of player* i *and each location, the rectangle in which the move is enabled. Given a location* l*, the rectangle* $enabled_1(time, l) \cap enabled_2(time, l)$ *is said to be the invariant region of* l*, and is denoted* $inv(l)$*.*
- $flow : V \to Rect(\dot{X})$*, constraints the behavior of the first derivatives of the variables.*
- $E \subseteq (L \times M_1 \times M_2^{time} \times L) \cup (L \times M_1^{time} \times M_2 \times L)$ *a set of edges,*
- $jump : E \to 2^{\{1,\ldots,n\}}$ *maps each edge to the indices of those variables whose values may change when the discrete state proceeds along that edge.*
- $post : E \to Rect(X)$*, maps each edge to a* bounded *rectangle that contains the new continuous state when the discrete state proceeds along that edge.*

Definition 34 (Hybrid Game Structure)**.** *With a hybrid game* R*, the following game structure is associated:*

$$G_R = (Q_R, L, \langle\rangle, M_1^{time}, M_2^{time}, Enabled_1, Enabled_2, \delta) \ ,$$

where $Q_R = \{(l, \mathsf{val}) \in L \times \mathbb{R}^n \mid \mathsf{val} \in inv(l)\}$*,* $\langle (l, \mathsf{val}) \rangle = \{l\}$*,* $Enabled_i(a) = \{(l, \mathsf{val}) \in Q_R \mid x \in enabled_i(a, l)\}$*, and* $(l', \mathsf{val}') \in \delta((l, \mathsf{val}), a_1, a_2)$ *is either of the following two conditions is met:*

- Time step of duration t and slope s. *We have* $a_1 = a_2 = time$, $l' = l$, *and* $\mathsf{val}' = \mathsf{val} + t \times s$ *for some real vector* $s \in flow(l)$, *and some real* $t \geq 0$ *such that* $(\mathsf{val} + t' \times s) \in inv(l)$ *for all* $0 \leq t' \leq t$.

- Discrete step along edge e. *There extists an edge* $(l, a_1, a_2, l') \in E$ *such that* $(l, \mathsf{val}) \in Enabled(a_i)$ *for* $i = 1, 2$, *and* $\mathsf{val}'_k \in post(e)_k$ *for all* $k \in jump(e)$, *and* $\mathsf{val}'_k = \mathsf{val}_k$ *for all* $k \notin jump(e)$.

Runs and traces of R are inherited from the game structure G_R. A strategy for player i is a function $f_i : Q^+ \to 2^{Moves_i}$.

A CHA $H = (V, E, v_0, \ell, \rho)$ can be translated into a hybrid game RG as follows:

Given a set of drugs $\mathcal{D}$ and a CHA H with states V, we construct a hybrid game automaton RG_H in the following way: For each state $v \in V$ and each cocktail $C \in 2^{\mathcal{D}}$, the automaton R_{HG} contains a state (v, C) with the same clock invariants as v. For any edge between two states $v, v' \in V$, and $C \in 2^{\mathcal{D}}$, RG_H contains an edge between (v, C) and (v', C) controlled by player 2 (nature) with the same clock constraints and resets as on the CHA edge. In addition, there are controllable directed edges from (v, C) to (v, C') for each v, C and C'. These edges represent changes of therapy controlled by player 1, and have no clock constraints or resets. At a state (v, C), the rate of each clock $x \in X$ is fixed, given by $\rho(v, C, x)$.

Thus therapists can change a therapy from C to C' at state v by moving from (v, C) to (v, C'), and cancer picks an accessible new CHA state from the available $((v, C)(v', C))$ transitions. Formally, the translation is as follows:

Definition 35. *Given a timed CHA* $H = (V, E, v_0, \ell, \rho)$ *we define a (CHA-based) rectangular game automaton*

$$RG_H = (X, L, M_1, M_2, enabled_1, enabled_2, flow, E, jump, post)$$

as follows.

- $L = \{(v, C) \mid v \in V, C \in 2^{\mathcal{D}}\}$,
- $M_1 = 2^{\mathcal{D}}$, $M_2 = V$,
- $enabled_1(C', (v, C)) = \ell(v)$ $enabled_2(v', (v, C)) = \phi$ *such that* $(v, \phi, v') \in E$.
- $flow((v, C))(x) = \rho(v, C, x)$.
- $$\begin{aligned} E \;=\; & \{((v, C), v', time, (v', C)) \mid (v, \phi, v') \in E \text{ for some } \phi\} \\ \cup\; & \{((v, C), time, C', (v, C')) \mid \forall C \neq C' \in 2^{\mathcal{D}}\} \\ \cup\; & \{((v, C), v', C', (v', C')) \mid (v, \phi, v') \in E \text{ for some } \phi \forall C \neq C' \in 2^{\mathcal{D}}\} \end{aligned}$$
- $jump(((v, C), (v, C'))) = \emptyset$, $jump(((v, C), (v', C))) = \{1, \ldots, n\}$,
- $post(((v, C), (v, C')))_i = [0, m]_i$, *where* m *denotes the upper bound on the clocks, and*
  $post(((v, C), (v, C'))) = [X := 0]$, *all clock values are reset to* 0.

By construction, the resulting automaton is a rectangular hybrid game automaton. Moreover, the automaton RG_H exactly captures the CHA H. To see how, note the following properties:

- The fact that the therapist can change his therapy at any time, is captured by the fact that every action C' is enabled at every state as long as the invariant is satisfied: $enabled_1(C', (v, C)) = \ell(v)$
- The cancer can move to a new state of the progression, as long as the edge constraint is satisfied: $enabled_2(v', (v, C)) = \phi$ such that $(v, \phi, v') \in E$, and clocks are reset: $post(((v, C), (v, C'))) = [X := 0]$ and $jump(((v, C), (v', C))) = \{1, \ldots, n\}$.

- The clock dynamics is controlled by the drugs administered: $flow((v, C))(x) = \rho(v, C, x)$, and changing drugs does not change the clock values : $jump(((v, C), (v, C'))) = \emptyset$.
- Therapies and strategies are both functions from partial runs to actions.

To be more precise, the two automata H and RG_H are *trace equivalent* in the following sense:

Proposition 4. *For every run* $(v_0, \mathsf{val}_0) \to_0 (v_1, \mathsf{val}_1) \to_1 (v_2, \mathsf{val}_2) \to_2 \ldots$ *in* H *there exist an equivalent run* $(l_0, \mathsf{val}_0) \to_0 (l_1, \mathsf{val}_1) \to_1 (l_2, \mathsf{val}_2) \to_2 \ldots$ *of* RG_H *such that for every* i*:* $l_i = (v_i, C_i)$ *and if it is the case that* $(v_i, \mathsf{val}_i) \to_i (v_{i+1}, \mathsf{val}_{i+1})$ *is a delay transition* $(v_i, \mathsf{val}_i) \xrightarrow{\delta, C} (v_{i+1}, \mathsf{val}_{i+1})$*, then* $C_i = C$*.*

Proof. Follows immediately from the construction of RG_H and the preceding remarks. □

The reverse holds as well: for every run of RG_H there exists an equivalent run of H.

From the above, it follows that every therapy π for H has an equivalent induced strategy $f_{\pi,1}$ for player 1 for the game RG_H, and vice versa.

It also follows that the two automata are equivalent with respect to CTL control:

Corollary 1. *Given a CTL formula* ϕ*,* π *controls* H *for* ϕ *iff* $f_{\pi,1}$ *controls* RG_H *for* ϕ*.*

Game bisimulation

We define the notion of a game bisimulation as in [67]:

Definition 36 (Game bisimulation)**.** *Given a game structure*

$$G = (W, \Pi, M_1, M_2, enabled_1, enabled_2, \delta) \ ,$$

a binary relation $\cong \subseteq Q \times Q$ *is a* game bisimulation *if* $p \cong q$ *implies that the following three conditions hold.*

1. $\langle p \rangle = \langle q \rangle$
2. $M_1(p) = M_1(q)$ *and* $M_2(p) = M_2(q)$
3. - $\forall m_1 \in M_1, m_2 \in M_2, p' \in \delta(p, m_1, m_2) \exists q' \in \delta(q, m_1, m_2)$ *and* $p' \cong q'$.
   - $\forall m_1 \in M_1, m_2 \in M_2, q' \in \delta(q, m_1, m_2) \exists p' \in \delta(p, m_1, m_2)$ *and* $p' \cong q'$.

Proposition 5. *Consider two states* p *and* q *of a game structure* G*. If* p *and* q *are game bisimilar then for every CTL formula* ϕ*, player* 1 *can control* p *for* ϕ *iff player* 1 *can control* q *for* ϕ

Proof. By induction on the size of the formula ϕ □

The largest bisimulation on S is called the *bisimilarity* on S and the *bisimilarity quotient* on S is the transition system induced by the bisimilarity.

Hybrid game RG_H to discretized hybrid game DRG_H

Now we will extend the discretization method as given in [65] for rectangular automata to hybrid games.

We define a *sampling control game* DRG_H in which the players can make at most one move every time unit.

To show that controller synthesis for sampling control games can be solved in exponential time, we reduce the sampling control problem to discrete time control problem. Towards this goal, we add a new variable x_{n+1} to make sure that every discrete transition, a move by the cancer or the therapist, is followed by a flow transition of exactly one time unit. We define the initial value of x_{n+1} as 1. Also, at every state, the invariant of x_{n+1} is exactly 1 ($inv((v, C), x_{n+1}) = 1$) and $flow((v, C), x_{n+1}) = 1$ for all states. Thus, no matter which current cocktail is being administered, the

clock x_{n+1} always runs at a rate of 1, and cannot exceed 1. Finally, for every edge e, $jump(e) = jump_{RG_H}(e) \cup \{n+1\}$. It follows from the construction that in this automaton moves by the cancer and therapist are followed by a flow transition of duration 1.

Once the one time unit has passed, we would like both the controller and the cancer to have the opportunity to wait another time unit, as long as the invariant is satisfied. For this purpose we include a reflexive edge for each state (v, C). On this edge, only the clock x_{n+1} is reset: $jump(e) = \{n+1\}$ and the actions are only enabled if the invariant will be satisfied for another time unit.

Note that the time unit needs to be chosen such that the automaton is big enough: in the original automaton it may not be possible for a move to be allowed only in an interval smaller than one time unit. Formally, there cannot be three states $(q, \mathsf{val}), (q, \mathsf{val}'), (q, \mathsf{val}'')$ and an action a and player i such that $(q, \mathsf{val}) \xrightarrow{\delta} (q, \mathsf{val}')$ and $(q, \mathsf{val}') \xrightarrow{1-\delta} (q, \mathsf{val}'')$ for some $\delta \in (0, 1)$ (where 1 is the sampling interval), and $(q, \mathsf{val}), (q, \mathsf{val}') \notin Enabled_i(a)$ while $(q, \mathsf{val}'') \in Enabled_i(a)$.

To be able to solve the controller synthesis problem, we need a finite representation of the game DRG_H. We proceed by defining the game bisimilarity as in [65].

First, we define an equivalence relation $\approx_n^m$ on $\mathbb{R}^n$ as follows. $\mathbf{y} \approx_n^m \mathbf{z}$ iff for all $0 \leq i \leq n$, $\lfloor y_i \rfloor = \lfloor z_i \rfloor$ and $\lceil y_i \rceil = \lceil z_i \rceil$, or both $y_i, z_i > m$. Now, given two states $((v, C), \mathsf{val})$ and $((v', C'), \mathsf{val}')$ we define $((v, C), \mathsf{val}) \cong_{DRG_H} ((v', C'), \mathsf{val}')$ if $v = v'$, $C = C'$ and $\mathsf{val} \approx_n^m \mathsf{val}'$.

Proposition 6. *$\cong_{DRG_H}$ is a bisimulation (and in fact the bisimilarity) relation on the discretized hybrid game.*

The bisimilarity quotient is finite in size. To be precise, the bisimilarity quotient has no more than $(|V| \times 2^{|\mathcal{D}|}) \times (2m+2)^{(n+1)}$ states, where $(|V| \times 2^{|\mathcal{D}|})$ is the number of states of the game automaton RG_H.

Corollary 2. *The controller synthesis problem of bounded discretized CHAs for CTL formulae can be solved in EXPTIME.*

Proof. By propositions 5 and 6 it follows that we can perform supervisory control on the bisimilarity quotient. This is a finite graph. Supervisory control for CTL formulae in finite automata with both controllable and non-controllable actions is shown to be EXPTIME complete and algorithms exist [79].

□

Appendix B

Driver Gene Detection

B.1 Segment-length distribution

Let AVG_C and STD_C (resp AVG_N and STD_N) denote the average segment-length and the standard deviation of all segments derived from tumor (resp blood-derived normal) cells.

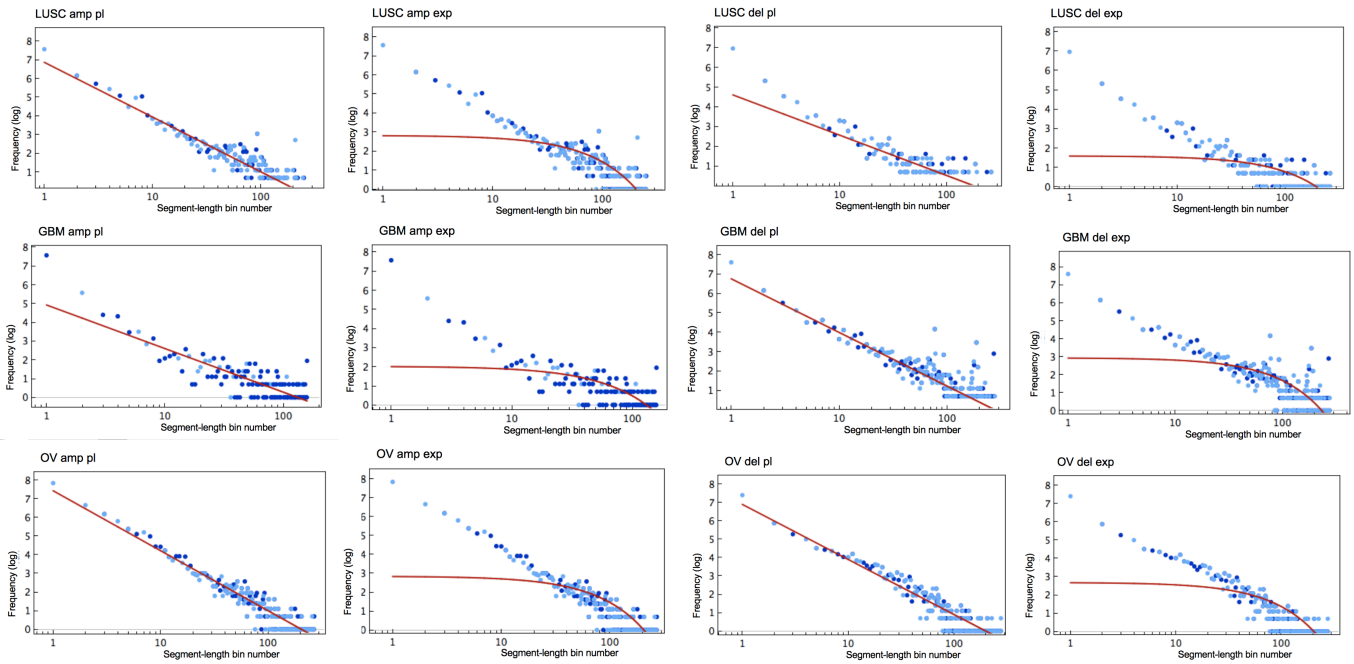

Figure B.1: :LUSC, OV, GBM. The thresholds are $AVG_C \pm 2STD_C$.

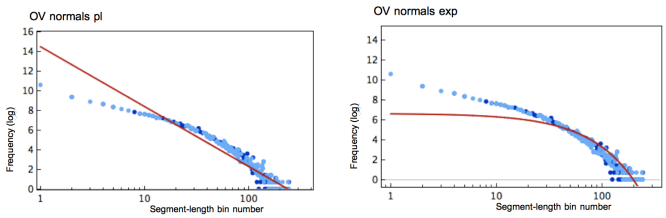

Figure B.2: That is, all segments with segment values in $[AVG_C - 2STD_C, AVG_C + 2STD_C]$.

	best exponential fit		best power-law fit	
	function	R^2	function	R^2
LUSC Nrm	$e^{-0.020}$	0.70	$x^{-1.30}$	0.57
OV Nrm	$e^{-0.033}$	0.89	$x^{-2.65}$	0.92
GBM Nrm	$e^{-0.016}$	0.50	$x^{-1.09}$	0.43

Table B.1:

Treshold	OV									
	AMP					DEL				
	th	PL		EXP		th	PL		EXP	
		α	R^2	λ	R^2		α	R^2	λ	R^2
± 1.0	1.00	1.41	0.90	0.024	0.64	-1.00	1.20	0.85	0.015	0.52
$AVG_C \pm 2STD_C$	0.76	1.39	0.91	0.014	0.67	-0.87	1.30	0.91	0.013	0.64
$AVG_C \pm 1.5STD_C$	0.59	1.79	0.93	0.031	0.81	-0.66	1.82	0.93	0.028	0.75
$AVG_C \pm 1STD_C$	0.36	1.94	0.93	0.030	0.84	-0.46	1.99	0.90	0.033	0.85
$AVG_N \pm 5STD_N$	0.21	2.11	0.88	0.0251	0.85	-0.27	2.21	0.85	0.0343	0.87
$AVG_N \pm 3STD_N$	0.11	1.85	0.85	0.0255	0.87	-0.18	1.99	0.86	0.0343	0.89
$AVG_N \pm 2STD_N$	0.06	1.79	0.83	0.025	0.89	-0.13	1.96	0.86	0.034	0.90
0.0	0.00	1.79	0.83	0.025	0.89	0.00	1.96	0.86	0.034	0.90

Table B.2: Using the OV dataset, this table shows how different tresholds influence the power-law and exponential fits.

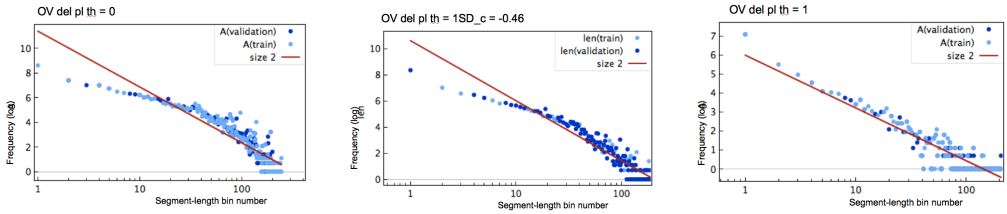

Figure B.3: OV deletions segment length distributions for different thresholds: 0, $AVG_C \pm 1SD_C = -0.46$ and -1.

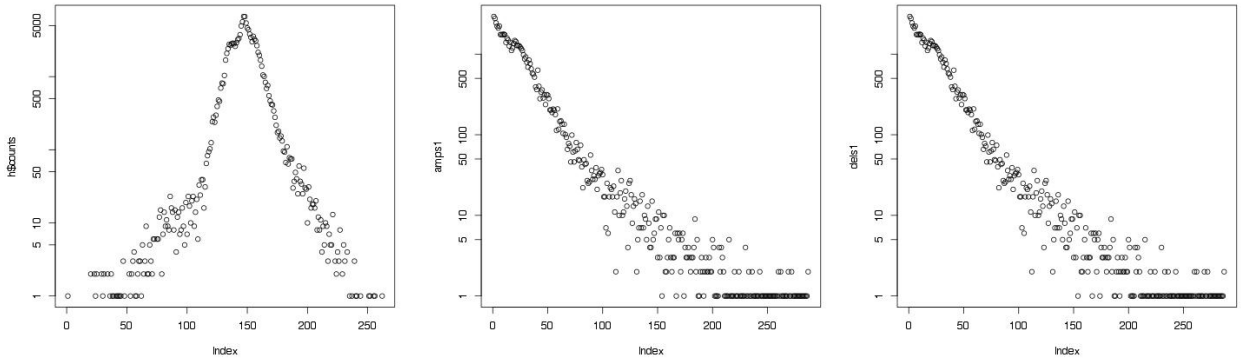

Figure B.4: Distribution of segment values of all segments (left), all positive segment values (middle) and negative segment values (right), on a log-log scale

	best exponential fit		best power-law fit	
	function	R^2	function	R^2
LUSC Amp	$e^{-0.27x}$	0.96	$x^{-1.26}$	0.97
LUSC Del	$e^{-0.48x}$	0.94	$x^{-1.58}$	0.99
OV Amp	$e^{-0.26x}$	0.96	$x^{-1.50}$	0.99
OV Del	$e^{-0.30x}$	0.91	$x^{-1.22}$	0.99
GBM Amp	$e^{-1.51x}$	1.00	$x^{-2.71}$	1.00
GBM Del	$e^{-0.41x}$	0.91	$x^{-1.39}$	0.97

Table B.3: CNV segment-length distribution exponential and power-law fits for non-log data.

	best exponential fit R^2	best power-law fit R^2	best log-normal fit R^2
LUSC Amp	0.65	0.86	0.82
LUSC Del	0.45	0.79	0.77
OV Amp	0.67	0.91	0.77
OV Del	0.64	0.91	0.86
GBM Amp	0.39	0.71	0.71
GBM Del	0.60	0.78	0.76

Table B.4: Exponential ($ce^{-\lambda x}$), power-law ($cx^{-\alpha}$) and log-normal ($\frac{1}{x\sqrt{2\pi\sigma^2}}e^{-\frac{(ln(x)-\sigma)^2}{2\sigma^2}}$) fits.

B.2 Proof of proposition 1 (Power-law null model)

The model We assume that, at any genomic location, a breakpoint (starting point) may occur as a Poisson process at a rate of $\mu \geq 0$. We consider two different μ's: one for amplifications μ_{AMP} and one for deletions μ_{DEL}, but we drop the subscript when no confusion arises. Segments are modeled as vectors: starting at a breakpoint x and moving left (or right) with probability $\frac{1}{2}$. The length t of each segment is distributed according to a powerlaw distribution: $t^{-\alpha}$, with $1 \leq \alpha \leq 2$. Let ϵ be the constant that represents the shortest length an interval could possibly have.

Proposition. *Assuming that segment lengths are power-law distributed :*

1. *The probability that an interval $I = [a, b]$ is lost is as follows:*

$$\begin{aligned}\mathcal{P}([a,b]\ \textit{lost}) &= 1 - e^{-\mu(b-a)} \times \\ & \quad e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]} \times \\ & \quad e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-b)^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]};\end{aligned}$$

2. The probability that an interval $I = [a, b]$ is gained is as follows:

$$\begin{aligned}\mathcal{P}([a,b]\ \textit{gained}) &= 1 - e^{-\mu\frac{\epsilon^{\alpha-1}}{2}}\left[\frac{b^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right] \times \\ & \quad e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-a)^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right]}.\end{aligned}$$

Proof. 1. We want to calculate the probability that the interval $[a, b]$ is 'lost'. This is the probability that there exists a deleted interval I that intersects with $[a, b]$:

$$\mathcal{P}([a,b] \text{ lost}) = \mathcal{P}(\exists I : I \cap [a,b] \neq \emptyset \text{ and } I \text{ is deleted}).$$

Instead, we compute $\mathcal{P}([a, b]$ is NOT lost) by computing:

($\mathcal{P}_1$) No deletion starting in the interval $[a, b]$,

($\mathcal{P}_2$) The probability that each deletion starting in $[0, a]$ does not overlap $[a, b]$, and

($\mathcal{P}_3$) The probability that each deletion starting in $[b, G]$ does not overlap $[a, b]$.

It follows that $\mathcal{P}([a, b]$ is NOT lost) $= \mathcal{P}_1 \times \mathcal{P}_2 \times \mathcal{P}_3$. And,

$$\mathcal{P}([a,b] \text{ lost}) = 1 - \mathcal{P}([a,b] \text{ is NOT lost}).$$

($\mathcal{P}_1$) $\mathcal{P}$(no deletion starts in $[a, b]$) $= e^{-\mu(b-a)}$. This follows immediately from the assumption that breakpoints are generated by a poisson process. Note that we drop the subscript DEL in μ_{DEL}.

($\mathcal{P}_2$) $\mathcal{P}$(each interval starting in $[0, a]$ does not overlap with $[a, b]$) can be broken down as the following infinite sum:

$\mathcal{P}$(each interval starting in $[0, a]$ does not overlap with $[a, b]$) =

$$\begin{aligned} & \mathcal{P}(\text{no deletions start in } [0,a]) \\ + \;& \mathcal{P}(\text{1 deletion starts in } [0,a]) \times \mathcal{P}(\text{the deleted interval } \cap [a,b] = \emptyset) \\ + \;& \mathcal{P}(\text{2 deletions start in } [0,a]) \times \mathcal{P}(\text{both deleted intervals } \cap [a,b] = \emptyset) \\ + \;& \ldots \end{aligned}$$

By the assumption that breakpoints are generated as a poisson process, the probability $\mathcal{P}(n \text{ deletions start in } [0, a]) = (\mu a)^n \frac{e^{-\mu a}}{n!}$ for each n. The probability $\mathcal{P}(\text{1 deleted interval } \cap [a, b] = \emptyset)$ can be computed as follows. From our model it follows that

$\mathcal{P}(\text{ deleted interval } \cap [a, b] = \emptyset \mid \text{1 deletion starts in } [0, a])$ is the probability that each deletion starting at x in the interval $[0, a]$ does not reach all the way to a:

$\mathcal{P}(\text{deleted interval } \cap [a, b] = \emptyset \mid \text{1 deletion starts in } [0, a])$

$$= \frac{1}{2} + \frac{1}{2}\frac{1}{a}\left(\int_0^{a-\epsilon}\int_\epsilon^{a-x} c_{\epsilon,G}\, t^{-\alpha} dt dx + \epsilon\right),$$

where:

- The constant $c_{\epsilon,G}$ depends on the length ϵ and G and is computed below.
- The $\frac{1}{2}$'s are to take into account the possibility that the deletion moves left instead of right.
- The last term $+\epsilon$ takes into account the possibility that the starting point of the deleted interval is in $[a - \epsilon, a]$.

This can be solved as follows:

$$\mathcal{P}(\text{deleted interval } \cap [a,b] = \emptyset \mid 1 \text{ deletion starts in } [0,a])$$

$$\begin{aligned}
&= \tfrac{1}{2} + \tfrac{1}{2}\tfrac{1}{a}\left(\int_0^{a-\epsilon}\int_\epsilon^{a-x} c_{\epsilon,G} t^{-\alpha} dt dx + \epsilon\right) \\
&= \tfrac{1}{2} + \tfrac{1}{2a}\left(\tfrac{c_{\epsilon,G}}{1-\alpha}\int_0^{a-\epsilon}\left[(a-x)^{1-\alpha} - \epsilon^{1-\alpha}\right] dx + \epsilon\right) \\
&= \tfrac{1}{2} + \tfrac{1}{2a}\left(\tfrac{c_{\epsilon,G}}{1-\alpha}\left[\left(-\tfrac{(a-(a-\epsilon))^{2-\alpha}}{(2-\alpha)} - \epsilon^{1-\alpha}(a-\epsilon)\right) + \left(\tfrac{a^{2-\alpha}}{2-\alpha}\right)\right] + \epsilon\right) \\
&= \tfrac{1}{2} + \tfrac{1}{2a}\left(\tfrac{c_{\epsilon,G}}{1-\alpha}\left[\tfrac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)\right] + \epsilon\right) \\
&= \tfrac{1}{2} + \tfrac{1}{2a}\left(\tfrac{c_{\epsilon,G}}{(1-\alpha)}\left[\tfrac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)\right] + \epsilon\right)
\end{aligned}$$

There are a few things to note about this derivation

- We ignore the integration constants, as they cancel each other out.
- Since $\alpha \geq 1$, and $c_{\epsilon,G}, a \geq 0$, the term $\frac{c_{\epsilon,G}}{2a(1-\alpha)}$ is negative. We thus need to show that $\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)\right]$ is always negative to obtain a positive probability. This follows from the mean-value theorem. Namely, for any function f that is concave and increasing the following holds:

  $$f(x) - f(x-\delta) \leq \delta f'(x-\delta)$$

  the function $f(x) = \frac{x^{2-\alpha}}{2-\alpha}$ is concave and increasing with $f'(x) = x^{1-\alpha}$. If we let $x = a$ and $\delta = a - \epsilon$, then $x - \delta = \epsilon$ and we have

  $$\frac{a^{2-\alpha}}{2-\alpha} - \frac{\epsilon^{2-\alpha}}{2-\alpha} \leq (a-\epsilon)\epsilon^{1-\alpha}$$

  from which it follows that $\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)$ is negative.

The normalizing constant $c_{\epsilon,G}$ can be computed as follows. It has to be such that

$$\int_\epsilon^G c_{\epsilon,G} t^{-\alpha} dt = 1.$$

It follows that

$$\begin{aligned} c_{\epsilon,G} &= (\int_{\epsilon}^{G} t^{-\alpha} dt)^{-1} \\ &= \left(\frac{G^{1-\alpha}}{1-\alpha} - \frac{\epsilon^{1-\alpha}}{1-\alpha}\right)^{-1} \\ &= \frac{1-\alpha}{G^{1-\alpha}-\epsilon^{1-\alpha}} \end{aligned}$$

Since $\alpha > 1$ and $G >> \epsilon$ this approaches

$$\approx \frac{\alpha-1}{\epsilon^{1-\alpha}}$$

Using $c_{\epsilon,G} = \frac{\alpha-1}{\epsilon^{1-\alpha}}$, we can simplify

$\mathcal{P}(\text{ deleted interval } \cap [a,b] = \emptyset \mid 1 \text{ deletion starts in } [0,a])$ as follows:

$$\begin{aligned} &= \frac{1}{2} + \frac{1}{2a}\left(\frac{\alpha-1}{\epsilon^{1-\alpha}}\frac{1}{(1-\alpha)}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)\right] + \epsilon\right) \\ &= \frac{1}{2} + \frac{1}{2a}\left(-\frac{1}{\epsilon^{1-\alpha}}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha} - \epsilon^{1-\alpha}(a-\epsilon)\right] + \epsilon\right) \\ &= \frac{1}{2} + \frac{1}{2a}\left(-\epsilon^{\alpha-1}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right] + (a-\epsilon) + \epsilon\right) \\ &= 1 - \frac{\epsilon^{\alpha-1}}{2a}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right] \end{aligned}$$

Since deletions are assumed to be independent events that can overlap it follows that

$$\mathcal{P}(n \text{ deleted intervals } \cap [a,b] = \emptyset \mid n \text{ deletions starts in } [0,a]) =$$
$$\mathcal{P}(\text{deleted interval } \cap [a,b] = \emptyset \mid 1 \text{ deletion starts in } [0,a])^n)$$

Hence, we get the following series:

$$\mathcal{P}_2 = e^{-\mu a} + (\mu a)^1 \frac{e^{-\mu a}}{1!}(1-w) + (\mu a)^2 \frac{e^{-\mu a}}{2!}(1-w)^2 + \ldots$$

with $w = \frac{\epsilon^{\alpha-1}}{2a}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]$.

It follows that

$$\mathcal{P}_2 = e^{-\mu a \frac{\epsilon^{\alpha-1}}{2a}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]}$$

which can be simplified to

$$\mathcal{P}_2 = e^{-\mu \frac{\epsilon^{\alpha-1}}{2}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]}$$

($\mathcal{P}_3$) $\mathcal{P}$(each interval starting in $[b, G]$ does not overlap with $[a, b]$) is computed in the same way as $\mathcal{P}_2$, but now starting at $x \in [b, G]$ and moving left. In this case

$$\mathcal{P}(\text{ deleted interval } \cap [a, b] = \emptyset \mid 1 \text{ deletion starts in } [b, G])$$

$$\begin{aligned} &= \tfrac{1}{2} + \tfrac{1}{2}\tfrac{1}{G-b}\left(\int_{b+\epsilon}^{G}\int_{\epsilon}^{x-b} c_{\epsilon,G}t^{-\alpha}dtdx + \epsilon\right) \\ &= 1 - \tfrac{\epsilon^{\alpha-1}}{2(G-b)}\left[\frac{(G-b)^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right] \end{aligned}$$

and we obtain

$$\mathcal{P}_3 = e^{-\mu \frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-b)^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]}$$

It follows that

$$\begin{aligned} \mathcal{P}([a, b] \text{ lost}) &= 1 - e^{-\mu(b-a)} \times \\ &\quad e^{-\mu \frac{\epsilon^{\alpha-1}}{2}\left[\frac{a^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]} \times \\ &\quad e^{-\mu \frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-b)^{2-\alpha}-\epsilon^{2-\alpha}}{2-\alpha}\right]} \end{aligned}$$

2. In an analogue fashion we calculate the probability that the interval $[a, b]$ is‘ gained’. This is the probability that there exists a deleted interval I that includes $[a, b]$:

$$\mathcal{P}([a, b] \text{ gained}) = \mathcal{P}(\exists I : [a, b] \subseteq I \text{ and } I \text{ is amplified})$$

We compute $\mathcal{P}([a, b]$ is NOT gained) by computing:

($\mathcal{P}_1$) The probability that each interval starting in $[0, a]$ does not include $[a, b]$, and

($\mathcal{P}_2$) The probability that interval starting in $[b, G]$ does not include $[a, b]$.

The computation of $\mathcal{P}_1$ (and $\mathcal{P}_2$) is analogues to that of deletions, except for the fact that we have to intergrate over all intervals reaching up to b (down to a). In the case of $\mathcal{P}_1$ we solve

$$\mathcal{P}([a,b] \subseteq \text{ amplified interval } \mid 1 \text{ amplification starts in } [0,a])$$

$$\begin{aligned} &= \tfrac{1}{2} + \tfrac{1}{2}\tfrac{1}{a}\left(\int_0^{a-\epsilon}\int_\epsilon^{b-x} c_{\epsilon,G}t^{-\alpha}dtdx + \epsilon\right) \\ &= 1 - \tfrac{\epsilon^{\alpha-1}}{2a}\left[\tfrac{b^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right] \end{aligned}$$

and in the case of $\mathcal{P}_2$

$$\mathcal{P}([a,b] \subseteq \text{ amplified interval } \mid 1 \text{ amplification starts in } [b,G])$$

$$\begin{aligned} &= \tfrac{1}{2} + \tfrac{1}{2}\tfrac{1}{G-b}\left(\int_{b+\epsilon}^{G}\int_\epsilon^{x-a} c_{\epsilon,G}t^{-\alpha}dtdx + \epsilon\right) \\ &= 1 - \tfrac{\epsilon^{\alpha-1}}{2(G-b)}\left[\tfrac{(G-a)^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right] \end{aligned}$$

We obtain:

$$\begin{aligned} \mathcal{P}([a,b] \text{ gained}) &= 1 - e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{b^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right]} \times \\ & \quad e^{-\mu\frac{\epsilon^{\alpha-1}}{2}\left[\frac{(G-a)^{2-\alpha}-(b-a+\epsilon)^{2-\alpha}}{2-\alpha}\right]} \end{aligned}$$

□

B.3 Detecting driver genes

Cancer	amplifications	deletions	normals
OV (337)	13416	10237	82633
LUSC (201)	3637	1832	46215
GBM (299)	2131	3959	41458

Table B.5: Number of deleted and amplified segment for three TCGA data sets using a threshold of $AVG_C \pm 2STD_C$.

Cancer	Gene	Function	Location	Power-law	Exponential
OV	BRCA1	TSG	17:(41196312..41277500)	no	no
	BRCA2	TSG	13:(32889617..32973809)	no	no
	ERBB2	OG	17:(37844393..37884915)	no	no
	K-ras	OG	12: (25358180..25403854)	yes (25289555..25421243)	yes (25177510..26726740)
	AKT2	OG	19: (40736224..40791302)	no	no
	PIK3CA	OG	3: (178866311..178952500)	no	no
	c-MYC	OG	8: (128748315..128753680)	next (128797789..128989029)	no
	p53	TSG	17: (7571720..7590868)	no	no
LUSC	CDKN2A	TSG	9: (21967751..21994490)	yes (18947155..28723296)	yes (21983401..21993651)
	FGFR1	OG	8:(38268656..38326352)	yes (38303346..38369274)	no
	PDGFR	OG	4: (55095264..55164412)	no	no
	SOX2	OG	3: (34650005..34652461)	no	no
	WHSC1L1	OG	8: (38132560..38239790)	next (38303346..38369274)	no
GBM	EGFR	ONCG	7: (55086725..55275031)	next (55049021..55065490)	next (54998411..55043660)
	MDM2	ONCG	12: (69201971..69239320)	no	no
	PDGFR	ONCG	5: (149493402..149535422)?	no	no
	CDK4	ONCG	12: (58141510..58146230)	no	no
	Rb	TSG	13: (48877883..49056026)	no	no
	p53	TSG	17: (7571720..7590868)	no	no
	PTEN	TSG	10: (89623195..89728532)	no	no
	CDKN2A	TSG	9: (21967751..21994490)	yes (21973069..21983401)	yes (21973069.. 21983401)

Table B.6: List of genes with their locations that are commonly altered in OV, LUSC and GBM cancer cells, and whether or not they were found by the power-law and exponential methods using the three highest scoring non-overlapping intervals. The OV and GBM genes were taken from the Kegg database (`http://www.genome.jp/dbget-bin/www_bget?ds:H00027` and `http://www.genome.jp/dbget-bin/www_bget?ds:H00042`); the LUSC genes, for which no Kegg entry exists, are commonly amplified/deleted LUSC driver genes from [59] (mentioned on page 519). A gene is considered 'found' if the selected interval intersects with the region containing the gene. In this table 'next' indicates within 100kbp from a border of the gene interval. The parameters μ, α, and λ were estimated from the data as in [76] and Table 3.1, with the exception of α of LUSC del which was set to 1, as the computation of RR score assumes $\alpha \geq 1$. Segments shorter than 10^4 base pairs (corresponding to the distance between two probes) and longer than 10^7 base pairs were excluded.

Appendix C

Progression Extraction

C.1 Proof of Proposition 2 (Probability raising temporal priority)

Proposition (Probability raising and probabilities)**.** *For any two events a and b such that the probability raising $\mathcal{P}(a \mid b) > \mathcal{P}(a \mid \overline{b})$ holds, we have*

$$\mathcal{P}(a) > \mathcal{P}(b) \iff \frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})}\,.$$

Proof. We first prove the left-to-right direction "⇒". Let $x = \mathcal{P}(\overline{a}, b)$, $y = \mathcal{P}(a, b)$ and $z = \mathcal{P}(a, \overline{b})$. We have two assumptions we shall use later on:

1. $\mathcal{P}(a) > \mathcal{P}(b)$ which implies $\mathcal{P}(\overline{a}, b) < \mathcal{P}(a, \overline{b})$, i.e. $x < z$.

2. $\mathcal{P}(a \mid b) > \mathcal{P}(a \mid \overline{b})$ which implies, by simple algebraic rearrangements when $0 < x + y < 1$, inequality

$$y[1 - x - y - z] > xz \ . \tag{C.1}$$

We proceed by rewriting $\mathcal{P}(b \mid a)/\mathcal{P}(b \mid \overline{a}) > \mathcal{P}(a \mid b)/\mathcal{P}(a \mid \overline{b})$ as

$$\frac{\mathcal{P}(a,b)\mathcal{P}(\overline{a})}{\mathcal{P}(\overline{a},b)\mathcal{P}(a)} > \frac{\mathcal{P}(a,b)\mathcal{P}(\overline{b})}{\mathcal{P}(a,\overline{b})\mathcal{P}(b)} \ ,$$

which means that

$$\frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \iff \frac{\mathcal{P}(\overline{a})}{\mathcal{P}(\overline{a},b)\mathcal{P}(a)} > \frac{\mathcal{P}(\overline{b})}{\mathcal{P}(a,\overline{b})\mathcal{P}(b)} \ . \tag{C.2}$$

We can rewrite the right side of (C.2) by using x, y, z where $\mathcal{P}(a) = \mathcal{P}(a \mid b) + \mathcal{P}(a \mid \overline{b}) = y + z$ and $\mathcal{P}(b) = \mathcal{P}(b \mid a) + \mathcal{P}(b \mid \overline{a}) = x + y$, and then do some algebraic manipulations. We have

$$\frac{1-y-z}{x(y+z)} > \frac{1-x-y}{z(x+y)} \iff yz - y^2z - xz^2 - yz^2 > xy - x^2y - x^2z - xy^2 \ , \tag{C.3}$$

when $x(y+z) \neq 0$ and $z(x+y) \neq 0$. To check that the right side of (C.3) holds we show that

$$(xy - x^2y - x^2z - xy^2) - (yz - y^2z - xz^2 - yz^2) < 0 \ .$$

First, we rearrange it to $(x-z)[y - y^2 - xz - y(x+z)] < 0$ so to show that

$$(x-z)[y(1-y-x-z) - zx] < 0 \tag{C.4}$$

is always negative. By observing that by assumption 1 we have $z > x$ and thus $(x-z) < 0$, and by equation (C.1) we have $y(1-y-x-z) - zx > 0$ we derive

$$\frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \ ,$$

which concludes the "$\Rightarrow$" direction. The other direction "$\Leftarrow$" follows immediately by contraposition: assume $\mathcal{P}(a \mid b) > \mathcal{P}(a \mid \overline{b})$ and $\mathcal{P}(b \mid a)/\mathcal{P}(b \mid \overline{a}) > \mathcal{P}(a \mid b)/\mathcal{P}(a \mid \overline{b})$, and suppose $\mathcal{P}(b) \leq \mathcal{P}(a)$. We distinguish two cases:

1. $\mathcal{P}(b) = \mathcal{P}(a)$, then $\mathcal{P}(b \mid a)/\mathcal{P}(b \mid \overline{a}) = \mathcal{P}(a \mid b)/\mathcal{P}(a \mid \overline{b})$.

2. $\mathcal{P}(b) < \mathcal{P}(a)$, then by symmetry of probability raising $\mathcal{P}(b \mid a) > \mathcal{P}(b \mid \overline{a})$, and by the "⇒"-direction of the proposition it follows that $\mathcal{P}(b \mid a)/\mathcal{P}(b \mid \overline{a}) < \mathcal{P}(a \mid b)/\mathcal{P}(a \mid \overline{b})$.

In both cases we have a contradiction. This finishes the proof. □

C.2 Proof of Proposition 3 (Monotonic normalization)

Proposition (Monotonic normalization)**.** *For any two events a and b we have*

$$\frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \iff m_{b,a} > m_{a,b}$$

Proof. We prove the left-to-right direction "⇒", the other direction follows by a similar argument. Let us assume

$$\frac{\mathcal{P}(b \mid a)}{\mathcal{P}(b \mid \overline{a})} > \frac{\mathcal{P}(a \mid b)}{\mathcal{P}(a \mid \overline{b})} \ , \tag{C.5}$$

then $\mathcal{P}(b \mid a)\mathcal{P}(a \mid \overline{b}) > \mathcal{P}(a \mid b)\mathcal{P}(b \mid \overline{a})$. Now, to show the righthand side of the implication, we will show that

$$\Big[\mathcal{P}(b \mid a) - \mathcal{P}(b \mid \overline{a})\Big]\Big[\mathcal{P}(a \mid b) + \mathcal{P}(a \mid \overline{b})\Big] > \Big[\mathcal{P}(b \mid a) + \mathcal{P}(b \mid \overline{a})\Big]\Big[\mathcal{P}(a \mid b) - \mathcal{P}(a \mid \overline{b})\Big] \ ,$$

which reduces to show

$$\mathcal{P}(b \mid a)\mathcal{P}(a \mid \overline{b}) - \mathcal{P}(b \mid \overline{a})\mathcal{P}(a \mid b) > \mathcal{P}(b \mid \overline{a})\mathcal{P}(a \mid b) - \mathcal{P}(b \mid a)\mathcal{P}(a \mid \overline{b}) \ .$$

By (C.5), two equivalent inequalities hold

$$\mathcal{P}(b \mid a)\mathcal{P}(a \mid \overline{b}) - \mathcal{P}(b \mid \overline{a})\mathcal{P}(a \mid b) > 0$$
$$\mathcal{P}(b \mid \overline{a})\mathcal{P}(a \mid b) - \mathcal{P}(b \mid a)\mathcal{P}(a \mid \overline{b}) < 0$$

and hence the implication holds. □

C.3 Proof of Theorem 11 (Algorithm correctness)

Theorem (Algorithm correctness)**.** *Algorithm 1 reconstructs a well defined tree $\mathcal{T}$ without disconnected components, transitive connections and cycles.*

Proof. It is clear that Algorithm 1 does not create disconnected components since, to each node in G, a unique parent is attached (either from G or $\diamond$). For the same reason, no transitive connections can appear.

The absence of cycles results from both Proposition 2 and 3. Indeed, suppose for contradiction that there is a cycle $(a_1, a_2), (a_2, a_3), \ldots, (a_n, a_1)$ in E, then by the two propositions we have

$$\mathcal{P}(a_1) > \mathcal{P}(a_2) > \ldots > \mathcal{P}(a_n) > \mathcal{P}(a_1) \ ,$$

which is a contradiction. □

Bibliography

[1] Altisen, K., Gossler, G., Pnueli, A., Sifakis, J., Tripakis, S., and Yovine, S. A framework for scheduler synthesis. In *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*, pages 154 –163, 1999.

[2] Alur, R., Courcoubetis, C., and Dill, D. Model-checking in dense real-time. *Information and Computation*, 104(1):2 – 34, 1993. ISSN 0890-5401. doi: 10.1006/inco.1993.1024. URL `http://www.sciencedirect.com/science/article/pii/S0890540183710242`.

[3] Alur, R. and Dill, D. Automata for modeling real-time systems. In Paterson, M., editor, *Automata, Languages and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer Berlin / Heidelberg, 1990. ISBN 978-3-540-52826-5. URL `http://dx.doi.org/10.1007/BFb0032042`. 10.1007/BFb0032042.

[4] Alur, R. and Dill, D. L. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994. ISSN 0304-3975. doi: 10.1016/0304-3975(94)90010-8. URL `http://www.sciencedirect.com/science/article/pii/0304397594900108`.

[5] Alur, R., Bernadsky, M., and Madhusudan, P. Optimal reachability for weighted timed games. In Díaz, J., Karhumaki, J., Lepistö, A., and Sannella, D., editors, *Automata, Languages and Programming*, volume 3142 of *Lecture Notes*

in Computer Science, pages 97–112. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22849-3.

[6] Asarin, E. and Maler, O. As soon as possible: Time optimal control for timed automata. In Vaandrager, F. and van Schuppen, J., editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 19–30. Springer Berlin / Heidelberg, 1999. ISBN 978-3-540-65734-7.

[7] Asarin, E., Maler, O., and Pnueli, A. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II, LNCS 999*, pages 1–20. Springer, 1995.

[8] Asarin, E., Maler, O., Pnueli, A., and Sifakis, J. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469—474. Elsevier, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.5874`.

[9] Beerenwinkel, N., Rahnenführer, J., Däumer, M., Hoffmann, D., Kaiser, R., Selbig, J., and Lengauer, T. Learning multiple evolutionary pathways from cross-sectional data. *Journal of Computational Biology*, 12(6):584–598, 2005.

[10] Beerenwinkel, N., Antal, T., Dingli, D., Traulsen, A., Kinzler, K. W., Velculescu, V. E., Vogelstein, B., and Nowak, M. A. Genetic progression and the waiting time to cancer. *PLoS computational biology*, 3(11):e225, 2007.

[11] Beerenwinkel, N., Eriksson, N., and Sturmfels, B. Conjunctive bayesian networks. *Bernoulli*, pages 893–909, 2007.

[12] Behrmann, G., Cougnard, A., David, A., Fleury, E., Larsen, K. G., and Lime, D. Uppaal-tiga: time for playing games! In *Proceedings of the 19th international conference on Computer aided verification*, CAV'07, pages 121–125, Berlin,

Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73367-6. URL http://dl.acm.org/citation.cfm?id=1770351.1770370.

[13] Bell, D., Berchuck, A., Birrer, M., Chien, J., Cramer, D., Dao, F., Dhir, R., DiSaia, P., Gabra, H., and Glenn, P. Integrated genomic analyses of ovarian carcinoma. 2011.

[14] Bengtsson, J. and Yi, W. Timed automata: Semantics, algorithms and tools. In Desel, J., Reisig, W., and Rozenberg, G., editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 87–124. Springer Berlin / Heidelberg, 2004. ISBN 978-3-540-22261-3.

[15] Beroukhim, R., Getz, G., Nghiemphu, L., Barretina, J., Hsueh, T., Linhart, D., Vivanco, I., Lee, J. C., Huang, J. H., Alexander, S., et al. Assessing the significance of chromosomal aberrations in cancer: methodology and application to glioma. *Proceedings of the National Academy of Sciences*, 104(50):20007–20012, 2007.

[16] Bertoli, P. Planning with extended goals and partial observability. In *In Proceedings of ICAPS04*, pages 270–278, 2004.

[17] Bouyer, P., Brihaye, T., and Chevalier, F. Control in o-minimal hybrid systems. In *Logic in Computer Science, 2006 21st Annual IEEE Symposium on*, pages 367 –378, 0-0 2006.

[18] Bouyer, P. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24:281–320, 2004. ISSN 0925-9856. URL http://dx.doi.org/10.1023/B:FORM.0000026093.21513.31. 10.1023/B:FORM.0000026093.21513.31.

[19] Bouyer, P. Weighted timed automata: Model-checking and games. *Electronic*

Notes in Theoretical Computer Science, 158(0):3–17, 2006. ISSN 1571-0661. doi: 10.1016/j.entcs.2006.04.002.

[20] Bouyer, P., D'Souza, D., Madhusudan, P., and Petit, A. Timed control with partial observability. In Hunt, W. A. and Somenzi, F., editors, *Computer Aided Verification*, volume 2725, pages 278–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-40524-5. URL `http://www.springerlink.com/content/djf805g74a0jwxlq/`.

[21] Bouyer, P., Bozzelli, L., and Chevalier, F. Controller synthesis for mtl specifications. In Baier, C. and Hermanns, H., editors, *CONCUR 2006 – Concurrency Theory*, volume 4137 of *Lecture Notes in Computer Science*, pages 450–464. Springer Berlin, Heidelberg, 2006. ISBN 978-3-540-37376-6.

[22] Bouyer, P., Fahrenberg, U., Larsen, K. G., and Markey, N. Quantitative analysis of real-time systems using priced timed automata. *Commun. ACM*, 54(9):78–87, September 2011. ISSN 0001-0782. doi: 10.1145/1995376.1995396. URL `http://doi.acm.org/10.1145/1995376.1995396`.

[23] Bouyer, P., Fahrenberg, U., Larsen, K. G., and Markey, N. Quantitative analysis of real-time systems using priced timed automata. *Commun. ACM*, 54(9):78–87, 2011.

[24] Bozic, I., Reiter, J. G., Allen, B., Antal, T., Chatterjee, K., Shah, P., Moon, Y. S., Yaqubie, A., Kelly, N., Le, D. T., et al. Evolutionary dynamics of cancer in response to targeted combination therapy. *eLife*, 2, 2013.

[25] Cassez, F. Efficient on-the-fly algorithms for partially observable timed games. In Raskin, J.-F. and Thiagarajan, P., editors, *Formal Modeling and Analysis of Timed Systems*, volume 4763 of *Lecture Notes in Computer Science*, pages 5–24. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-75453-4.

[26] Cassez, F., David, A., Fleury, E., Larsen, K., and Lime, D. Efficient on-the-fly algorithms for the analysis of timed games. In Abadi, M. and de Alfaro, L., editors, *CONCUR 2005 – Concurrency Theory*, volume 3653 of *Lecture Notes in Computer Science*, pages 66–80. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-28309-6.

[27] Cassez, F., David, A., Larsen, K. G., Lime, D., and Raskin, J. Timed control with observation based and stuttering invariant strategies. In Namjoshi, K. S., Yoneda, T., Higashino, T., and Okamura, Y., editors, *Automated Technology for Verification and Analysis*, volume 4762, pages 192–206. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-75595-1. URL `http://www.springerlink.com/content/b03282876h733023/`.

[28] Cassez, F., Jessen, J., Larsen, K., Raskin, J.-F., and Reynier, P.-A. Automatic synthesis of robust and optimal controllers – an industrial case study. In Majumdar, R. and Tabuada, P., editors, *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 90–104. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-00601-2.

[29] Chapman, P. B., Hauschild, A., Robert, C., Haanen, J. B., Ascierto, P., Larkin, J., Dummer, R., Garbe, C., Testori, A., Maio, M., et al. Improved survival with vemurafenib in melanoma with braf v600e mutation. *New England Journal of Medicine*, 364(26):2507–2516, 2011.

[30] Chatterjee, K., Doyen, L., Henzinger, T. A., and Raskin, J.-F. Algorithms for omega-regular games with imperfect information. In *Proceedings of the 20th international conference on Computer Science Logic*, CSL'06, pages 287–302, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45458-6, 978-3-540-45458-8. doi: 10.1007/11874683_19. URL `http://dx.doi.org/10.1007/11874683_19`.

[31] Clarke, E. M., Emerson, E. A., and Sistla, A. P. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8:244–263, April 1986. ISSN 0164-0925.

[32] Clarke, E. M., Emerson, E. A., and Sistla, A. P. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, April 1986. ISSN 0164-0925. doi: 10.1145/5397.5399. URL `http://doi.acm.org/10.1145/5397.5399`.

[33] Clarke, E. and Emerson, E. Design and synthesis of synchronization skeletons using branching time temporal logic. In Kozen, D., editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin / Heidelberg, 1982. ISBN 978-3-540-11212-9. URL `http://dx.doi.org/10.1007/BFb0025774`.

[34] Clarke, E. M., Grumberg, O., and Peled, D. A. *Model Checking*. MIT Press, January 1999. ISBN 0262032708.

[35] Comen, E., Norton, L., and Massague, J. Clinical implications of cancer self-seeding. *Nat Rev Clin Oncol*, 8(6):369–377, June 2011. ISSN 1759-4774. doi: 10.1038/nrclinonc.2011.64.

[36] Daruwala, R., Rudra, A., Ostrer, H., Lucito, R., Wigler, M., and Mishra, B. A versatile statistical analysis algorithm to detect genome copy number variation. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16292–16297, November 2004. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0407247101. URL `http://www.pnas.org/content/101/46/16292`.

[37] de Visser, K. E., Eichten, A., and Coussens, L. M. Paradoxical roles of the immune system during cancer development. *Nature Reviews Cancer*, 6(1):24–37, January 2006. ISSN 1474-175X. doi: 10.1038/nrc1782.

[38] De Wulf, M., Doyen, L., and Raskin, J.-F. A lattice theory for solving games of imperfect information. In Hespanha, J. and Tiwari, A., editors, *Hybrid Systems: Computation and Control*, volume 3927 of *Lecture Notes in Computer Science*, pages 153–168. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33170-4.

[39] Desper, R., Jiang, F., Kallioniemi, O., Moch, H., Papadimitriou, C., and Schäffer, A. Inferring tree models for oncogenesis from comparative genome hybridization data. *Journal of Computational Biology*, 6(1):37–51, 1999.

[40] Desper, R., Jiang, F., Kallioniemi, O., Moch, H., Papadimitriou, C., and Schäffer, A. Distance-based reconstruction of tree models for oncogenesis. *Journal of Computational Biology*, 7(6):789–803, 2000.

[41] Druker, B. J., Guilhot, F., O'Brien, S. G., Gathmann, I., Kantarjian, H., Gattermann, N., Deininger, M. W., Silver, R. T., Goldman, J. M., Stone, R. M., et al. Five-year follow-up of patients receiving imatinib for chronic myeloid leukemia. *New England Journal of Medicine*, 355(23):2408–2417, 2006.

[42] Edmonds, J. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71:233–240, 1967.

[43] Efron, B. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

[44] Efron, B. Logistic regression, survival analysis, and the kaplan-meier curve. *Journal of the American Statistical Association*, 83(402):414–425, 1988.

[45] Efron, B. Large-scale simultaneous hypothesis testing: the choice of a null hypothesis. *Journal of the American Statistical Association*, 99(465):96–104, 2004.

[46] Efron, B. and Efron, B. *The jackknife, the bootstrap and other resampling plans*, volume 38. SIAM, 1982.

[47] Faella, M., Torre, S. L., and Murano, A. Automata-theoretic decision of timed games. In *Revised Papers from the Third International Workshop on Verification, Model Checking, and Abstract Interpretation*, VMCAI '02, pages 94–108, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43631-6. URL http://dl.acm.org/citation.cfm?id=646541.696186.

[48] Fahrenberg, U. and Larsen, K. G. Discount-optimal infinite runs in priced timed automata. *Electronic Notes in Theoretical Computer Science*, 239(0):179 – 191, 2009. ISSN 1571-0661. doi: 10.1016/j.entcs.2009.05.039. URL http://www.sciencedirect.com/science/article/pii/S1571066109001613. <ce:title>Joint Proceedings of the 8th, 9th, and 10th International Workshops on Verification of Infinite-State Systems (INFINITY 2006, 2007, 2008)</ce:title>.

[49] Fidler, I. J. Tumor heterogeneity and the biology of cancer invasion and metastasis. *Cancer Research*, 38(9):2651 –2660, 1978. URL http://cancerres.aacrjournals.org/content/38/9/2651.abstract.

[50] Forejt, V., Kwiatkowska, M., Norman, G., and Trivedi, A. Expected reachability-time games. In Chatterjee, K. and Henzinger, T., editors, *Formal Modeling and Analysis of Timed Systems*, volume 6246 of *Lecture Notes in Computer Science*, pages 122–136. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-15296-2.

[51] Fried, D. B., Morris, D. E., Poole, C., Rosenman, J. G., Halle, J. S., Detterbeck, F. C., Hensing, T. A., and Socinski, M. A. Systematic review evaluating the timing of thoracic radiation therapy in combined modality therapy for limited-

stage small-cell lung cancer. *Journal of clinical oncology*, 22(23):4837–4845, 2004.

[52] Gärdenfors, P. *Belief revision*, volume 29. Cambridge University Press, 2003.

[53] Gatenby, R. A. and Vincent, T. L. An evolutionary model of carcinogenesis. *Cancer Research*, 63(19):6212–6220, 2003.

[54] Gerstung, M., Baudis, M., Moch, H., and Beerenwinkel, N. Quantifying cancer progression with conjunctive bayesian networks. *Bioinformatics*, 25(21):2809–2815, 2009.

[55] Gerstung, M., Eriksson, N., Lin, J., Vogelstein, B., and Beerenwinkel, N. The temporal order of genetic and pathway alterations in tumorigenesis. *PLoS One*, 6(11):e27136, 2011.

[56] Greaves, M. and Maley, C. C. Clonal evolution in cancer. *Nature*, 481(7381): 306–313, 2012.

[57] Gunawan, B., Von Heydebreck, A., Sander, B., Schulten, H.-J., Haller, F., Langer, C., Armbrust, T., Bollmann, M., Gašparov, S., Kovač, D., et al. An oncogenetic tree model in gastrointestinal stromal tumours (gists) identifies different pathways of cytogenetic evolution with prognostic implications. *The Journal of pathology*, 211(4):463–470, 2007.

[58] Hahn, W. C. and Weinberg, R. A. *A subway map of cancer pathways*. Nature Publishing Group, 2002.

[59] Hammerman, P. S., Lawrence, M. S., Voet, D., Jing, R., Cibulskis, K., Sivachenko, A., Stojanov, P., McKenna, A., Lander, E. S., Gabriel, S., et al. Comprehensive genomic characterization of squamous cell lung cancers. *Nature*, 489(7417):519–525, 2012.

[60] Hanahan, D. and Weinberg, R. A. The hallmarks of cancer. *Cell*, 100(1):57–70, 2000. doi: 10.1016/S0092-8674(00)81683-9.

[61] Hanahan, D. and Weinberg, R. A. Hallmarks of cancer: The next generation. *Cell*, 144(5):646–674, 2011. ISSN 00928674. doi: 10.1016/j.cell.2011.02.013. URL `http://linkinghub.elsevier.com/retrieve/pii/S0092867411001279`.

[62] Henzinger, T. The theory of hybrid automata. In *Logic in Computer Science, 1996. LICS'96. Proceedings., Eleventh Annual IEEE Symposium on*, pages 278–292. IEEE, 1996.

[63] Henzinger, T., Nicollin, X., Sifakis, J., and Yovine, S. Symbolic model checking for real-time systems. In *Logic in Computer Science, 1992. LICS '92., Proceedings of the Seventh Annual IEEE Symposium on*, pages 394 –406, jun 1992. doi: 10.1109/LICS.1992.185551.

[64] Henzinger, T., Ho, P.-H., and Wong-Toi, H. Algorithmic analysis of nonlinear hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):540 –554, apr 1998. ISSN 0018-9286. doi: 10.1109/9.664156.

[65] Henzinger, T. A. and Kopke, P. W. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1-2):369 – 392, 1999. ISSN 0304-3975. doi: 10.1016/S0304-3975(99)00038-9. URL `http://www.sciencedirect.com/science/article/pii/S0304397599000389`.

[66] Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. What's decidable about hybrid automata? In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 373–382, New York, NY, USA, 1995. ACM. ISBN 0-89791-718-9.

[67] Henzinger, T. A., Horowitz, B., and Majumdar, R. Rectangular hybrid games. In *In CONCUR 99, LNCS 1664*, pages 320–335. Springer, 1999.

[68] Heppner, G. H. Tumor heterogeneity. *Cancer Research*, 44(6):2259 –2265, June 1984. URL `http://cancerres.aacrjournals.org/content/44/6/2259.short`.

[69] Hicks, J., Krasnitz, A., Lakshmi, B., Navin, N. E., Riggs, M., Leibu, E., Esposito, D., Alexander, J., Troge, J., Grubor, V., et al. Novel patterns of genome rearrangement and their association with survival in breast cancer. *Genome research*, 16(12):1465–1479, 2006.

[70] Hitchcock, C. Probabilistic causation. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Winter 2012 edition, 2012.

[71] Hruban, R. H., Goggins, M., Parsons, J., and Kern, S. E. Progression model for pancreatic cancer. *Clinical cancer research*, 6(8):2969, 2000.

[72] Huang, Q., Yu, G. P., McCormick, S. A., Mo, J., Datta, B., Mahimkar, M., Lazarus, P., Schäffer, A. A., Desper, R., and Schantz, S. P. Genetic differences detected by comparative genomic hybridization in head and neck squamous cell carcinomas from different tumor sites: construction of oncogenetic trees for tumor progression. *Genes, Chromosomes and Cancer*, 34(2):224–233, 2002.

[73] Hupé, P., Stransky, N., Thiery, J.-P., Radvanyi, F., and Barillot, E. Analysis of array cgh data: from signal ratio to gain and loss of dna regions. *Bioinformatics*, 20(18):3413–3422, December 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bth418. URL `http://dx.doi.org/10.1093/bioinformatics/bth418`.

[74] Imielinski, M., Berger, A. H., Hammerman, P. S., Hernandez, B., Pugh, T. J., Hodis, E., Cho, J., Suh, J., Capelletti, M., Sivachenko, A., et al. Mapping the hallmarks of lung adenocarcinoma with massively parallel sequencing. *Cell*, 150 (6):1107–1120, 2012.

[75] Ionita, I. *Multimarker genetic analysis methods for high throughput array data*. PhD thesis, New York University, 2006.

[76] Ionita, I., Daruwala, R., and Mishra, B. Mapping Tumor-Suppressor genes with multipoint statistics from Copy-Number–Variation data. *American Journal of Human Genetics*, 79(1):13–22, July 2006. ISSN 0002-9297. PMID: 16773561 PMCID: 1474131.

[77] Jessen, J. J., Rasmussen, J. I., Larsen, K. G., and David, A. Guided controller synthesis for climate controller using uppaal tiga. In *Proceedings of the 5th international conference on Formal modeling and analysis of timed systems*, FORMATS'07, pages 227–240, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-75453-9, 978-3-540-75453-4.

[78] Jiang, S. and Kumar, R. Supervisory control of discrete event systems with ctl* temporal logic specifications. *SIAM Journal on Control and Optimization*, 44 (6):2079–2103, 2006. doi: 10.1137/S0363012902409982.

[79] Jiang, S. and Kumar, R. Supervisory control of discrete event systems with CTL* temporal logic specifications. *SIAM Journal on Control and Optimization*, 44(6):2079–2103, January 2006. doi: 10.1137/S0363012902409982. URL `http://link.aip.org/link/?SJC/44/2079/1`.

[80] Jones, S., Chen, W.-d., Parmigiani, G., Diehl, F., Beerenwinkel, N., Antal, T., Traulsen, A., Nowak, M. A., Siegel, C., Velculescu, V. E., et al. Comparative lesion sequencing provides insights into tumor evolution. *Proceedings of the National Academy of Sciences*, 105(11):4283–4288, 2008.

[81] Kaelin, W. G. The concept of synthetic lethality in the context of anticancer therapy. *Nature reviews cancer*, 5(9):689–698, 2005.

[82] Kaelin Jr, W. G. The concept of synthetic lethality in the context of anticancer therapy. *Nat Rev Cancer*, 5(9):689–698, 2005.

[83] Kainu, T., Juo, S.-H. H., Desper, R., Schäffer, A. A., and Gillanders, E. Somatic deletions in hereditary breast cancers implicate 13q21 as a putative novel breast cancer susceptibility locus. *Proceedings of the National Academy of Sciences*, 97 (17):9603–9608, 2000.

[84] Kleinberg, S. An algorithmic enquiry concerning causality. *Thesis*, 2010.

[85] Knutsen, T., Gobu, V., Knaus, R., Padilla-Nash, H., Augustus, M., Strausberg, R. L., Kirsch, I. R., Sirotkin, K., and Ried, T. The interactive online sky/m-fish & cgh database and the entrez cancer chromosomes search database: Linkage of chromosomal aberrations with the genome sequence. *Genes, Chromosomes and Cancer*, 44(1):52–64, 2005.

[86] Komarova, N. L. and Boland, C. R. Cancer: Calculated treatment. *Nature*, 499 (7458):291–292, 2013.

[87] LaValle, S. M. Planning algorithms, 2004.

[88] Liu, X. and Smolka, S. Simple linear-time algorithms for minimal fixed points. In Larsen, K., Skyum, S., and Winskel, G., editors, *Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 53–66. Springer Berlin / Heidelberg, 1998. ISBN 978-3-540-64781-2.

[89] Luo, J., Solimini, N. L., and Elledge, S. J. Principles of cancer therapy: Oncogene and non-oncogene addiction. *Cell*, 136(5):823–837, March 2009. ISSN 0092-8674. doi: 10.1016/j.cell.2009.02.024. URL `http://www.sciencedirect.com/science/article/B6WSN-4VS49KS-9/2/2dbe52f01a823f0c41169dbad5978c2f`.

[90] Maler, O. Control from computer science. *Annual Reviews in Control*, 26(2): 175–187, 2002.

[91] Maler, O., Pnueli, A., and Sifakis, J. On the synthesis of discrete controllers for timed systems. In Mayr, E. and Puech, C., editors, *STACS 95*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer Berlin / Heidelberg, 1995. ISBN 978-3-540-59042-2.

[92] Michor, F., Iwasa, Y., and Nowak, M. A. Dynamics of cancer progression. *Nature Reviews Cancer*, 4(3):197–205, 2004.

[93] Navin, N., Kendall, J., Troge, J., Andrews, P., Rodgers, L., McIndoo, J., Cook, K., Stepansky, A., Levy, D., Esposito, D., Muthuswamy, L., Krasnitz, A., McCombie, W. R., Hicks, J., and Wigler, M. Tumour evolution inferred by single-cell sequencing. *Nature*, 472(7341):90–94, April 2011. ISSN 0028-0836. doi: 10.1038/nature09807.

[94] Olde Loohuis, L., Witzel, A., and Mishra, B. Towards cancer hybrid automata. *Proceedings of Hybrid Systems and Biology (HSB)*, 92:137–151, 2012.

[95] Olde Loohuis, L., Witzel, A., and Mishra, B. Cancer hybrid automata: Model, beliefs & therapy. *Submitted to: Journal of Information and Computation*, 2013.

[96] Olde Loohuis, L., Caravagna, G., Graudenzi, A., Ramazzotti, D., Antoniotti, M., and Mishra, B. Extracting progression trees using probabilistic causality. *unpublished manuscript*, 2013.

[97] Olde Loohuis, L., Witzel, A., and Mishra, B. Power-law null model for bystander mutations in cancer. *Submitted to IEEE Transactions on Computational Biology and Bioinformatics*, 2013.

[98] Olshen, A. B., Venkatraman, E., Lucito, R., and Wigler, M. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, 5(4):557–572, 2004.

[99] Orlando, P. A., Gatenby, R. A., and Brown, J. S. Cancer treatment as a game: integrating evolutionary game theory into the optimal control of chemotherapy. *Physical biology*, 9(6):065007, 2012.

[100] Parikh, R., Taşdemir, Ç., and Witzel, A. Choice and uncertainty in games. In *Logic and Program Semantics*, pages 244–255. Springer, 2012.

[101] Pathare, S., Schäffer, A. A., Beerenwinkel, N., and Mahimkar, M. Construction of oncogenetic tree models reveals multiple pathways of oral cancer progression. *International journal of cancer*, 124(12):2864–2871, 2009.

[102] Radmacher, M. D., Simon, R., Desper, R., Taetle, R., SCHÄFFER, A. A., and Nelson, M. A. Graph models of oncogenesis with an application to melanoma. *Journal of theoretical biology*, 212(4):535–548, 2001.

[103] Rahman, A., Korzekwa, K., Grogan, J., Gonzalez, F., and Harris, J. Selective biotransformation of taxol to 6α-hydroxytaxol by human cytochrome p450 2c8. *Cancer research*, 54(21):5543–5546, 1994.

[104] Reif, J. H. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274 – 301, 1984. ISSN 0022-0000.

[105] Rodriguez-Brenes, I., Komarova, N., and Wodarz, D. Evolutionary dynamics of feedback escape and the development of stem-cell-driven cancers. *Proceedings of the National Academy of Sciences*, 108(47), 2011.

[106] Sottoriva, A., Spiteri, I., Shibata, D., Curtis, C., and Tavaré, S. Single-molecule genomic data delineate patient-specific tumor profiles and cancer stem cell organization. *Cancer research*, 73(1):41–49, 2013.

[107] Suppes, P. *A probabilistic theory of causality*. North Holland Publishing Company, 1970.

[108] Thomas, W. On the synthesis of strategies in infinite games. In Mayr, E. and Puech, C., editors, *STACS 95*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin / Heidelberg, 1995. ISBN 978-3-540-59042-2.

[109] Toi, H. W. The synthesis of controllers for linear hybrid automata, 1997. URL `http://citeseer.ist.psu.edu/wong-toi97synthesis.html`.

[110] Tomlin, C., Pappas, G., and Sastry, S. Conflict resolution for air traffic management: a study in multiagent hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):509 –521, apr 1998. ISSN 0018-9286.

[111] Torre, S. L., Mukhopadhyay, S., and Murano, A. Optimal-reachability and control for acyclic weighted timed automata. In *In Proceedings of IFIP TCS'02, IFIP Conference Proceedings 223, 485–497, Kluwer*, pages 485–497. Kluwer, 2002.

[112] van der Hoek, W. and Wooldridge, M. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[113] Vander Heiden, M. G., Cantley, L. C., and Thompson, C. B. Understanding the warburg effect: the metabolic requirements of cell proliferation. *science*, 324 (5930):1029–1033, 2009.

[114] Vogelstein, B. and Kinzler, K. Cancer genes and the pathways they control. *Nature medicine*, 10(8):789–799, 2004.

[115] Wallenstein, S. and Neff, N. An approximation for the distribution of the scan statistic. *Statistics in Medicine*, 6(2):197–207, 1987.

[116] Xue, W., Kitzing, T., Roessler, S., Zuber, J., Krasnitz, A., Schultz, N., Revill, K., Weissmueller, S., Rappaport, A. R., Simon, J., et al. A cluster of cooperating tumor-suppressor gene candidates in chromosomal deletions. *Proceedings of the National Academy of Sciences*, 109(21):8212–8217, 2012.

[117] Yan, C., Kim, Y.-W., Ha, Y.-S., Kim, I. Y., Kim, Y.-J., Yun, S.-J., Moon, S.-K., Bae, S.-C., and Kim, W.-J. Runx3 methylation as a predictor for disease progression in patients with non-muscle-invasive bladder cancer. *Journal of surgical oncology*, 105(4):425–430, 2012.

[118] Zhang, K. and Shasha, D. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.