Analysis of state-transition graphs of ecosystems using model-checking

Colin Thomas

université
PARIS-SACLAY

Analysis of state-transition graphs of ecosystems using model-checking

Analyse de graphes état-transition d'écosystèmes à l'aide de model-checking

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et sciences du numérique
Référent : Université d'Évry Val d'Essonne

Thèse préparée dans les unités de recherche IBISC (Université Paris-Saclay, Univ Evry) et UMR AMAP (INRAE), sous la direction de Franck POMMEREAU, professeur, et de Cédric GAUCHEREL, directeur de recherche à INRAE

Thèse soutenue à Evry, le 12 décembre 2022, par

Colin THOMAS

Composition du jury

Hanna KLAUDEL Professeure des universités, IBISC, Université d'Evry Val d'Essonne, Université Paris- Saclay	Présidente
Christine LARGOUËT Maîtresse de conférences, Institut Agro Rennes-Angers, Université de Rennes 1	Rapporteure & Examinatrice
Yann THIERRY-MIEG Maître de conférences, LIP6, Paris Sorbonne Universités	Rapporteur & Examinateur
Isabelle BOULANGEAT Chargée de recherche, LESSEM, Université Grenoble Alpes	Examinatrice
Serenella CERITTO Professeure des universités, IBISC, Université d'Evry Val d'Essonne, Université Paris- Saclay	Examinatrice
Loïc PAULEVÉ Chargé de recherche, CNRS, Université de Bordeaux	Examinateur
Franck POMMEREAU Professeur des universités, IBISC, Université d'Evry val d'Essonne, Université Paris- Saclay	Directeur de thèse

THESE DE DOCTORAT
NNT : 2020UPASG087

Contents

1 Introduction

Earth is experiencing its 6th mass extinction: experts estimate that 20% to 50% of living species may go extinct during the 21st century [Mil05]. In this context, understanding ecosystems and their dynamics is crucial to be able to take conservation actions. Ecosystems follow trajectories that can be abstracted into a state-transition graph mapping every possible pathway, just like a road map in which the ecosystem's course can be tracked. For example, a savanna can branch off towards alternative futures, such as getting encroached by bushes, turning into a forest, or drying out into a desert. These changes may be reversible, cyclic or irreversible, i.e. resulting in the ecosystem being trapped in some dead-end configuration. How an ecosystem moves along these trajectories is a question that remains not entirely understood. Yet, human societies want to impact the ecosystem's course, for example to restore a degraded ecosystem or to sustain agricultural production, all the more in the context of global warming that affects the ecosystem's pathways. Methods to analyse such branching trajectories have been developed for decades in computer science to verify automated systems, but remain largely unknown to ecologists. In this thesis, we showcase how this mathematical framework originating from computer science can be adapted to fit ecology needs, we illustrate the insights it can produce and discuss the new challenges that arise from it.

A *state-transition graph (STG)* describes the behaviour of a dynamical system, for example an ecosystem, as a graph whose nodes are the discrete states of the system and whose edges represent the transitions between these states. An STG is a kind of road map in which the ecosystem travels following pathways that may branch off into distinct directions, converge together, be cyclic, etc. An example STG describing vegetation pathways in Ethiopia is given in Figure 1.1. STGs are widespread in ecology, but they must not be confused with interaction graphs such as the iconic trophic networks. The former represent the temporal trajectories of a system while the latter represent the processes taking place between the entities composing a system.

In ecology, STGs typically represent *community pathways*, i.e. changes in the set of species or populations of an ecosystem through time. Ecological STGs are often designed as user-friendly tools enabling participatory model development and collaborative management of ecosystems. As a graphical representation of the behaviour of the ecosystem, STGs assist managers and scientists in collectively proposing policies driving the ecosystem through desired pathways while avoiding others. Ecology encompasses a wide variety of STGs whose analysis is mainly restricted to visual examination, yet recently an interest in automated tools has arisen [Lar+12; WW20; SA21].

In computer science, STGs model the executions of automated systems. Computer scientists design automated tools called *model-checkers* to ensure the absence of bugs during software executions [Cla+18b]. Model-checkers verify whether the pathways within an STG satisfy a given property, for example that a desired state is reached or that harm-

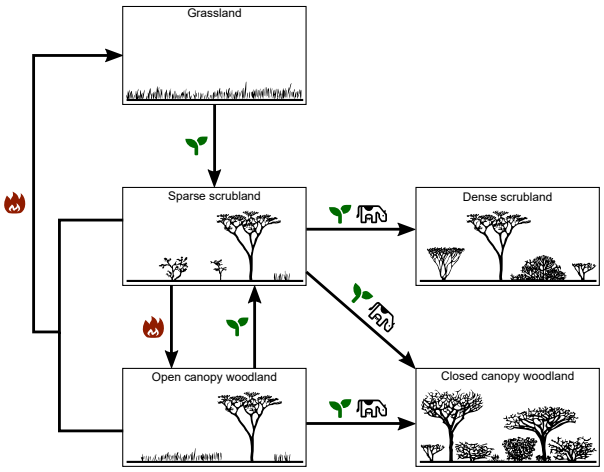


Figure 1.1: State-transition graph. The states are embodied by illustrated boxes and the transitions by arrows labelled with their main driving processes (🔥 for fire, 🌱 for plant recruitment, and 🐑 for intensive grazing). Adapted from [LC18].

ful behaviour is always avoided. Given a system description that can be computed into an STG and a dynamical property written as a *temporal logic formula*, a model-checker outputs whether the STG satisfies the property or not.

One main feature contributing to the success of model-checking is the design of *symbolic* model-checking algorithms [Bur+92] that mitigate the *combinatorial explosion problem* (i.e. the exponential growth of the number of states with the number of variables). Instead of explicitly enumerating the individual states, a symbolic algorithm computes its output by gathering sets of states into compact data structures [Bur+92; Cou+02]. In this thesis, we propose to arrange these symbolic sets of states inside a *component graph* that is a concise and graphical representation of a partition of the states of an STG.

Outline In this thesis, we argue that model-checking and component graphs are fitted for the analysis of ecological STGs. More precisely:

- The remainder of Chapter 1 gives some background for the concepts and case studies presented in this thesis.
- Chapter 2 gives formal definitions of state-transition graphs (STGs), component graphs, and CTL model-checking.
- Chapter 3 presents `ecco`, a software that provides modelling and analysis tools for ecological STGs.
- Chapter 4 presents two case studies using `ecco`: the vegetation successions of the Ethiopian region of Borana, and the laboratory assembly of protists communities.
- Chapter 5 extends the temporal logic ARCTL [PR07] with fairness, resulting in FARCTL, and provides a symbolic model-checking algorithm for FARCTL.
- Chapter 6 illustrates the insights that FARCTL can bring to the analysis of ecological STGs by applying it to the two case studies.
- Chapter 7 discusses the implementation choices and the perspectives.

- Chapter 8 proposes a short logbook about working on an interdisciplinary research subject.

An interdisciplinary thesis / How to read This thesis deals with an interdisciplinary topic mixing computer science and ecology. It was done between two laboratories, IBISC (Université Paris-Saclay, Univ Evry) for computer science and AMAP (INRAE) for ecology. This manuscript reflects this interdisciplinarity. It was written with the aim of being intelligible by both computer scientists and ecologists, explaining most technical concepts. The remarks directed at computer scientists are put inside light orange boxes titled "Technical remark", while the remarks directed at ecologists are put inside green boxes titled "Ecology". Readers interested in the computer science aspects of this work can skim through Chapter 4 and Chapter 6 which detail the ecological case studies. On the other hand, readers interested in the ecology aspects can skip the end of Chapter 5 which details the FARCTL symbolic algorithm.

1.1 State-transition graphs in ecology

1.1.1 A new concept?

Although the word "state-transition graph" is unusual in ecology, ecologists have drawn STGs for decades to represent the pathways between the discrete states of an ecosystem. Indeed one can draw STGs without having a formal definition in mind. Many ecological STGs were designed more as graphical summaries of the knowledge about the dynamics of the studied system rather than as actual data. Thus, STGs have been regularly drawn in ecology under various names , for example "behaviour graph" [Pat71], "kinematic graph" [Lon74], or "pattern" [Ber+14].

The lack of a unifying concept for STGs in ecology can be explained by the fact that STGs are found in historically isolated research fields. In ecology, the state of an ecosystem is often discretely abstracted by its *community* (i.e. restricted to its set of species or populations). Subsequently, STGs are found in a wide variety of studies focusing on the dynamics of ecological communities. But the dynamics of plant communities (*community succession*) and of animal communities (*community assembly*) were historically studied in isolation from one another [YCH01; CH16].

The intuitive nature of STGs and the isolation of their application fields resulted in the lack of a formal unifying concept encompassing every STG instance found in ecology. In this thesis, we propose the term "*state-transition graph*" (STG) to fill this conceptual gap. This term has a straightforward meaning, is already used in systems biology to represent the same concept [Abo+16], and is broad enough to encompass the particular features of existing STGs. Moreover, it is in lexical proximity to the existing "state-and-transition models" (STMs) which are a particular kind of ecological STGs, thus we hope that it will sound intuitive to most ecologists.

Regrouping the numerous existing ecological STGs inside a uniting conceptual framework

enables comparing their particular features and the design of common analysis tools. For example, this conceptual framework can be linked to the existing toolboxes designed in computer science and in systems biology [BČŠ13; BL16] to analyse such graphs. Moreover, the once isolated fields of community succession and community assembly are nowadays more and more integrated together in ecological studies [YCH01; CH16], thus the study of their STGs should also be integrated inside a common framework.

Therefore, "state-transition graph" is not a new concept in ecology, but rather a new name integrating existing objects and providing a common framework to think and to analyse them.

1.1.2 Background

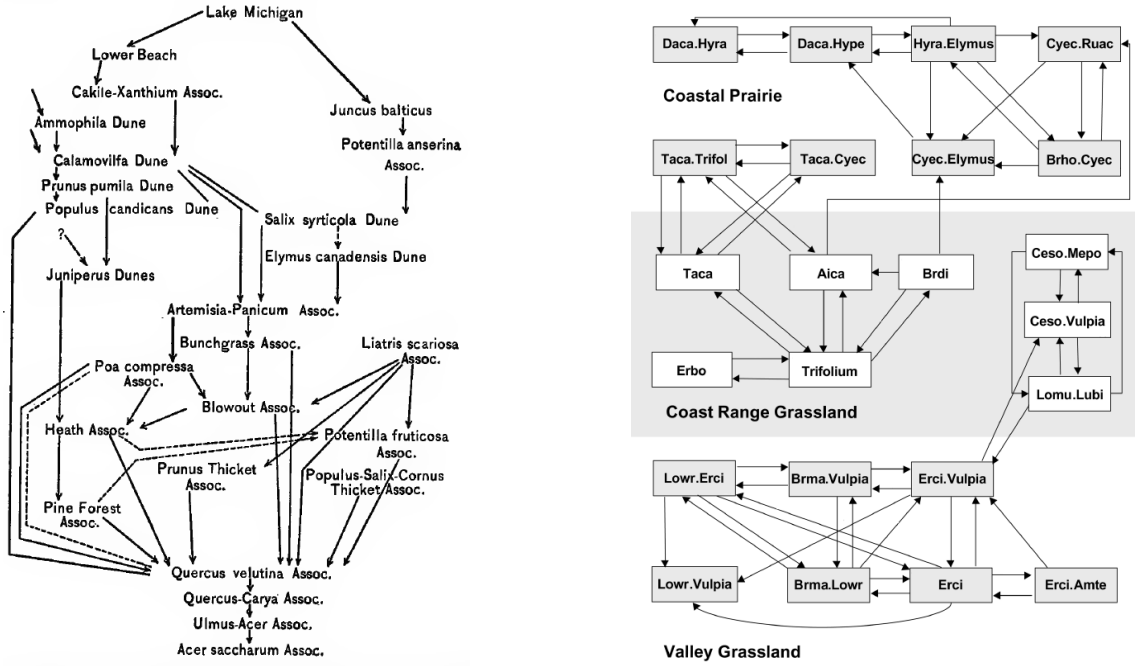
The history of ecological STGs can be traced back to the vegetation successions described by Clements at the beginning of the 20th century [Cle16]. Clements described how an ecological community (i.e. a group of species) changes following a disturbance or the initial colonization of the habitat. To that end, he drew STGs whose states were called "seral stages", whose most transitions were deterministic (i.e. with at most one outgoing transition for each state) and converged toward an end state called "climax community" [WWN89; YCH01]. Clements compared ecological successions with organism development, in his view an ecosystem develops itself toward the climax community that acts like a fixed phenotype. The theory of Clements was highly influential until the 1960s.

Modern theories describe much less deterministic ecological successions, with alternative pathways and multiple end states (or even no end state at all) [YCH01]. Chance and historical contingency are thought to play essential roles in ecological successions, which are thus considered less predictable. This evolution of the theory of ecological successions is reflected in the drawn STGs, as depicted in Figure 1.2.

Most STGs found in ecological succession studies (historically focusing on vegetation communities) are drawn from observations and thus have a relatively small size (a few dozen states at most). For example [CP02; Ber+14] depict the vegetation successions in boreal forests, and [Lon74] the successions of dune slack vegetation. In addition, vegetation succession studies also produced models outputting STGs, such as the "replacement sequences" of [NS77; Cat+79].

Other research fields in ecology, such as community assembly, produced STGs independently of the community succession field. For example, several studies outputted STGs from a wide diversity of modelling formalisms, such as boolean networks [Cam+11; RM16], timed automata [Lar+12; CLZ14], Petri nets [GP19], or qualitative reasoning [SB06].

We will now detail two actively studied STG formalisms: "state-and-transition models" stemming from community succession, and "assembly graphs" stemming from community assembly.



(a) Clements 1916 **(b)** Jackson and Bartolome 2002

Figure 1.2: STGs reflecting the evolution of the vegetation succession theory. (a) Vegetation succession of Lake Michigan from [Cle16, Fig.8 p.218], this STG converges toward a climax state (bottom-most state), showing primarily deterministic pathways. (b) Vegetation succession of Californian grasslands from [JB02, Fig.2], this STG shows alternative pathways without an end state.

State-and-transition models (STMs)

The most active research area using STGs and stemming from vegetation succession is the empirical *state-and-transition models (STMs)*. STMs are derived from observations and are designed to cope with the non-deterministic and irreversible nature of observed dynamics [WWN89]. An STM consists of both: (1) a catalogue of possible alternative states of the system; and (2) a catalogue of possible transitions between these states. STMs are not theoretical models of vegetation dynamics, but on the contrary are designed as an organisation of the empirical information suitable to the ecosystem management perspective [WWN89].

STMs aim to assist managers and scientists in collectively proposing policies driving the ecosystem through some desired pathways while avoiding others. In order to remain user-friendly [Bes+17], STMs' sizes usually do not exceed a few dozen states. While STMs originally stem from rangeland management [WWN89], they are now used in many fields such as natural park management [Cau13] (see for example the "EDIT" database housing a large catalogue of STMs [Bes+16]), geomorphology [PV17], or agroecology [Tit20].

Assembly graphs

The STGs stemming from the community assembly research field are called *assembly graphs* [HP93; SA21]. Compared to community succession, community assembly focuses on how a community assembles from a pool of species and on the short-term consequences of invasion events. In an assembly graph, every state is a stable community (i.e. a set of species), and every edge is an invasion by a species absent from the source state. Contrarily to STMs, most studies involving assembly graphs are theoretical [LM93; SFS21], yet a few are experimental [WLW03]. Indeed assembly graphs are used to answer theoretical questions about community assembly, while STMs are used to answer empirical questions about real-world ecosystem management.

1.2 Symbolic methods and model-checking

1.2.1 Background in computer science

Model-checking is an automated method for analysing any dynamical system that can be modelled by states and transitions [CHV18]. As depicted in Figure 1.3, its goal is to check that an automated system (hardware or software), given as a description that can be computed into an STG $\mathcal{G}$, satisfies a given dynamical property, usually written as a *temporal logic formula* φ. A *model-checker* is a software performing this operation, returning a *yes/no* output depending on whether the STG $\mathcal{G}$ satisfies the property φ or not, generally with a counterexample pathway for negative output. Most of the time, the queried property models a bug in the system's execution, thus the primary purpose of model-checking is to ensure that an automated system is bug-free.

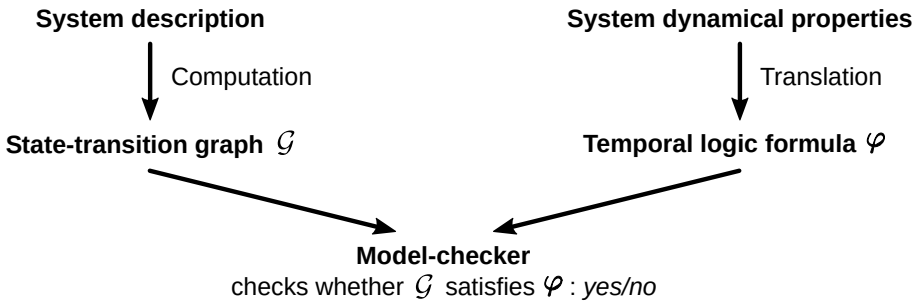


Figure 1.3: The model-checking methodology. Adapted from [CHV18].

Symbolic model-checking algorithms [Thi16] compute their output without explicitly exploring the state-transition graph. Instead of exploring the states one by one, symbolic algorithms operate on sets of states gathered into compact data structures such as *binary decision diagrams* [Bur+92]. Model-checking can thus be performed in a set-based setting, returning the set of states satisfying the queried property. The main benefit of symbolic model-checking algorithms is the mitigation of the combinatorial explosion problem (i.e. the exponential growth of the number of states with the number of variables) thanks to the compact data structures representing sets of states.

1.2.2 Background in biology

Although model-checking was developed to hunt bugs in software, it can analyse all kinds of STGs. In systems biology, STGs are outputted by models of reaction networks or regulatory networks [WSA12]. Model-checking is extensively used to analyse these models [BČŠ13; BL16], proving its suitability for studying biological systems. For example, model-checking helped to validate models of nutritional stress response of *Escherichia coli* [Bat+05], T-helper cell reprogramming [Abo+15], mammalian cell cycle [Tra+16] or BRAF inhibition pathways in two different cancers [Béa+21].

Yet, model-checking has been very scarcely used in ecology. So far, most formal analyses of STGs in ecology have been limited to graph measures [Phi11] and topology analyses [Cam+11; RM16; GP19]. We identified only a few precursory applications of model-checking in ecology [Lar+12; CLZ14; Bar+15]. These studies introduce a specific implementation of the model-checking methodology based on timed automata or P systems to model and analyse the dynamics of ecosystems, such as coral reef fisheries.

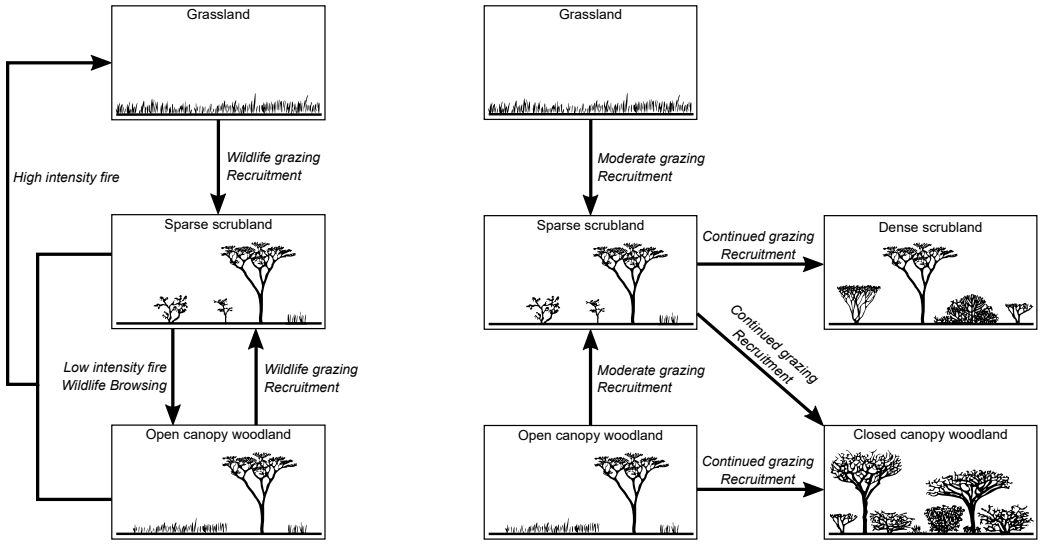
1.3 Case studies

Along this thesis, we will use two running examples taken from both STM and assembly graph literature, because they are the most active modern research fields using STGs. In this section, we present the ecological background behind these examples. These examples will then be modelled and analysed in Chapter 4 *Case studies* and Chapter 6 *Applications of FARCTL*.

1.3.1 Borana STMs

The STMs developed by Liao and Clark [LC18; LCD18; Lia+20; Lia16], see Figure 1.4, describe the vegetation pathways of the Borana Zone in southern Ethiopia. Open canopy woodland (a savanna-like vegetation class encompassing a grass layer with sparse trees) was historically the most prominent vegetation class in Borana [LCD18]. But since government banned the use of fire in the 1970s, the region has been undergoing a rapid increase in the density of woody plants (known as *bush encroachment*). As local people predominantly practice pastoralism, the reduction in herbaceous cover threatens their livelihood. Hence understanding the vegetation pathways is critical to help pastoralists and policymakers to mitigate bush encroachment [Lia+20].

The STMs' states represent vegetation classes [LCD18], the STMs' transitions are labelled by their main drivers, as is often the case in the STM framework. The STMs of Figure 1.4 showcase non-deterministic behaviour, i.e. some states have more than one outgoing transition, as it is often the case in ecological successions. Moreover, the STM representing bush encroachment (Figure 1.4b) exhibits *irreversible* pathways, i.e. one-way only, for example grasslands cannot be reached from any encroached state (dense scrubland or closed canopy woodland).



(a) Before livestock introduction **(b)** With livestock and fire ban

Figure 1.4: State-and-transition models of the Borana vegetation pathways. Taken from [LC18, Fig.5]. (a) Before pastoralism, fire was the main driver of the rangeland dynamics. The combination of fire, wildlife herbivory and vegetation recruitment maintained the entire system in a loop between open canopy woodland and grassland. (b) The presence of cattle and the fire ban gave a competitive advantage to woody plants, inducing an irreversible bush encroachment. Concurrently, wildlife increasingly avoided the Borana zone because of the denser human and livestock populations.

Accordingly to the STM perspective, the STMs of Figure 1.4 compare various ecosystem management policies in order to help maintaining herbaceous cover. They are drawn from empirical observations, and their sizes are limited to remain user-friendly. They were designed for the purpose of answering questions such as:

- *Which management policies prevent bush encroachment?*
- *Is bush encroachment reversible without changing management policy?*
- *Can a change in management policy reverse the bush encroachment induced by the preceding one?*

1.3.2 Protist assembly graph

The two laboratory experiments performed by Law, Warren and Weatherby [WWL98; WLW03] studied how microcosms assemble through time from a pool of 6 *protist* species (microscopic unicellular eukaryotes) feeding on a bacterial mixture. The first experiment [WWL98] studied how each combination of species (i.e. community), taken from the 2^6 possible ones, behaved through time. For each possible community, the authors made 6 replicates and observed whether the community persisted or collapsed, i.e. whether the community is *stable* or not. The outcomes differed between replicates for some communities, as depicted in Figure 1.5.

The second experiment [WLW03] studied how each stable community, found in the first

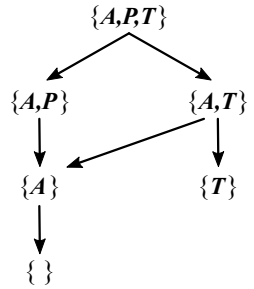


Figure 1.5: Community pathways from the first protist experiment. Each state represents a community (i.e. a set of species), labelled with the initials of the present species between brackets. Each transition represents an observed community change. This STG depicts the community pathways observed from the $\{A, P, T\}$ community. Taken from [Her+22, Fig.2].

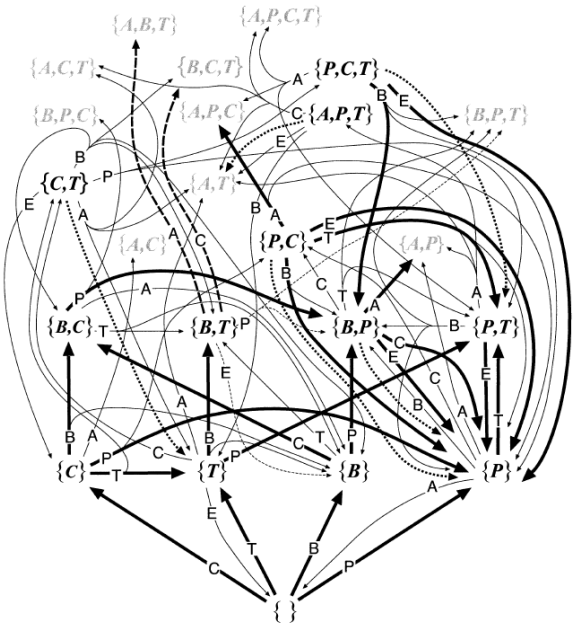


Figure 1.6: Assembly graph from the second protist experiment. Each state represents a community (i.e. a set of species), labelled with the initials of the present species between brackets. Black states were found stable in the first experiment, and thus were subjected to the invasion experiment; while grey states were found stable after invasion but not in the first experiment, and thus were not subjected to invasion. The transitions represent either invasion events (plain or dashed arrows labelled by the initial of the invading species), or collapses of the community without any invasion (dotted arrows without label). Taken from [WLW03, Fig.3].

experiment, responds to an invasion by another species from the pool [LWW00]. For each stable community and invading species, between 4 and 6 replicates were made depending on experimental constraints. As with the first experiment, the outcomes sometimes differed between replicates, resulting in branching arrows in the assembly graph depicted in Figure 1.6. To our knowledge, these experiments are unique in being the only replicated analysis which systematically explores the fate of all possible communities that can be built from a pool of species.

These experiments were designed for the purpose of answering questions such as:

- *From how many initial states are each of the stable communities obtained?* [WWL98]
- *What are the impacts of individual species on community collapse?* [WWL98]
- *Are there* catalytic *species, that invade, change the community, and then go extinct?* [WLW03]

1.4 Contributions

This thesis presents formal methods based on model-checking for the modelling and the analysis of ecological state-transition graphs. First, we introduce the concept of ecological state-transition graph that, while being a novelty, captures a long history of disparate representations of the dynamics of an ecosystem as a graph [Tho+22]. Then, we propose an analysis methodology based on the partitioning of the states of an STG using model-checking, which results in component graphs. This methodology is implemented inside ecco [PTG22a], a Python toolbox developed by Cédric Gaucherel and Franck Pommereau for the formal modelling and the analysis of ecosystems. This approach is exemplified in two case studies: (1) vegetation changes of the Borana Zone in Ethiopia under diverse management scenarios [LC18; LCD18; Lia+20; Lia16], and (2) protists community assembly [WWL98; WLW03]. We model and analyse the Borana vegetation dynamics [Tho+22], while Mathieu de Goër de Herve modelled and analysed the protists experiments [Her+22] (the analysis has been redone and extended in the symbolic perspective presented in this thesis). Both case studies are limited by the fact that we would want some specific events, for example changes in management scenarios or species invasions, to occur only in a controlled manner. This limitation can be overcome using Action-Restricted CTL (ARCTL) [PR07], an extension of CTL that allows to restrict the set of enabled actions along a formula. We extend ARCTL with fairness constraints, i.e. "realism" constraints upon order and happening rate of events along a path, resulting in Fair ARCTL (FARCTL). For this new temporal logic, we provide a symbolic model-checking algorithm, that we implemented inside ecco. Finally, we apply FARCTL to both case studies, to investigate the consequences of shifting between management scenarios, and to look for specific invasion behaviours.

2 Definitions

Summary

This chapter provides the formal definitions of the main concepts used throughout the thesis. First, we define state-transition graphs and maximal paths within them. Then we define component graphs, that is graphs representing partitions of the states of an STG. Lastly, we define the model-checking problem and the CTL temporal logic, in terms of both individual states (explicit perspective) and sets of states (symbolic perspective). We also provide a catalogue of query patterns translated into CTL, intending to ease the design of CTL formulas describing the system's behaviour.

2.1 State-transition graph

First, we give a definition of State-Transition Graphs (STGs) encompassing labelled transitions. This definition is very comprehensive in order to cover every case, even if particular instances may lack some features (although every instance presents at least a set of states and a transition relation). Various examples of ecological STGs are given throughout this thesis.

Definition 2.1 (STG $\mathcal{G}$)**.** Given a set of atomic state properties $\mathcal{P}_\mathcal{S}$ and a set of atomic action properties $\mathcal{P}_\mathcal{A}$, a *state-transition graph (STG)* is a tuple $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \rightarrow, \mathcal{V}_\mathcal{S}, \mathcal{V}_\mathcal{A})$, where:

- $\mathcal{S}$ is a finite set of states
- $\mathcal{S}_0 \subseteq \mathcal{S}$ a set of initial states
- $\mathcal{A}$ is a finite set of actions
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a transition relation
- $\mathcal{V}_\mathcal{S} : \mathcal{S} \mapsto 2^{\mathcal{P}_\mathcal{S}}$ maps every state to its atomic properties
- $\mathcal{V}_\mathcal{A} : \mathcal{A} \mapsto 2^{\mathcal{P}_\mathcal{A}}$ maps every action to its atomic properties

Ecology

Graphs are widespread in ecology, but STGs must be discriminated from *interaction networks* such as the iconic trophic networks [May06; Pil+17]. Indeed, the former grasp the temporal behaviour of an ecological system, while the latter grasp the processes taking place between its components. A node (resp. an edge) of an STG is a temporal stage (resp. an event, i.e. a state transition) of the system dynamics. Whereas a node (resp. an edge) of an interaction network is an entity (resp. a flux) of

the system. The methods presented in this thesis deal with the temporal changes of a discrete-event system and thus are designed to analyse STGs specifically.

Note also that despite their orthographic proximity, "STG" and "STM" shall not be mixed up. While STGs refer to a general mathematical concept, STMs are special instances of STGs linked to a particular research field with particular features and purposes (see Section 1.1.2).

We note $s \xrightarrow{a} s'$ the fact that $(s, a, s') \in \rightarrow$. A state without outgoing transition is called a *dead-end*, and we note $s \nrightarrow$ the fact that $\nexists (s, a, s') \in \rightarrow$.

Example 2.1. Not every STG found in ecology literature matches exactly definition 2.1. For example, in the STG of Figure 2.1, we can naturally consider that:

- $\mathcal{S} = \{\text{Grassland}, \text{Sparse Scrubland}, \text{Open canopy woodland}, \ldots\}$.
- $\mathcal{P}_{\mathcal{A}} = \{$🔥, 🌱, 🐑$\}$.

But for the other features:

- $\mathcal{S}_0$ is not defined, thus we can take every subset of $\mathcal{S}$ as initial states, for example we can take $\mathcal{S}_0 = \mathcal{S}$.
- $\mathcal{P}_{\mathcal{S}}$ is not clearly defined, and consequently $\mathcal{V}_{\mathcal{S}}$ is not defined either, yet states have various properties such as the presence of trees.
- $\mathcal{A}$ is not clearly defined but we can use the transition labels $\mathcal{A} = \{$🔥, 🌱, 🌱🐑$\}$, and define $\mathcal{V}_{\mathcal{A}}$ and $\rightarrow$ accordingly.

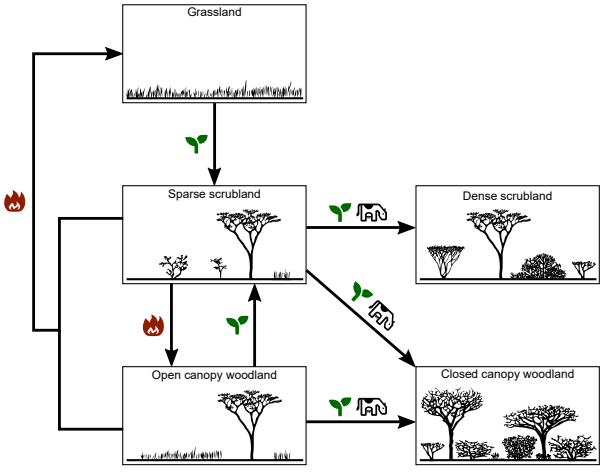


Figure 2.1: Example state-transition graph. Adapted from [LC18].

In computer science, the state and transition description of the system is most of the time defined as a *Kripke structure* (KS) or as a *labelled transition system* (LTS) [Cla+18b], depending if either its states or its transitions are labelled. Our STG definition is equivalent to *mixed transition system* [PR07], combining both state and action labels. As in systems biology [Abo+16; Tra+16], we will keep calling it state-transition graph (STG) for clarity.

Technical remark

Of course, an STG $\mathcal{G}$ can always be projected to a KS sub-structure $(\mathcal{S}, \mathcal{S}_0, \to', \mathcal{V}_\mathcal{S})$ by considering $\to' = \{(s, s') \mid (s, a, s') \in \to\}$, or to a LTS sub-structure $(\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \to, \mathcal{V}_\mathcal{A})$ by forgetting the state labels. Conversely, any KS or LTS can also be extended into an STG by adding empty state or action labels. Thus the methods developed for these descriptions can also be applied to STGs, and vice versa. In the following, $\mathcal{G}$ is always an STG combining state and action labels, unless specified otherwise.

Most questions about STGs focus on their induced pathways. A path π is a finite or infinite sequence of states and transitions: $s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots$. The length of a path π, noted by $|\pi|$, is its number of transitions. A path π of length $|\pi| = 0$ is restricted to a single state s_0, a path π of length $|\pi| = \infty$ is an infinite sequence of states and transitions. For $i \leqslant |\pi|$ and $i \in \mathbb{N}$, the i-th state of π is noted $\pi[i]_\mathcal{S}$, and its i-th action is noted $\pi[i]_\mathcal{A}$. We note $s \to^* s'$ the fact that there is a path from s to s', and $s \to^+ s'$ the fact that there is a path of length $\geqslant 1$ from s to s'.

Definition 2.2 (Π)**.** A *maximal path* π is an infinite sequence of states and transitions, or a finite sequence ending in a dead-end $s \not\to$. The set of all maximal paths starting from a state $s \in \mathcal{S}$ is noted $\Pi(s)$, the set of all maximal paths of $\mathcal{G}$ is noted $\Pi(\mathcal{G})$.

Ecology

In this thesis, we use the words “path”, “pathway” or “trajectory” interchangeably to designate arbitrary sequences of state and transitions. Indeed, all these names are used informally in ecology to represent this concept. Conversely, “maximal path” is a formal concept that refers to paths that cannot be further extended. Maximal paths are at the very heart of the model-checking framework. In a maximal path, the system cannot remain indefinitely in a state if there is an available transition: the maximal path must be extended and so the system must move on. Note that any arbitrary path can always be extended into one or many maximal paths. Thus, we can use model-checking to investigate the properties of arbitrary paths by investigating the properties of the beginning of maximal paths.

Example 2.2. If s is a dead-end $s \not\to$, then $\Pi(s)$ consists of a single path of length 0: $\Pi(s) = \{s \not\to\}$.

We define two simple dynamical properties of STGs:

Definition 2.3. An STG $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \to, \mathcal{V}_\mathcal{S}, \mathcal{V}_\mathcal{A})$ is *deterministic* if each state has at most one outgoing transition:

$$\forall s \in \mathcal{S} \quad \text{there is at most one } (s, a, s') \in \to$$

Every state of a deterministic STG is the start of a single maximal path, thus the behaviour of the system is also deterministic.

Technical remark

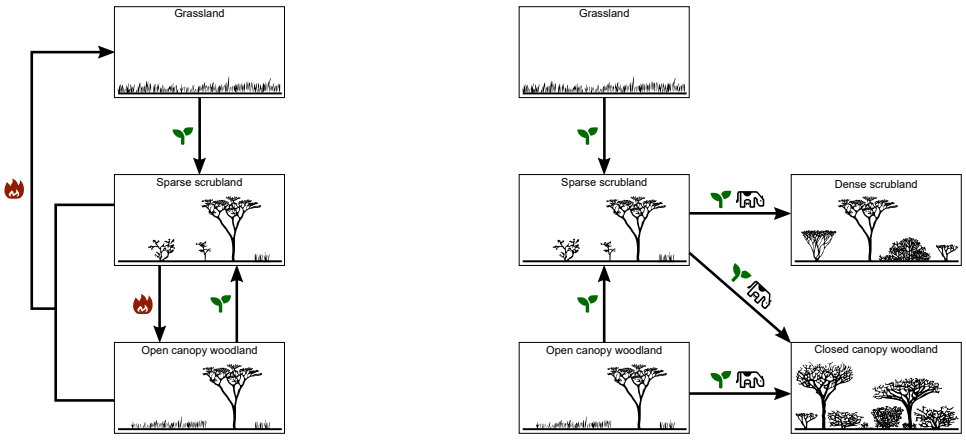
In other words, an STG is deterministic if its projection to a KS substructure is deterministic. An other definition of determinism could be that an STG is $\mathcal{A}$-deterministic if $\forall a \in \mathcal{A}$ each state has at most one outgoing a-transition: $\forall s \in \mathcal{S}$ there is at most one $(s, a, s') \in \rightarrow$.

Definition 2.4 (SCC)**.** An STG $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \rightarrow, \mathcal{V}_\mathcal{S}, \mathcal{V}_\mathcal{A})$ is *strongly connected* if every state is reachable from every other state:

$$\forall s \neq s' \in \mathcal{S} \quad s \rightarrow^+ s'$$

In a strongly connected STG, for every pathway leading from s to s', there is also a pathway leading from s' to s. Thus every state change can be reversed.

Example 2.3. Figure 2.2 presents two STGs derived from Figure 1.4. The STG of Figure 2.2a is strongly connected, i.e. every state can be reached from every other state. Consequently, every path can be reversed and thus every state change can be undone. Note that the reversed pathway may not be a simple backtracking of the original one. For example, *Open canopy woodland* $\rightarrow$ *Grassland* cannot be simply backtracked: *Grassland* $\not\rightarrow$ *Open canopy woodland*, but a reverse pathway exists: *Grassland* $\rightarrow$ *Sparse scrubland* $\rightarrow$ *Open canopy woodland*. Conversely, in an STG that is not strongly connected, such as the one of Figure 2.2b, some state change cannot be undone. For example, the pathway representing bush encroachment *Grassland* $\rightarrow$ *Sparse scrubland* $\rightarrow$ *Closed canopy woodland* cannot be reversed.



(a) Strongly connected **(b)** Not strongly connected

Figure 2.2: Example of strongly connected or not strongly connected STGs. Adapted from [LC18]. (a) The STG is strongly connected, thus every path can be reversed. (b) The STG is not strongly connected, thus some path cannot be reversed.

2.2 Component graph

The larger an STG is, the more difficult it is to analyse. In this thesis, we propose to build component graphs that partitions $\mathcal{S}$ into components according to some chosen properties of interest. Instead of representing each state $s \in \mathcal{S}$ individually, a component graph only represents the partition of $\mathcal{S}$ and is thus a condensed representation of $\mathcal{G}$ according the chosen properties.

Definition 2.5 ($\mathcal{C}$)**.** A partition $\mathcal{C}$ of a set $\mathcal{S}$ is a set of subsets of $\mathcal{S}$ such that every element of $\mathcal{S}$ is included in exactly one element of $\mathcal{C}$:

$$\bigcup_{c \in \mathcal{C}} c = \mathcal{S} \quad \text{and} \quad \forall c_1, c_2 \in \mathcal{C} \quad c_1 \neq c_2 \Rightarrow c_1 \cap c_2 = \emptyset$$

For $s \in \mathcal{S}$, we note $\langle s \rangle_{\mathcal{C}}$ the element of $\mathcal{C}$ such that $s \in \langle s \rangle_{\mathcal{C}}$.

In the following, we partition the set of states $\mathcal{S}$ of an STG $\mathcal{G}$, and we call *components* the elements of this partition. Given a partition $\mathcal{C}$ of $\mathcal{S}$, the states belonging to the same component can be merged into a single state, resulting in a compact representation of $\mathcal{C}$:

Definition 2.6 ($\mathcal{G}/\mathcal{C}$)**.** Given an STG $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \rightarrow, \mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{A}})$ and a partition $\mathcal{C}$ of $\mathcal{S}$ into components, the *component graph* $\mathcal{G}/\mathcal{C}$ is the quotient STG $\mathcal{G}/\mathcal{C} = (\mathcal{C}, \mathcal{C}_0, \mathcal{A}, \rightarrow_{\mathcal{C}}, \mathcal{V}_{\mathcal{C}}, \mathcal{V}_{\mathcal{A}})$ where:

- $\mathcal{C}_0 = \{\langle i \rangle_{\mathcal{C}} \mid i \in \mathcal{S}_0\}$
- $\rightarrow_{\mathcal{C}} = \{(\langle s \rangle_{\mathcal{C}}, a, \langle s' \rangle_{\mathcal{C}}) \mid (s, a, s') \in \rightarrow \text{ and } \langle s \rangle_{\mathcal{C}} \neq \langle s' \rangle_{\mathcal{C}}\}$
- $\mathcal{V}_{\mathcal{C}} : \mathcal{C} \mapsto 2^{\mathcal{P}_{\mathcal{S}}}$ maps every component c to the atomic properties that are true for every state $s \in c$, i.e. $\mathcal{V}_{\mathcal{C}} : c \mapsto \bigcap_{s \in c} \mathcal{V}_{\mathcal{S}}(s)$.

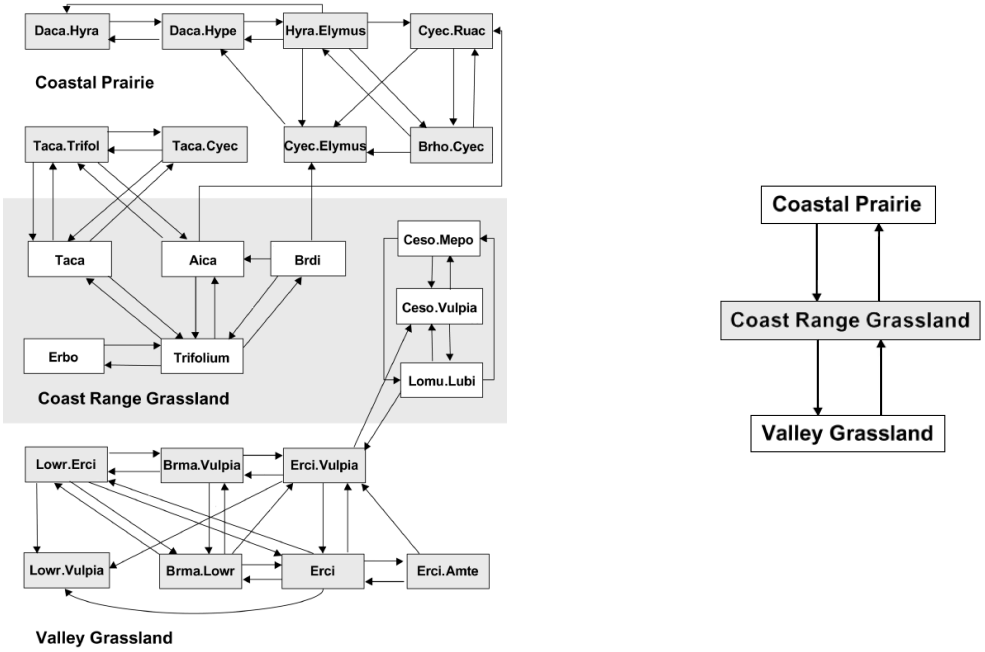
Technical remark

Labelling the components is not straightforward, as a component can embrace simultaneously states labelled with and states labelled without any given label. Thus we could give various definitions of the component labelling function $\mathcal{V}_{\mathcal{C}}$, for example labelling the components if all states are labelled accordingly as in the definition, or labelling the components if at least one state is labelled accordingly.

Example 2.4. The simplest component graph has a single component: $\mathcal{C} = \{\mathcal{S}\}$, and the most detailed component graph has only singleton components: $\mathcal{C} = \{\{s\} \mid s \in \mathcal{S}\}$ which is equivalent to the STG itself. Yet, ecologists are primarily interested in something in between with sufficiently few components to remain human-readable, but with enough details to exhibit critical aspects of the dynamics.

Example 2.5. In existing ecological STGs, the states are sometimes already partitioned according to some properties. For example, in Figure 2.3a the states are partitioned according to vegetation types into 3 components: *Coastal Prairie, Coast Range Grassland* and *Valley Grassland*. The states belonging to the same component can be merged into a single node, resulting in a component graph, see Figure 2.3b. Note that every

path in the STG matches an equivalent path in the component graph, but every path in the component graph does not necessarily correspond to a path in the STG. For example there is no path from a state of the *Coastal Prairie* component toward a state of the *Valley Grassland* component.



(a) Jackson and Bartolome 2002 **(b)** Component graph

Figure 2.3: Example component graph. (a) Vegetation successions of Californian grasslands from [JB02, Fig.2]. The partition of the states into three components is displayed by the colour of the states and background. (b) The corresponding component graph where each component is merged into a single node.

In a component graph, every node is a component, and every transition between two components represents a transition between two states belonging to either components. Each component can be represented *symbolically*, i.e. by a compact data structure that does not enumerate individual states but represents a set of states (for example binary decision diagrams). Thus a component graph is an *hybrid* object: combining symbolic sets (the components) with their explicit relations (the transitions). Such hybrid objects can provide efficient representations of large STGs [Bér+13; PTG22a].

2.3 CTL model checking

Model-checking is an automated method for analysing any dynamical system that can be modelled by states and transitions [CHV18]. A *model-checker* checks that an STG $\mathcal{G}$, satisfies a given dynamical property, usually written as a *temporal logic formula* φ, as depicted in Figure 2.4.

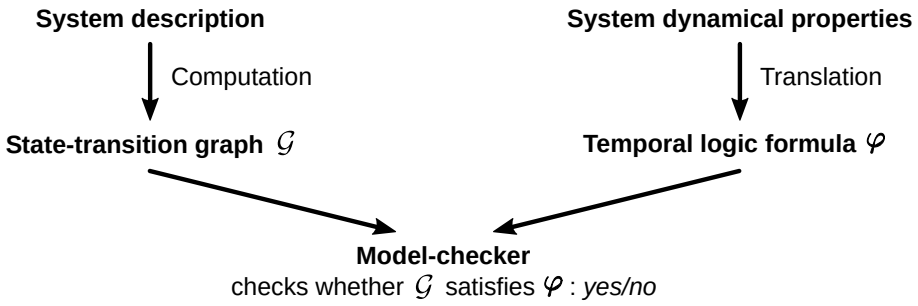


Figure 2.4: The model-checking methodology. Adapted from [CHV18].

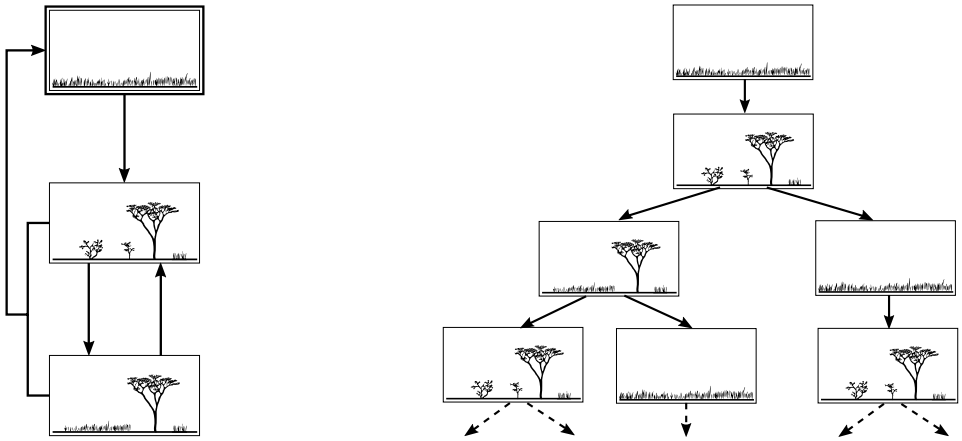
Ecology

The system description, i.e. the mathematical model from which the STG $\mathcal{G}$ is computed, see Figure 2.4, is not mandatory in the model-checking methodology. Indeed, an STG $\mathcal{G}$ can simply be given as input to a model-checker. Complex ecological STGs can be found directly inside empirical studies [WLW03; Bar+18], without being computed from any underlying mathematical description. Hence, model-checking is not only a tool for the analysis of mathematical models, but can also assist the automated investigation of empirical data.

Symbolic model-checking algorithms [Thi16] operate on sets of states without explicitly exploring the state-transition graph (hence the adjective "symbolic" as opposed to "explicit"). These algorithms use compact data structures such as *binary decision diagrams* [Bur+92] to gather a set of states without explicitly enumerating them, and use fixed point calculation to compute their results. Model-checking of state-based temporal logics can thus be performed in a set-based setting, returning the set of states satisfying the queried property. The main benefit of symbolic model-checking algorithms is the mitigation of the combinatorial explosion problem (i.e. the exponential growth of the number of states with the number of variables). In this thesis, we will also use the symbolic perspective of model-checking to partition the states into components, as discussed above.

2.3.1 Computation Tree Logic (CTL)

Computation Tree Logic (CTL) is one of the most popular temporal logics [Cla+18b], because it is particularly fitted to express properties of branching dynamics with alternative pathways. In this thesis, we focus on CTL (and some of its extensions in Chapter 5) because ecological STGs often involve such alternative pathways. Moreover, CTL is easily expressed using sets of states under the symbolic perspective [Bur+92]. A CTL formula describes a property over *computation trees*, noted CT*s* as at the beginning of CTL. A CT is rooted at a given state $s \in \mathcal{S}$ of the STG, and its branches are the alternative maximal paths $\Pi(s)$ starting from s, see Figure 2.5). In computer science, an STG represents the behaviours of an automated software system, thus every branch of a CT represents an alternative software computation, hence the name "computation tree". A CTL model-checker checks whether the CT rooted in each state satisfies a CTL formula or not. Thus, a



(a) STG **(b)** Computation tree rooted in the top state

Figure 2.5: Computation tree example. (b) The STG of Figure 1.4a. (a) The computation tree rooted at the state depicted with a double border in the STG.

CTL model-checker can discriminate between the states whose CT satisfies a given property and these whose CT does not.

CTL formulas are built by combining state properties, logical operators (such as $\neg, \wedge, \vee$) and temporal operators (such as $\forall$X or $\exists$G). Temporal operators define properties of the computation tree. For example, $\forall \mathsf{X}\varphi$ means that all successor states of the CT's root satisfy φ, while $\exists \mathsf{G}\varphi$ means at least one branch of the CT satisfies φ all along. Temporal operators always combine a *quantifier* ($\exists$ or $\forall$) dealing with path branching, and a *modality* (X, F, G, or U) specifying path properties.

First, we give the *syntax* of CTL, i.e. how state properties and operators (logical or temporal) can be combined into a formula.

Definition 2.7 (CTL syntax)**.** The *syntax* of *CTL* is given by the following grammar over state and path formulas:

- state formulas: $\varphi \stackrel{\text{def}}{=} \top \mid p \in \mathcal{P}_{\mathcal{S}} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists\gamma \mid \forall\gamma$
- path formulas: $\gamma \stackrel{\text{def}}{=} \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid \varphi_1 \mathsf{U} \varphi_2$

A CTL formula is a state formula φ.

Now let us give the *semantics* of CTL, i.e. the formulas' meaning. For $s \in \mathcal{S}$ and φ a state formula, we note $s \models \varphi$, the fact that the computation tree rooted at s satisfies φ. Similarly, we note $\pi \models \gamma$ the fact that a maximal path π satisfies a path formula γ.

Definition 2.8 (CTL semantics)**.** Given an STG $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \rightarrow, \mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{A}})$, let $s \in \mathcal{S}$ a state:

- $s \models \top$
- $s \models p$ iff $p \in \mathcal{V}_{\mathcal{S}}(s)$
- $s \models \neg\varphi$ iff $s \not\models \varphi$

- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$
- $s \models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$
- $s \models \exists\gamma$ iff $\exists\pi \in \Pi(s)$ such that $\pi \models \gamma$
- $s \models \forall\gamma$ iff $\forall\pi \in \Pi(s)$ we have $\pi \models \gamma$

Let $\pi \in \Pi(\mathcal{G})$ a maximal path:

- $\pi \models \mathsf{X}\varphi$ iff $|\pi| \geqslant 1$ and $\pi[1]_{\mathcal{S}} \models \varphi$
- $\pi \models \mathsf{F}\varphi$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi|$ and $\pi[i]_{\mathcal{S}} \models \varphi$
- $\pi \models \mathsf{G}\varphi$ iff $\forall i \leqslant |\pi|$ we have $i \in \mathbb{N} \Rightarrow \pi[i]_{\mathcal{S}} \models \varphi$
- $\pi \models \varphi_1 \mathsf{U} \varphi_2$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi| \quad \pi[i]_{\mathcal{S}} \models \varphi_2$ and $\forall 0 \leqslant j < i \quad \pi[j]_{\mathcal{S}} \models \varphi_1$

A *state property* $p \in \mathcal{P}_{\mathcal{S}}$ is a Boolean property mapping over states, $s \models p$ iff s is labelled by p. Complex state descriptions are built by combining state properties using the *Boolean logical operators*: *not* ($\neg$), *and* ($\wedge$), *or* ($\vee$). Other Boolean logical operators can be built on top of the three ones above, such as the implication ($\Rightarrow$) defined as $p \Rightarrow q$ is equivalent to $(\neg p) \vee q$.

The *temporal operators* of CTL are always the combination of two types of operators: first a *quantifier* ($\exists$ or $\forall$) dealing with branching by quantifying over the paths starting from a given state, second a *modality* (X, F, G, or U) specifying the order of properties along a path, see Figure 2.6. Modality X species that the property is true in the *neXt* state of the path. Modality F specifies that the property *Finally* becomes true at one step of the path. Modality G specifies that the property is *Globally* true all along the path. Modality U specifies that the left-hand-side property remains true along the path *Until* the right-hand-side property finally becomes true. Indeed the syntax of CTL enforces that state and path formulas must alternate when nested.

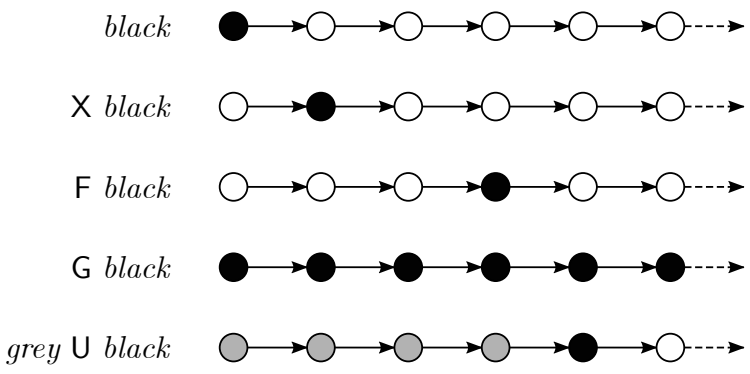


Figure 2.6: Semantics of the temporal modalities. Adapted from [BK08].

Temporal operators can be separated between existential and universal operators. *Existential operators* ($\exists$X, $\exists$F, $\exists$G, or $\exists$U) specify that their modality has to be verified by *at least one branch* of the CT (thus by at least one pathway of the STG starting from its root state). *Universal operators* ($\forall$X, $\forall$F, $\forall$G, or $\forall$U) specify that their modality has to be verified by *every branch* of the CT (thus by every pathway of the STG starting from its root state). Examples of computation trees satisfying basic CTL formulas are given in Figure 2.7.

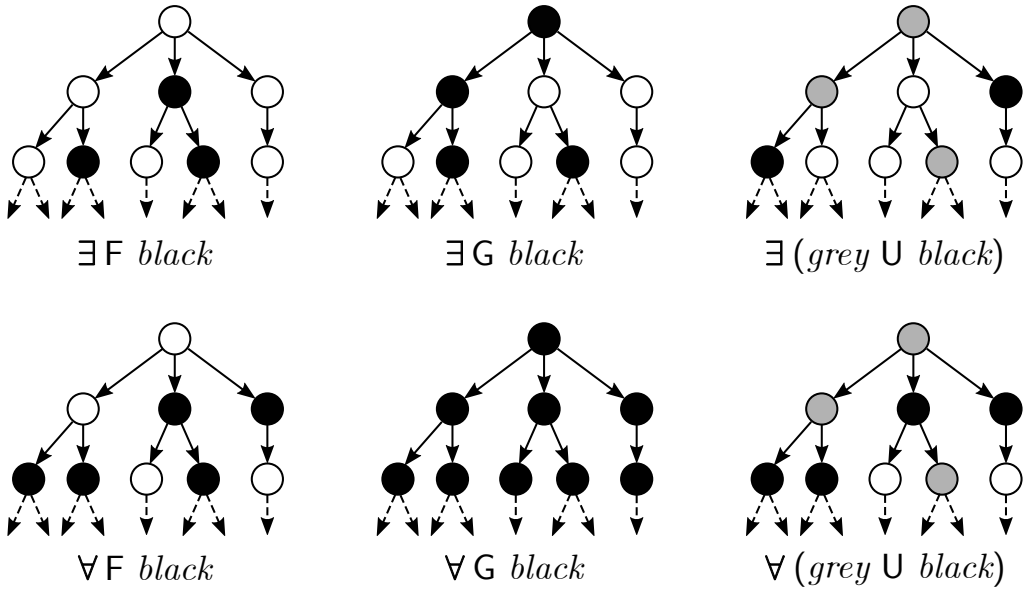


Figure 2.7: Example of computation trees satisfying basic CTL formulas. Adapted from [BK08].

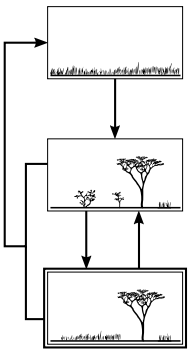
Example 2.6. The two CTs of Figure 2.8 have distinct properties:

- the CTL formula "$\exists \mathsf{F}\, \neg$🌳" specifies that at least one state without trees is reachable from the root of the CT, which is satisfied in Figure 2.8b but not in Figure 2.8d;
- the CTL formula "$\exists \mathsf{G}$ 🌳" specifies that trees are always present along at least one branch of the CT, which is satisfied for the left-most branch in Figure 2.8b but not for its other branches, thus this CTL property is satisfied in both Figure 2.8b and Figure 2.8d;
- the CTL formula "$\forall \mathsf{G}$ 🌳" specifies that trees are always present all along every branch of the CT, which is satisfied in Figure 2.8d, but not in Figure 2.8b.

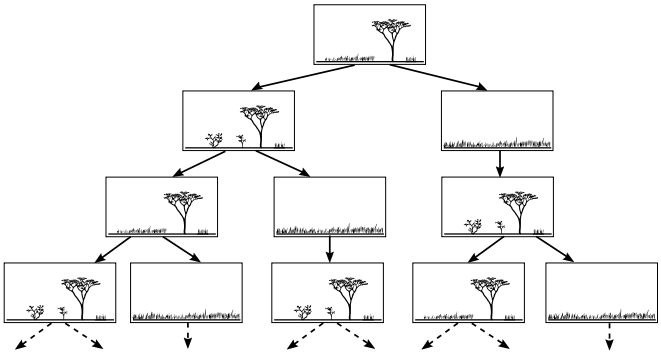
Lastly, CTL operators can be nested to express even subtler temporal behaviour. For example, $\forall \mathsf{G}(\exists \mathsf{F}\, \neg$🌳$)$ specifies that: all along every path ($\forall \mathsf{G}$), the path can always branch off to reach a future state ($\exists \mathsf{F}$) without trees ($\neg$🌳). While $\forall \mathsf{G}(\exists \mathsf{F}\, \neg$🌳$)$ holds in Figure 2.8b, the simpler property $\forall \mathsf{F}\, \neg$🌳 does not because trees never disappear in the left-most branch of the CT.

Technical remark

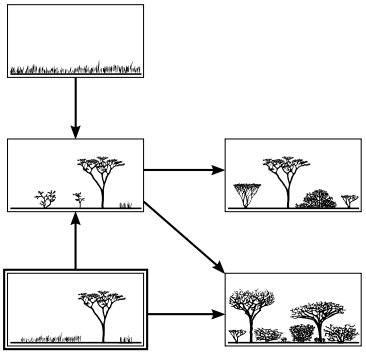
We defined CTL over *finite* and *infinite* maximal paths, whereas usual definitions [BK08; Cla+18b] deal only with infinite maximal paths. Indeed STGs with dead-ends, and thus finite maximal paths, are commonplace in ecology. Our definition considers finite maximal paths as infinite paths staying indefinitely in the dead-end. Thus in a dead-end $s \not\to$, $s \models \exists \mathsf{G}\varphi$ iff $s \models \varphi$. In contrast, we can define $\exists \mathsf{G}^{\infty}\varphi$ that holds only for infinite maximal paths: $\exists \mathsf{G}^{\infty}\varphi \stackrel{\text{def}}{=} \exists \mathsf{G}(\varphi \wedge \exists \mathsf{X} \top)$. Note that



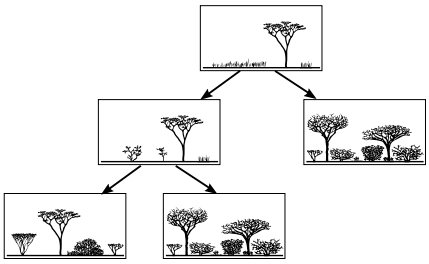
(a) Before livestock introduction [LC18].



(b) Computation tree rooted at the savanna state



(c) With livestock and fire ban [LC18].



(d) Computation tree rooted at the savanna state

Figure 2.8: Borana computation trees for various scenarios. The two STGs (a,c) represent the Borana vegetation pathways under two distinct management scenarios, see Figure 1.4 for details. The two CTs (b,d) are rooted at the savanna states (open canopy woodland) of the corresponding STGs, depicted with a double border. As the maximal paths are finite in the STG (c), the branches of the CT (d) are also finite.

our definition of CTL and the classical one are strictly equivalent for infinite maximal paths.

Note also that in a dead-end $s \nrightarrow$, there is only a single maximal path $\Pi(s) = \{s \nrightarrow\}$. As the length of this path is zero, neither $\exists \mathsf{X}\varphi$ nor $\forall \mathsf{X}\neg\varphi$ are satisfied. Therefore, contrarily to the classical semantics of CTL for infinite maximal paths, $\forall \mathsf{X}\neg\varphi \neq \neg\exists \mathsf{X}\varphi$.

Theorem 2.1. For any $\varphi \in CTL$ and any state $s \in \mathcal{S}$, one and only one of $\exists \mathsf{X}\varphi$, $\forall \mathsf{X}\neg\varphi$ and $\neg\exists \mathsf{X}\top$ holds in s. Thus they can be rewritten into one another:

$$\begin{aligned} \exists \mathsf{X}\varphi &= \neg(\forall \mathsf{X}\neg\varphi \vee \neg\exists \mathsf{X}\top) \\ \forall \mathsf{X}\neg\varphi &= \neg(\exists \mathsf{X}\varphi \vee \neg\exists \mathsf{X}\top) \\ \neg\exists \mathsf{X}\top &= \neg(\exists \mathsf{X}\varphi \vee \forall \mathsf{X}\neg\varphi) \end{aligned}$$

Proof. If s is a dead-end $s \nrightarrow$, then $\Pi(s) = \{s \nrightarrow\}$ but $s \nrightarrow \not\models \mathsf{X}\top$ because $|s \nrightarrow| = 0$,

consequently $s \not\models \exists \mathsf{X} \top$, i.e. $s \models \neg \exists \mathsf{X} \top$. Otherwise, either $s \models \exists \mathsf{X} \varphi$, or $s \models \forall \mathsf{X} \neg \varphi$ because $\forall \pi \in \Pi(s) \quad |\pi| \geqslant 1$. □

In computer science, model-checking is used either to test that specifications are met or to look for the presence of bugs. $\mathcal{G}$ represents the automated system being tested, and φ describes specifications or bugs. The model-checking problem consists in determining whether all paths in $\mathcal{G}$ meet the specifications, or whether at least one path in $\mathcal{G}$ presents a buggy behaviour (or said differently that all paths $\mathcal{G}$ do not present a buggy behaviour).

Definition 2.9. Given an STG $\mathcal{G}$ and a CTL formula φ, the *CTL model-checking problem* consists in determining whether $\forall s_0 \in \mathcal{S}_0 \quad s_0 \models \varphi$, noted $\mathcal{G} \models \varphi$.

Ecology

Model-checking is a multipurpose tool that can be used both to investigate the temporal behaviour of STGs (representing empirical data or resulting from modelling) and to validate models outputting STGs. Since model-checking is automated, it can process STGs too large to be examined by hand. For example, CTL model-checking was applied in systems biology to STG models made up of hundreds of variables [CF03].

Human work is then limited to the design of temporal logic formulas, which beneficial by removing ambiguity in definitions. Thus model-checking provides an adequate and rigorous conceptual framework for thinking about the dynamical properties of STGs. For example, CTL allows to represent the stability concept with invariance patterns.

Translating a dynamical property (i.e. a description of the system behaviour) written in English into a CTL formula can turn out to be a delicate exercise for non-expert users [DAC99]. One possible way to simplify this task is to provide users with a catalogue mapping query patterns to their translations in CTL, see Table 2.1 [Mon+08; Lar+12; Tho+22]. In order to create even more complex queries, these patterns can be nested by replacing x or y with any other pattern.

2.3.2 Symbolic CTL

The CTL syntax and semantics presented above are called the *explicit perspective* of CTL because they deal with states and paths taken individually, i.e. explicitly. In this section, we present the *symbolic perspective* of CTL that conversely deals with sets of states. While the explicit perspective provides a more accessible intuition to the meaning of CTL formulas, the symbolic perspective is computationally more efficient.

Until now, we considered CTL formulas as boolean functions over states:

$$\begin{array}{rccl} \varphi : & \mathcal{S} & \longrightarrow & \mathbb{B} \\ & s & \longmapsto & \begin{cases} \top & \text{if } s \models \varphi \\ \bot & \text{if } s \not\models \varphi \end{cases} \end{array}$$

English description of the pattern	CTL formula φ
Reachability pattern	
An x state *can* be reached	$\exists \mathsf{F}(x)$
An x state *cannot* be reached	$\neg\exists \mathsf{F}(x)$
Consequence pattern	
If an x state is reached, then it is *possibly* followed by an y state	$\forall \mathsf{G}(x \Rightarrow \exists \mathsf{F}(y))$
If an x state is reached, then it is *necessarily* followed by an y state	$\forall \mathsf{G}(x \Rightarrow \forall \mathsf{F}(y))$
Sequence pattern	
An y state is reachable and is *possibly* preceded *at some time* by an x state	$\exists \mathsf{F}(x \wedge \exists \mathsf{F}(y))$
An y state is reachable and is *possibly* preceded *all the time* by an x state	$\exists (x \mathsf{U} y)$
An y state is reachable and is *necessarily* preceded *at some time* by an x state	$\exists \mathsf{F}(y) \wedge \neg\exists (\neg x \mathsf{U} y)$
An y state is reachable and is *necessarily* preceded *all the time* by an x state	$\exists \mathsf{F}(y) \wedge \forall \mathsf{G}(\neg x \Rightarrow \forall \mathsf{G}(\neg y))$
Invariance pattern	
x states *can* persist forever	$\exists \mathsf{G}(x)$
x states *must* persist forever	$\forall \mathsf{G}(x)$
x states *possibly* remain forever reachable	$\exists \mathsf{G}(\exists \mathsf{F}(x))$
x states *necessarily* remain forever reachable	$\forall \mathsf{G}(\exists \mathsf{F}(x))$
x states are *necessarily* reached infinitely often	$\forall \mathsf{G}(\forall \mathsf{F}(x))$
Reachability & Invariance pattern	
It is *possible* to reach a state from which x states *can* persist forever	$\exists \mathsf{F}(\exists \mathsf{G}(x))$
It is *possible* to reach a state from which x states *must* persist forever	$\exists \mathsf{F}(\forall \mathsf{G}(x))$

Table 2.1: Catalogue mapping query patterns to their translations in CTL. x and y are place-holders for state properties. Adapted in [Tho+22] from [Mon+08].

This perspective was *explicit*, meaning that the semantics of φ was defined element by element of $\mathcal{S}$. But the semantics of φ can also be defined under a *symbolic* perspective as a subset of $\mathcal{S}$:

$$\varphi = \{s \in \mathcal{S} \mid s \models \varphi\} \subseteq \mathcal{S}$$

Under the symbolic perspective, state properties $p \in \mathcal{P}_{\mathcal{S}}$ and logical operators are defined as:

- $\top = \mathcal{S}$
- $p = \{s \in \mathcal{S} \mid p \in \mathcal{V}_{\mathcal{S}}(s)\}$
- $\wedge = \cap$ is the set intersection
- $\vee = \cup$ is the set union
- $\neg : Z \subseteq \mathcal{S} \mapsto \mathcal{S} \setminus Z$ is the set complement

From now on, we only consider CTL state formulas, i.e. we only consider temporal operators resulting from the combination of quantifiers ($\exists, \forall$) and modalities (X, F, G, U), and overlook isolated modalities related to CTL path formulas. Under the symbolic perspective, the temporal operators of CTL ($\exists\mathsf{X}, \forall\mathsf{X}, \exists\mathsf{F}, \forall\mathsf{F}, \exists\mathsf{U}, \forall\mathsf{U}, \ldots$) are functions over set of states: $\tau : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$. The temporal operators of CTL can be defined as iterative com-

binations of the predecessor function $Pred : Z \in \mathcal{P}(\mathcal{S}) \mapsto \{s \in \mathcal{S} \mid \exists s' \in Z \quad s \to s'\} \in \mathcal{P}(\mathcal{S})$, logical operators and state properties.

Proposition 2.1. $\exists \mathsf{X}\varphi = Pred(\varphi)$, we thus use $\exists \mathsf{X}$ for $Pred$ in the following.

Proof. The states that have a successor in φ, i.e. $\exists \mathsf{X}\varphi$, are the predecessors of φ, i.e. $Pred(\varphi)$. □

If we combine Proposition 2.1 with Theorem 2.1, we have a symbolic definition of both $\exists \mathsf{X}$ and $\forall \mathsf{X}$.

Example 2.7. The CTL formula $\exists \mathsf{F} p$ (the set of states that start a path leading to p) can be computed iteratively from p using the set union and the predecessor function. Starting from p, we can add the predecessors of p using the set union. Then we can add the predecessors of this set using the set union (i.e. the predecessors of p from a path of length two). We can keep adding predecessors which are more and more remote from p, until we add the most remote predecessors (as $\mathcal{S}$ is finite, predecessors cannot be infinitely remote). At this point, adding predecessors does not enlarge the set, i.e. a fixed point is reached, which is our stopping criteria. This procedure computes $\exists \mathsf{F} p$ in a finite number of iterative steps (that is the distance of the most remote predecessor of p).

Example 2.8. We give a more formal description of the iterative procedure computing $\exists \mathsf{F} p$ using $\tau : Z \in \mathcal{P}(\mathcal{S}) \mapsto (p \vee \exists \mathsf{X} Z) \in \mathcal{P}(\mathcal{S})$:

- the set $\tau(\emptyset) = p \vee \exists \mathsf{X}(\emptyset) = p$ is the set of states that are in p
- the set $\tau^2(\emptyset) = p \vee \exists \mathsf{X}(p)$ is the set of states that are in p or which have a successor in p
- the set $\tau^3(\emptyset) = p \vee \exists \mathsf{X}(p \vee \exists \mathsf{X}(p))$ is the set of states that are in p, or which have a successor in p, or which have a successor that have a successor in p
- the set $\tau^k(\emptyset)$ is the set of states which have a path of length at most $k-1$ leading to p

The sequence $(\tau^i(\emptyset))_{i \in \mathbb{N}}$ is increasing (for the set inclusion $\subseteq$). Because $\mathcal{S}$ is finite it reaches a maximum, i.e. a fixed point is reached, in a finite number $n \leq |\mathcal{S}|$ of steps (or more precisely $n \leq$ the diameter of $\mathcal{G}$, i.e. the length of the longest path visiting distinct states). $\tau^n(\emptyset)$ is the set of states that are the start of a path leading to p. Thus we have an iterative and set-based way of computing $\exists \mathsf{F} p$.

Every remaining CTL operator can be decomposed into two sub-properties: one that involves only the current state, and another that involves the successor states ($\exists \mathsf{X}$ or $\forall \mathsf{X}$) recursively:

$$
\begin{array}{lll}
\exists \mathsf{F}\varphi & = \varphi \vee \exists \mathsf{X}(\exists \mathsf{F}\varphi) & \forall \mathsf{F}\varphi = \varphi \vee \forall \mathsf{X}(\forall \mathsf{F}\varphi) \\
\exists \mathsf{G}\varphi & = \varphi \wedge (\exists \mathsf{X}(\exists \mathsf{G}\varphi) \vee \neg \exists \mathsf{X} \top) & \forall \mathsf{G}\varphi = \varphi \wedge (\forall \mathsf{X}(\forall \mathsf{G}\varphi) \vee \neg \exists \mathsf{X} \top) \\
\exists(\varphi_1 \mathsf{U} \varphi_2) & = \varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X}(\exists(\varphi_1 \mathsf{U} \varphi_2))) & \forall(\varphi_1 \mathsf{U} \varphi_2) = \varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X}(\forall(\varphi_1 \mathsf{U} \varphi_2)))
\end{array}
$$

Then, the recursive part of this decomposition can be addressed using a fixed point, i.e. the recursion is repeated until a fixed point is reached.

Definition 2.10. Let $\tau : \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ a function, a *fixed point* of τ is any $Z \in \mathcal{P}(\mathcal{S})$ such that $\tau(Z) = Z$.

Theorem 2.2 ([McM93])**.** If $\mathcal{S}$ is finite, and if τ is monotonic ($S \subseteq S' \Rightarrow \tau(S) \subseteq \tau(S')$), then τ has a *least fixed point* noted $\mu\, Z.\tau(Z)$ (the smallest for the set inclusion) and a *greatest fixed point* noted $\nu\, Z.\tau(Z)$ (the largest for the set inclusion).

Moreover $\exists n \in \mathbb{N}$ such that $\mu\, Z.\tau(Z) = \tau^n(\emptyset)$ and $\nu\, Z.\tau(Z) = \tau^n(\mathcal{S})$.

Starting from $\emptyset$, $\tau(\emptyset), \tau^2(\emptyset), \tau^3(\emptyset), \ldots$ forms an increasing sequence of sets, thus $\mu\, Z.\tau(Z)$ can be built by adding states iteratively. Intuitively, "μ means *finite looping*" [BS07b] because each state in $\mu\, Z.\tau(Z)$ has been added at some step, depending on a finite chain of iterations. Conversely, starting from the whole $\mathcal{S}$, $\tau(\mathcal{S}), \tau^2(\mathcal{S}), \tau^3(\mathcal{S}), \ldots$ forms a decreasing sequence of sets, thus $\nu\, Z.\tau(Z)$ can be built by removing states iteratively. Intuitively, "ν means *infinite looping*" because each state not in $\nu\, Z.\tau(Z)$ has been removed at some step, depending on a finite chain of iterations. Thus, for the states remaining in $\nu\, Z.\tau(Z)$ we can iterate the removal condition indefinitely without removing them.

This intuition matches the definition of CTL temporal operators as fixed points, see Table 2.2 and proofs in Annex A.1. For example, $\exists \mathsf{F}\varphi$ is defined as $\mu\, Z.\varphi \vee \exists \mathsf{X} Z$, corresponding to the states that have a path along which we can finitely travel before reaching φ. Similarly, $\exists \mathsf{G}\varphi$ is defined as $\nu\, Z.\varphi \wedge (\exists \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)$, corresponding to the states that have a path along which we can infinitely travel while satisfying φ.

	$\exists$	$\forall$
X	$\exists \mathsf{X}\varphi = Pred(\varphi)$	$\forall \mathsf{X}\varphi = \exists X\top \wedge \neg\exists \mathsf{X}\neg\varphi$
F	$\exists \mathsf{F}\varphi = \mu\, Z.\varphi \vee \exists \mathsf{X} Z$	$\forall \mathsf{F}\varphi = \mu\, Z.\varphi \vee \forall \mathsf{X} Z$
G	$\exists \mathsf{G}\varphi = \nu\, Z.\varphi \wedge (\exists \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)$	$\forall \mathsf{G}\varphi = \nu\, Z.\varphi \wedge (\forall \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)$
U	$\exists(\varphi_1 \mathsf{U} \varphi_2) = \mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X} Z)$	$\forall(\varphi_1 \mathsf{U} \varphi_2) = \mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X} Z)$

Table 2.2: Fixed point definitions of CTL operators.

3 ecco: a modelling and analysis toolbox

Summary

This chapter presents ecco, a Python toolbox for the design and the analysis of formal models of ecosystems that is used in the case studies throughout the thesis. Models in ecco result in STGs, whose states are based upon Boolean variables, and whose transitions are derived from if-then rules. The model analysis in ecco is performed by interactively refining a partition of the state space, using either topological or CTL state properties. Thus, a component graph is incrementally built during the analysis, highlighting some particular behaviours of the system defined by the chosen state properties.

ecco [PTG22a] is a Python [Pyt] library aimed at providing tools for the formal modelling and analysis of ecosystems. ecco consists of both: (1) a modelling language called Reaction Rules (RR) computing into STGs; and (2) an interactive analysis toolbox based on component graphs. In this thesis, we use ecco in the case studies of Chapter 4 and Chapter 6 to compute STGs and to analyse them. ecco has been developed and used for years to model and analyse varied ecosystems [GP19; Di +20; Gau+21; Mao+21; Cos+22; PTG22b]. ecco is available as a free software released under the GNU LGPL, hosted at `http://github.com/fpom/ecco`, and is intended to be used within Jupyter notebooks [Per18].

Each state of an RR model consists of a vector of Boolean variables representing the *functional* presence or absence of the components of the system. The transitions are generated from the execution of if-then rules (*if* the condition is fulfilled, *then* the consequence may arise). A similar a methodology had been previously proposed [Ryk89; Sta90] to model expert knowledge about ecosystem dynamics. Starting from a set of initial states, the full STG is computed by the cascading applications of rules. This modelling approach is exemplified in Figure 3.1.

Instead of analysing the STG explicitly (by considering every state and transition individually), ecco proposes to incrementally build a component graph representing critical features of the dynamics in a human-readable way. Starting from the simplest component graph (composed of a single component: $\mathcal{C} = \{\mathcal{S}\}$), the modeller incrementally and interactively refines the partition by splitting components with regard to user-chosen properties. Throughout the various component splits, the modeller deepens their understanding of the system dynamics, enabling them to formalise new questions to split the component graph further. The global question resulting in the final component graph often could not have been raised without these incremental steps, because the modeller does not know in advance how the system behaves. Moreover, this interactive and incremental workflow provides a user-friendly interface to ecologists unfamiliar with formal analysis. Indeed, whereas the explicit STG can often be overwhelming for the modeller because of its size

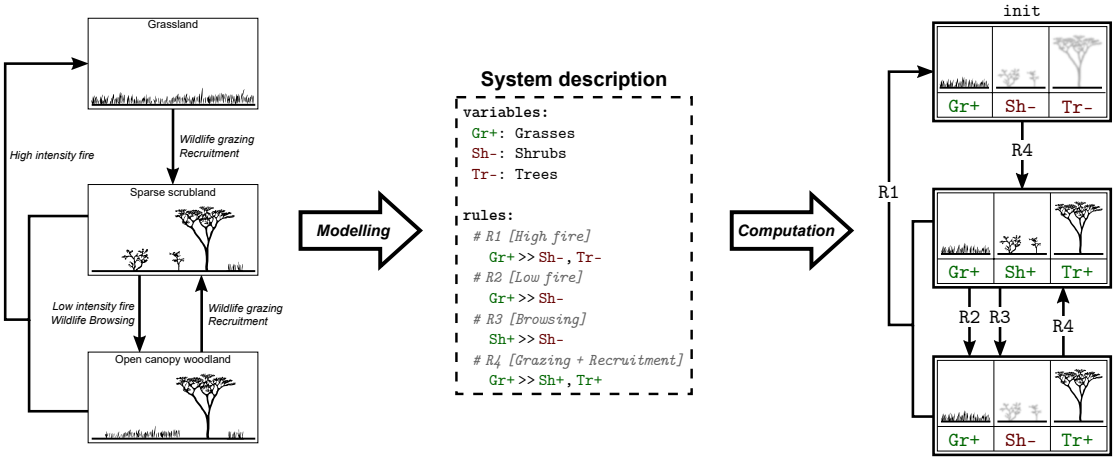


Figure 3.1: Toy example of RR modelling. Modelling of the STM of Figure 1.4a [LC18]. (left) into an RR model (middle: system description) from which an STG can be computed (right).

and the complexity of its dynamics, this workflow starts with the simplest representation of the STG and incrementally complexifies it at each splitting step. Thus the complexity of the component graph (the refinement of the partition) can be finely tuned by the modeller. The global question resulting from these incremental steps may be much more complex than what the modeller would be able to formalise a priori.

3.1 A modelling language : Reaction Rules (RR)

The RR modelling language used in ecco involves *variables* and *rules* [GP19; PTG22b]. Variables are the biotic, abiotic and anthropic entities of the ecosystem, modelled as Boolean variables (on/off, noted as +/−). Rules define how variable values may evolve by applying an effect (assignment of variable values) depending on a guard (condition on variable values).

3.1.1 Syntax

This section describes how to write an RR model. Figure 3.2 shows a toy version of the *Borana model* that will be presented in Chapter 4. Variables are declared first by giving each of them a name, an initial state ("+" for on, "−" for off, or "∗" to allow either initial values), and a textual description. For instance, variable `Gr` is declared initially on and models the presence of grasses in the ecosystem. A variable is considered functionally present if its presence has an observable influence on the system, and functionally absent otherwise. Variables influencing the system without being influenced in turn are called *controls*, for example climatic conditions or management policies. Controls remain constant along the dynamics, thus two states with distinct control values are out of reach from one another. The initial state of a control is often undefined, for example `Fb*`, in order to model distinct scenarios: both `Fb+` and `Fb-` are considered as initial states of the system. Variable declarations are organised into arbitrarily chosen categories (except for "`rules`", which is a

reserved keyword). For instance, variable `Gr` is declared within category `variables` while variable `Fb` is declared within category `controls`. These categories are for information purposes only and have no consequence on the semantics.

```
variables:
  Gr+: Grasses
  Sh-: Shrubs
  Tr-: Trees
controls:
  Fb*: Fire ban
rules:
  [high fire]                Fb-, Gr+ >> Sh-, Tr-      # R1
  [low fire]                 Fb-, Gr+ >> Sh-           # R2
  [browsing]                 Sh+ >> Sh-                # R3
  [grazing + recruitment]    Gr+ >> Sh+, Tr+           # R4
```

Figure 3.2: RR toy model of the Borana vegetation pathways. Rules are named for reference using a comment at the end of lines.

Rules are listed after variable declarations and consist of two sides separated by ">>": the left-hand side is the *guard*, that is, the condition for the execution of the rule; the right-hand side is the *effect*, that is, the variables assignment that takes place upon execution of the rule.

For instance, rule `R1` specifies that if fire is not banned (`Fb-`) and grasses are present (`Gr+`) then high fire may occur, resulting in the disappearance of both shrubs and trees (`Sh-` and `Tr-`). Rules may be prefixed by arbitrary tags enclosed into square brackets, like "`[high fire]`" in rule `R1`, that can be referred to during analysis. Comments, like "`# R1`", play no role in the semantics.

Limitation

RR could easily be extended with multivalued variables, representing different thresholds having distinct effects. The possible values would be defined during variable declaration and used in the guards and effects of the rules. For the time being, `ecco` is limited to Boolean variables in order to keep models as abstract as possible, and because in practice data is often lacking about thresholds' existence or ordering.

3.1.2 Semantics

This section describes how an STG is computed from an RR model. The execution of an RR system is defined in terms of operational rules involving *states*, i.e. valuation of its variables, and *transitions*, that are executions (called *firings*) of the rules allowing to build new states from existing ones:

- The *initial states* are defined from the declaration of the variables, either they are initially on/off, or both values are considered (like `Fb*` in Figure 3.2).

- A rule is *enabled* when its guard is satisfied by the state and its effect is not already realised.
- If no rule is enabled at a state, then this state is a *dead-end*.
- Firing a rule R enabled at a state s is made by applying the effect of R onto s, yielding a new state $s' \neq s$, which is a transition noted by $s \xrightarrow{\mathtt{R}} s'$.
- The states obtained by firing rules from a given state are called its *successors*.

If we note a state by the set of variables it valuates to on, the model defined in Figure 3.2 has two initial states: $\{\mathtt{Fb}, \mathtt{Gr}\}$ and $\{\mathtt{Gr}\}$. Rule R4 is enabled in both states because we have Gr+ (condition) but not Sh+ nor Tr+ (effect). So it may fire and we have two transitions $\{\mathtt{Fb}, \mathtt{Gr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$ and $\{\mathtt{Gr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$. In the initial states, rule R3 is not enabled because its guard is not satisfied. Rules R1 and R2 are not enabled from the initial states either because in $\{\mathtt{Fb}, \mathtt{Gr}\}$, even if the guard is satisfied, the effect is already realised, and in $\{\mathtt{Gr}\}$ the guard is not satisfied.

An STG is generated from an RR model by repeatedly applying the firing rules from the initial states and all newly obtained successor states. The STG obtained from our toy model is depicted in Figure 3.3 and is computed as follows:

- We start from the initial states $\{\mathtt{Fb}, \mathtt{Gr}\}$ and $\{\mathtt{Gr}\}$ (drawn at the top).
- The possible transitions are:
  - $\{\mathtt{Fb}, \mathtt{Gr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$
  - $\{\mathtt{Gr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$

  which yields two new states (drawn in the middle).
- Then, the possible transitions are:
  - $\{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\} \xrightarrow{\mathtt{R3}} \{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Tr}\}$
  - $\{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\} \xrightarrow{\mathtt{R1}} \{\mathtt{Gr}\}$
  - $\{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\} \xrightarrow{\mathtt{R2}} \{\mathtt{Gr}, \mathtt{Tr}\}$
  - $\{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\} \xrightarrow{\mathtt{R3}} \{\mathtt{Gr}, \mathtt{Tr}\}$

  which yields only two new states (drawn at the bottom), one of which being obtained twice, and the state $\{\mathtt{Gr}\}$ already existing.
- Then, the possible transitions are:
  - $\{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Tr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Fb}, \mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$
  - $\{\mathtt{Gr}, \mathtt{Tr}\} \xrightarrow{\mathtt{R1}} \{\mathtt{Gr}\}$
  - $\{\mathtt{Gr}, \mathtt{Tr}\} \xrightarrow{\mathtt{R4}} \{\mathtt{Gr}, \mathtt{Sh}, \mathtt{Tr}\}$

  which does not yield any new state, so the computation stops.

The STG resulting from this computation exhibits two disjoint subgraphs corresponding to the two initial values of control Fb.

Technical remark

In this thesis, we give an operational semantics of RR models, i.e. describing how the computation of an RR model takes place step by step. The semantics of RR models can also be given in terms of Petri Nets, see [PTG22b].

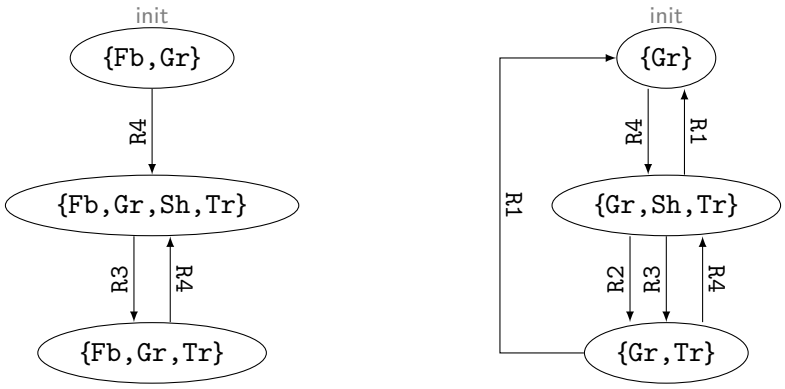


Figure 3.3: STG computed from the RR toy model of Figure 3.2.

Moreover, instead of a list of variables and rules (Figure 3.2), an RR model can be represented as a hypergraph (each variable is a node and each rule is a hyperedge) where the relationships between the variables are made visual, see [PTG22a].

3.2 An interactive analysis interface based on component graphs

The STG computed from an RR model is represented in ecco as a component graph. Starting from the simplest component graph consisting of a single node (i.e. the partition is restricted to a single component $\mathcal{S}$ enclosing all the states), the user incrementally and interactively refines the partition by splitting or merging components. In doing so, a large STG may be represented efficiently by a small component graph that can be seen as a hybrid object mixing symbolic components with explicit information about their relationship. The size of the component graph can be finely tuned as every incremental splitting or merging step only add or remove a limited number of components. As the user incrementally refines the partition step by step, they are able to fully understand the component graph even when the partition ends up being quite complex.

3.2.1 Splitting with respect to state properties

State properties, such as the value of the variables, can be taken as a splitting criterion. For example, a component can be split between the states where a chosen variable is on and the states where this variable is off. More generally, a component can be split with respect to any Boolean formula over the variables (built with $\neg, \wedge, \vee$), separating the states whose variables satisfy the formula from these that do not.

Similarly, the set of *initial states* $\mathcal{S}_0$ may be split apart, as being an initial state is a state property. Indeed, initial states were chosen by the modeller because they are an interesting starting point for the investigation of the system's behaviour.

Finally, as the semantics of CTL formulas is defined upon states, satisfying a CTL formula

φ is a state property. It can be taken as a splitting criterion, just like in the symbolic perspective of CTL:

$$\varphi = \{s \in \mathcal{S} \mid s \models \varphi\} \subseteq \mathcal{S}$$

Thus by intersecting a component with φ, we can split it between the states that satisfy the formula and the states that do not. Any other temporal logic whose semantics is defined upon states could be used as a splitting criterion in the same way (such as the extensions of CTL that will be presented in Chapter 5).

3.2.2 Splitting with respect to topology

Considering an STG, we can partition its set of states $\mathcal{S}$ into its topological components:

- A *strongly connected component* (SCC) is a maximal set $SCC \subseteq \mathcal{S}$ of states such that $\forall s \neq s' \in SCC : s \rightarrow^+ s'$. Intuitively, an SCC represents a set of states within which the system may stay in the long term, which corresponds to a stability (or a multistability, as systems biology calls it) of the system.
- The *convex hull of the SCCs* is the smallest set $H \subseteq \mathcal{S}$ that contains all the SCCs and such that $\forall s \neq s' \in H$ if $\exists s''$ such that $s \rightarrow^+ s''$ and $s'' \rightarrow^+ s'$ then $s'' \in H$. Intuitively H is the union of the SCCs plus the states between them. Our experience shows that, at a first attempt, the SCC hull H makes a better component choice than the individual SCCs themselves, which are often too numerous to build a human-readable component graph.
- The *dead-ends* are the states with no successors $\{s \in \mathcal{S} \mid s \not\rightarrow\}$.

Technical remark

Note that in contrast with the usual definition, we exclude trivial SCCs consisting of only one state, because they do not represent a long-term behaviour (contrarily to multi-state SCC within which the system can remain trapped indefinitely long). Moreover, splitting with respect to trivial SCCs could lead to a large component graph with many singleton components, which conflicts with our readability goals.

A topological decomposition splits apart some topological components chosen among SCCs, SCCs hull, and dead-ends. Afterwards, the remaining states can be split into the *basins* leading to these chosen components. Given a set $\{C_1, \ldots, C_k\}$ of components, two states $s, s' \notin \{C_1, \ldots, C_k\}$ are in the same basin iff they allow to reach exactly the same C_is [Bér+13]. For a cleaner presentation, the dead-end component may be merged with the basin leading solely to it. Such a decomposition can be applied to the full STG, or to any arbitrary component.

3.2.3 Implementation

ecco can be run or installed as a Docker image [Boe15] to ease its use. The distribution includes a script that can be the single installed element and that takes care of starting

Docker with the appropriate options. For instance, with this script and Docker installed, starting `ecco` just requires running "`ecco -mount=.`" from the command line. This retrieves online the Docker image for the latest version of `ecco`, starts it with access to the files in the current directory, and opens Jupyter in the default web browser. The Docker image also features JupyterHub that is a multi-user server for Jupyter notebooks [Per18]. Thus, `ecco` is readily configured to support multiple users with separated accounts.

The library `ecco` mainly consists of (1) a Cython module that interfaces with `ITS-tools` and `libDDD` [Thi15] to provide a symbolic STG class that is used by (2) a frontend that provides a component graph class to compute and split hybrid component graphs from symbolic STG objects. `ecco` gathers set of states into a compact data structure (based on *Data Decision Diagram*, *DDD* [Cou+02]) from which the set of successor (or predecessor) states can be efficiently computed [Bry18]. The CTL model-checker integrated into `ecco` is symbolic as well and based on the fixed points of Table 2.2. It computes as a DDD the set of states of the STG satisfying a query formula, and a *yes/no* answer can be obtained by intersecting this set with the set of initial states. As sets of states are symbolically stored, the computation remains efficient even for huge models, and thus the analysis can remain interactive.

A typical `ecco` session is organised as follows:

1. An RR model is loaded.
2. An STG object is built by translating the RR source file into a GAL source file [Thi15, Sec. 5]. The GAL is then loaded by `ITS-tools` to provide the symbolic transition relations of the STG object. From a set of initial states, the set of reachable states $\mathcal{S}$ is computed using the symbolic transition relations.
3. The user builds component graphs from the STG by splitting or merging components.

Examples of `ecco` sessions are given in the next chapter.

Each component graph is an immutable object, and the components are numbered consistently across all component graphs. A component graph object also comes with tabular representations of its nodes and edges stored as Pandas dataframes [McK11]. The nodes table has the following columns displaying information about each component:

- node: the component number.
- size: the number of states in the component.
- on: the variables that are on for every state within the component.
- off: the variables that are off for every state within the component.
- topo: the topological properties of the component, has_init means that the component has initial states, is_init that it has all the initial states, etc.
- hasno, has, contains, isin, equals: the relationships between the component and the temporal logic formulas that have been tested against it, as illustrated in Figure 3.4.

The edges table is simpler and straightforward. Both tables may be augmented by the user with arbitrary columns. The columns' content may be used to tune the graphical presentation of the component graph, e.g. the nodes' colours or labels.

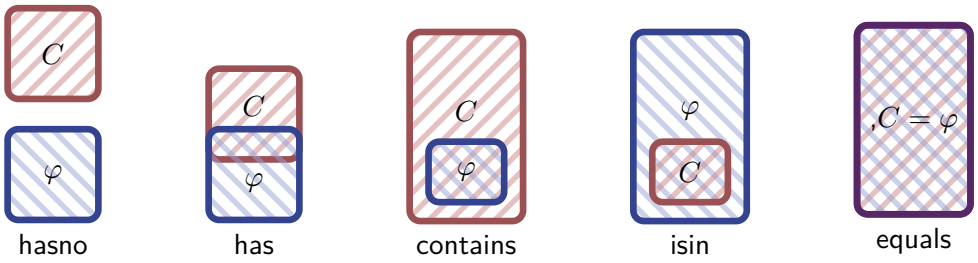


Figure 3.4: The possible relationships between a component C and the set of states that satisfy a property φ [PTG22a].

A critical aspect of the nodes table is that its "relationship" columns (hasno to equals) are updated each time a formula is checked against some components. This information is then inherited by any new component graph obtained through further splits so that the nodes table keeps track of how a component graph has been obtained through successive splits. Altogether, the shape of a component graph and its nodes and edges tables exhibit the understanding built incrementally by the user.

4 Case studies

Summary

This chapter presents two case studies exemplifying the concepts and tools presented in the previous chapters. First, we model the vegetation changes of the Borana Zone in Ethiopia, taking into account diverse management scenarios. Using CTL formulas, we build a component graph representing bush encroachment, and find management policies preventing it or making it reversible. Second, we model protists community assembly based on the results of laboratory experiments. We build a component graph partitioning the states with respect to the stable communities they lead to using topological properties. Then we compare two versions of this component graph, fixing a specific species as initially present or absent, to investigate the impact of this species on the behaviour of the system.

The `ecco` notebooks encompassing the case studies presented in this chapter can be found in [Tho22]. Both the RR model file and the python analysis notebook are given for each case study, along with a static html preview of the notebook. Computing all the results took only a few seconds on a modern laptop (Linux 5.4 Mint/Ubuntu, 32G RAM, CPU Intel Core i7–7820HQ 2.9GHz).

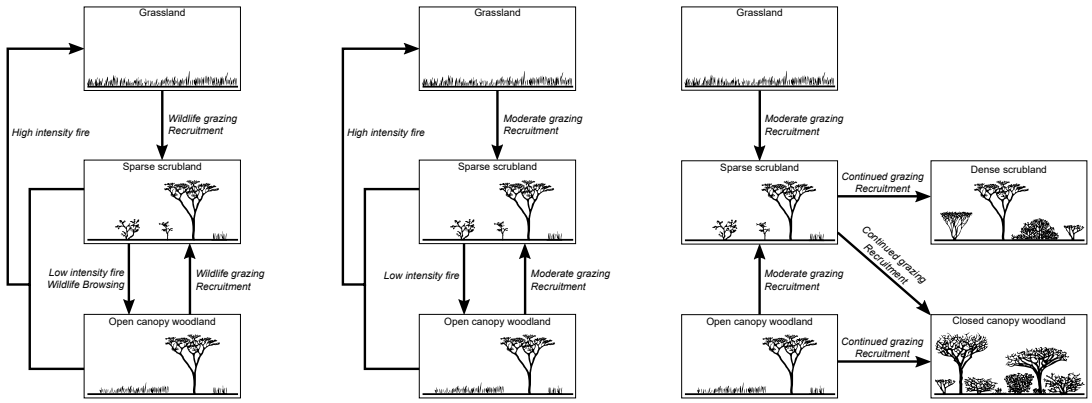
4.1 Borana vegetation model

The vegetation of Borana Zone in southern Ethiopia, see Figure 4.1, is undergoing bush encroachment, that is the proliferation of woody plants resulting in impenetrable thickets where grasses are absent. The lack of grasses threatens local people's livelihood, which predominantly relies upon cattle herding. Bush encroachment results from a change in ecosystem management policies, such as the fire ban by the government from the 1970s towards the 2000s, or the increase in grazing intensity resulting from sedentarization. Borana Zone has historically and spatially experienced various management policies, for example the crop cultivation ban until the 1950s' or the grazing ban in some forest areas for conservation purposes, resulting in different vegetation pathways. Changes in vegetation pathways cannot always be reversed, for example traditional pastoralism using fire can prevent bush encroachment but cannot reverse it, because both fire and grazing need a minimal herbaceous cover. Local people and policymakers are interested in finding management policies (historical ones or new recommendations) achieving particular goals, for example preventing or reversing bush encroachment while maintaining livelihood.

Liao et al. described the vegetation pathways of the Borana Zone in terms of states and

Figure 4.1: Pictures of Borana Zone. Taken from Workshop in Borana, Ethiopia, 2012 (flickr) authored by *Anton Eitzinger, Climate Change, Agriculture and Food Security* under (CC BY-NC-SA 2.0).

transitions. First, they drew historical vegetation pathways as State-and-Transition Models, see Figure 4.2, using plant survey and cattle tracking [LC18]. Second, they defined states as vegetation classes, and recorded the transitions among vegetation classes between 2003 and 2013 using satellite imagery [LCD18]. Finally, they discussed existing and recommended management policies to mitigate bush encroachment [Lia+20].



(a) Before livestock introduction

(b) With livestock and fire

(c) With livestock and fire ban

Figure 4.2: State-and-transition models of the Borana vegetation pathways. Bush encroachment is prevented in the "before livestock introduction" scenario and in the "with livestock and fire" scenario, but always occurs in the "with livestock and fire ban" scenario. Taken from [LC18, Fig.5].

4.1.1 Modelling

> *"A big opportunity lies in applying the STM framework to forecasting future ecosystems under novel conditions. [...] Exploring the future in an STM framework can build a constructive ecology where novel ecosystems can be imagined, but in a disciplined way, restricted by well founded knowledge about processes and transitions." [WW20]*

Based on the literature [Lia16; LCD18; LC18; Lia+20], we built an RR description of the Borana vegetation pathways [Tho+22], called "*Borana model*" in the following, from which an STG can be computed using `ecco` [PTG22a]. While most STGs found in ecology are directly drawn from observations, we want to show that a complex STG can be computed from a compact mathematical system description. This enables the construction of models not only based on past observations but also forecasting novel behaviours, which is a current challenge in the STM framework [WW20]. While each STM of Figure 4.2 represents an observed scenario, the *Borana model* embraces the same historical scenarios as well as other recommended management scenarios to foresee their cascading effects.

The complete *Borana model* consists of 15 variables, including seven controls, see Figure 4.3), and 19 rules, see Figure 4.4. The variables represent plants, animals or management scenarios, while the rules are built from the description of recorded transitions available in the literature. For example, rule `R2` embodies the following description: "*High intensity fire could change the landscape into a grass-dominated system.*" [LC18] Justifications of the modelling choices assumed by the *Borana model* are given in Annex A.2. Each valuation of the variables describes a state of the Borana ecosystem, that can be classified into vegetation classes [LCD18], see Table 4.1. Each valuation of the controls defines a specific *scenario* for the management of the Borana Zone, i.e. a combination of altitude and management policies, inspired by historical management and recommendations to limit encroachment [Lia+20]. The control variables never change in consequence of the rules, see Figure 4.4, hence they influence the system without being influenced by it.

```
variables:
  Gr+: Grasses
  Sh-: Shrubs
  Tr-: Savanna trees
  Sa-: Tree saplings
  Cr-: Crops
  Lv-: Livestock (cattle)
  Gz-: Wild grazers
  Bw-: Wild browsers
```

```
controls:
  Alt*: Altitude
  Fb*: Fire ban
  Cb*: Crop ban
  Wl*: Wildlife
  Ps*: Pastoralism
  Ig*: Intensive grazing
  BLv*: Browsing livestock
```

Figure 4.3: Variables of the *Borana model*.

Vegetation Class	State property
`Closed Canopy Woodland`	`Gr-`, `Sh-`, `Tr+`, `Sa*`, `Cr-`
`Dense Scrubland`	`Gr-`, `Sh+`, `Tr+`, `Sa*`, `Cr-`
`Bushland`	`Gr-`, `Sh+`, `Tr-`, `Sa*`, `Cr-`
`Open Canopy Woodland`	`Gr+`, `Sh-`, `Tr+`, `Sa*`, `Cr-`
`Sparse Scrubland`	`Gr+`, `Sh+`, `Tr*`, `Sa*`, `Cr-`
`Cultivated Land`	`Gr-`, `Sh-`, `Tr*`, `Sa-`, `Cr+`
`Grassland`	`Gr+`, `Sh-`, `Tr-`, `Sa-`, `Cr-`
`Sparsely Vegetated Land`	`Gr-`, `Sh-`, `Tr-`, `Sa-`, `Cr-`

Table 4.1: Borana vegetation classes [LCD18] as state properties. These vegetation classes form a partition of the states. Note that grasses are considered functionally present (`Gr+`) in the `Sparse Scrubland` class although covering only between 10% and 30% of the surface [LCD18], indeed both fire and grazing occur in `Sparse Scrubland` [Lia+20].

The *Borana model* has $2^7 = 128$ initial states, one for each scenario, i.e. one for each val-

```
rules:
  [Low fire]       Fb-, Gr+ >> Sh-, Sa-, Lv-, Gz-, Bw-                  # R1
  [High fire]      Fb-, Gr+ >> Sh-, Tr-, Sa-, Lv-, Gz-, Bw-             # R2
  [Trees]          Sa+ >> Tr+                                           # R3
  [Grass]          Sh-, Tr-, Sa-, Cr- >> Gr+                            # R4
  [CCW]            Alt+, Fb+, Gr-, Sa+ >> Sh-, Tr+                      # R5
  [Bushland]       Alt-, Sh+, Tr- >> Sa-                                # R6
  [Grazers]        Wl+, Gr+, Lv- >> Gz+                                 # R7
  [Browsers]       Wl+, Sh+, Lv- >> Bw+                                 # R8
  [Browsers]       Wl+, Sa+, Lv- >> Bw+                                 # R9
  [Livestock]      Ps+, Gr+ >> Lv+, Gz-, Bw-                            # R10
  [Livestock]      Ps+, BLv+, Sh+ >> Lv+, Gz-, Bw-                      # R11
  [Livestock]      Ps+, BLv+, Sa+ >> Lv+, Gz-, Bw-                      # R12
  [Grazing]        Gr+, Lv+ >> Sh+, Sa+                                 # R13
  [Grazing]        Gr+, Gz+ >> Sh+, Sa+                                 # R14
  [Intens. graz.]  Ig+, Lv+ >> Gr-, Lv-                                 # R15
  [Browsing]       Bw+ >> Gr+, Sh-, Sa-, Bw-                            # R16
  [Browsing]       BLv+, Lv+ >> Gr+, Sh-, Sa-, Bw-                      # R17
  [Crops]          Alt+, Cb-, Tr+ >> Gr-, Sh-, Sa-, Cr+, Lv-, Gz-, Bw-  # R18
  [Crops]          Cr+ >> Gr+, Cr-                                      # R19
```

Figure 4.4: Ruleset of the *Borana model*.

uation of the 7 control variables. All initial states correspond to the grassland vegetation class [LC18]: only grasses are present, see Figure 4.3 and Table 4.1. A subgraph is generated from each initial state by the cascading applications of the rules. These subgraphs are disconnected (no rule changes the controls) and form together the full STG of 1185 states computed from the *Borana model*. The largest scenario subgraph has 26 states.

The subgraph corresponding to the scenario before livestock introduction at high altitude is given in Figure 4.5 as an example. This subgraph can be partitioned with respect to vegetation classes (Table 4.1), resulting in a component graph that will be compared in the following with the corresponding STM.

Technical remark

For clarity, we depicted the states $s \in \mathcal{S}$ of an STG $\mathcal{G}$ as ellipses, while the components $c \in \mathcal{C}$ of a component graph $\mathcal{G}/\mathcal{C}$ are depicted as rectangles with rounded corners.

The three available STMs [LC18] drawn from observations, see Figure 4.2, were compared to the component graphs computed from the *Borana model* for the corresponding scenarios (each component graph encompasses two subgraphs, one for each altitude level). The first STM describes the vegetation pathways under wildlife herbivory and fire. It is almost identical to its corresponding component graph, see Figure 4.6. Their only difference is that in the component graph, the transition from `Sparse Scrubland` to `Grassland` is additionally labelled by `Low intensity fire` and `Browsing`, which is correct because these events may happen in `Sparse Scrubland` before the establishment of trees.

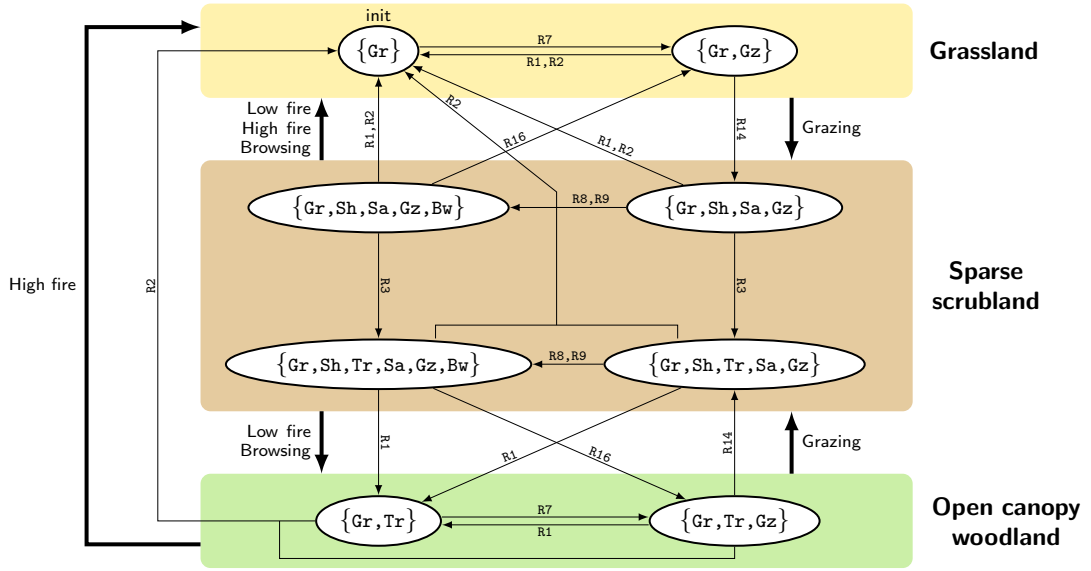
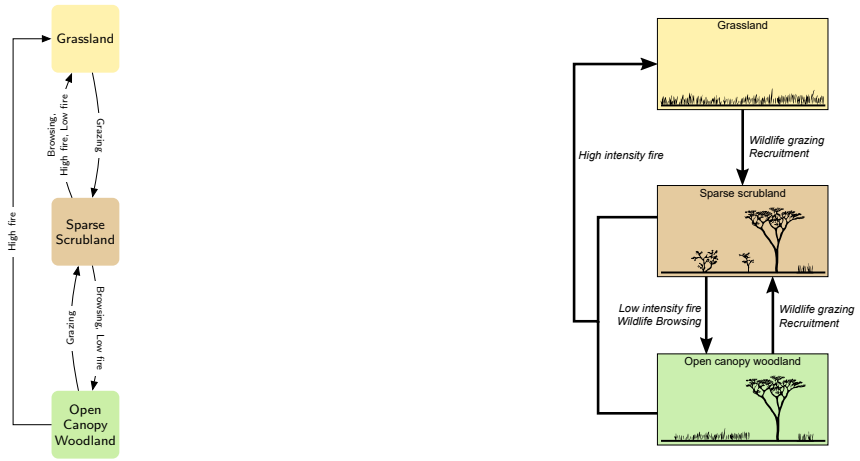


Figure 4.5: Scenario subgraph partitioned between vegetation classes. This scenario corresponds to the STM before livestock introduction, see Figure 4.2a, at low altitude. It was computed using the following initial control values: `Alt+`, `Fb-`, `Cb+`, `Wl+`, `Ps-`, `Ig-`, `BLv-`. The states of the STG are depicted as white ellipses labelled with the variables valuated on. The vegetation classes are depicted as colored rounded rectangles. The vegetation classes form a component graph whose edges are labelled with the tags of the corresponding rules.



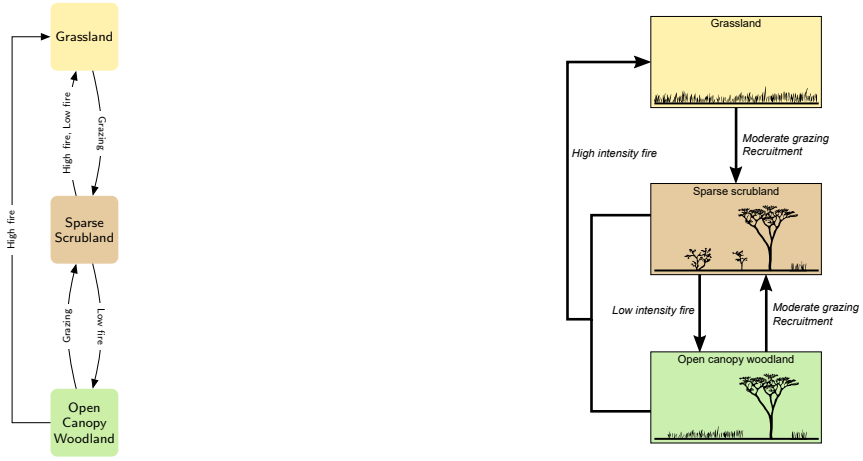
(a) *Borana model* component graph

(b) STM [LC18, Fig.5]

Figure 4.6: Component graph vs. STM for wildlife herbivory and fire. The component graph was computed using the following initial control values: `Alt*`, `Fb-`, `Cb+`, `Wl+`, `Ps-`, `Ig-`, `BLv-`.

The second STM describes the vegetation pathways under extensive grazing and fire. It is also almost identical to its corresponding component graph, except for the additional labels mentioned above, see Figure 4.7.

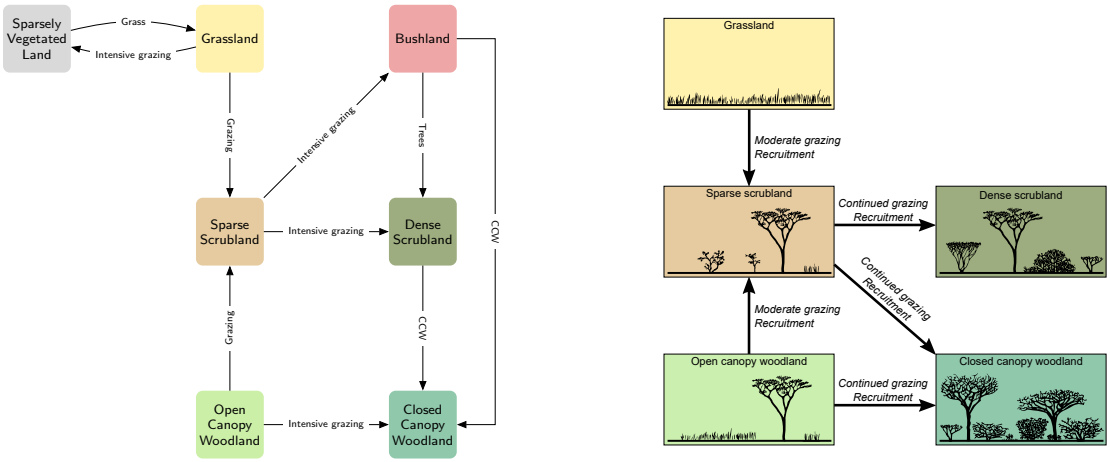
The third STM describes the vegetation pathways under intensive grazing and fire ban. It presents more differences from its corresponding component graph, see Figure 4.8. Yet the additional vegetation classes and most of the additional transitions in the component graph were empirically observed [LCD18], such as the transition from `Dense Scrubland` to `Closed Canopy Woodland`. In particular `Sparse Scrubland` is described as a transi-



(a) *Borana model* component graph **(b)** STM [Lia+20, Fig.3]

Figure 4.7: Component graph vs. STM for extensive grazing and fire. The component graph was computed using the following initial control values: `Alt*`, `Fb-`, `Cb+`, `Wl-`, `Ps+`, `Ig-`, `BLv-`.

tory state between `Grassland` and `Dense Scrubland` [LCD18], and acts as such in the component graph. The additional unobserved transitions in the component graph revolve around the transitory nature of `Sparse Scrubland` and `Dense Scrubland`. This lack of observation may be caused by the ten years of delay between the observations, that may be a too long time period to notice such brief changes. Moreover, the component graph showcases the main features of the STM: encroachment is not reversible, and `Open Canopy Woodland` is not reachable from `Grassland`.



(a) *Borana model* component graph **(b)** STM [LC18, Fig.5]

Figure 4.8: Component graph vs. STM for intensive grazing and fire ban. The component graph was computed using the following initial control values: `Alt*`, `Fb+`, `Cb+`, `Wl-`, `Ps+`, `Ig+`, `BLv-`. Additionally we set `Tr*` initially because both `Grassland` and `Open Canopy Woodland` are initial states in the STM.

4.1.2 Results

In this section, we progressively build a component graph aiming to provide answers to the following questions:

- *Which management policies prevent bush encroachment?*
- *Is bush encroachment reversible without changing management policy?*

Starting from the whole state space $\mathcal{S}$ of the *Borana model*, we incrementally refine a partition answering both questions. The first step is to split $\mathcal{S}$ between the states that are encroached, and these that are not. To do so, we define bush encroachment as the state property $= (\texttt{Sh+} \vee \texttt{Tr+}) \wedge \texttt{Gr-} \wedge \texttt{Cr-}$, which corresponds to the vegetation classes with shrubs or trees but without grass nor crop (`Closed Canopy Woodland`, `Dense Scrubland`, and `Bushland`, see Table 4.1). Using this state property, we can partition $\mathcal{S}$ between the states that satisfy it (component #2) and these that do not (component #1), see Figure 4.9.

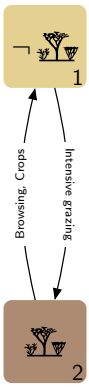


Figure 4.9: First splitting step.

The second step is to split the components depending on whether bush encroachment can happen or not. To do so, we define the CTL formula $\exists \mathsf{F}(\ )$, a reachability pattern stating that an encroached state is reachable. Using this formula, we can partition component #1 of Figure 4.9 between the states that satisfy the formula (component #3) and these that do not (component #4), see Figure 4.10.

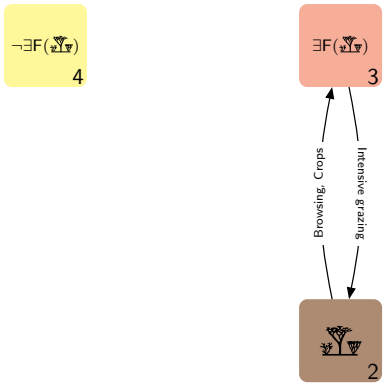


Figure 4.10: Second splitting step.

Technical remark

Each component is numbered in the bottom right corner. When several components of distinct component graphs encompass the exact same set of states, they are all given the same number and the same colour across all component graphs. Thus, if a split does not divide a component, then it keeps the same number and the same colour before and after the split. For example the two components numbered #2 in Figure 4.9 and in Figure 4.10 embody exactly the same set of states.

Some states prevent bush encroachment, that is the states of component #4 labelled $\neg\exists F(\text{🌳})$. Hence why component #4 is disconnected from the rest of the component graph. Bush encroachment is reachable in the other states, but may be reversible as indicated by the transition from component #2 🌳 to component #3 $\exists F(\text{🌳})$. The third step of the partition refinement is to split the components #2 and #3 where bush encroachment can happen depending on whether bush encroachment can be reversed or not. To do so, we define the CTL formula $\exists F\left(\text{🌳} \wedge \exists F(\neg\text{🌳})\right)$, a sequence pattern stating that bush encroachment can be reversed. Using this formula, we can partition components #2 and #3 of Figure 4.10 between the states that satisfy the formula (components #5 and #7) and these that do not (components #6 and #8), see Figure 4.11. Although components #7 and #8 differ by the formulas they satisfy, we labelled them equally by 🌳 to ease the interpretation of the component graph.

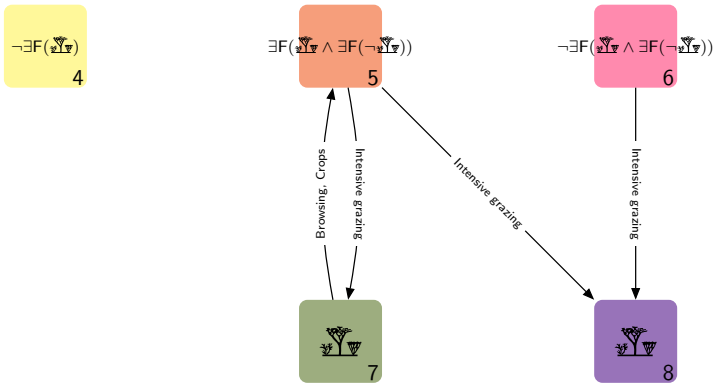


Figure 4.11: Third splitting step.

Some states of the STG can lead to an irreversible bush encroachment, that is the states of the component #6 $\neg\exists F\left(\text{🌳} \wedge \exists F(\neg\text{🌳})\right)$. Bush encroachment is at least sometimes reversible for others, that is the states of the component #5 $\exists F\left(\text{🌳} \wedge \exists F(\neg\text{🌳})\right)$. Note that the encroached states 🌳 are split as well, as some may be reversed (component #7) and others may not (component #8). The fourth and final step is to split component #5, where bush encroachment is sometimes reversible, between the states where bush encroachment is always reversible, and the states where bush encroachment is sometimes irreversible. To do so we define the CTL formula $\forall G(\text{🌳} \Rightarrow \exists F(\neg\text{🌳}))$, an invariance pattern stating that bush encroachment is always reversible. Using this formula, we can partition components #5 and #7 of Figure 4.11 between the states that satisfy the formula (component #9 and #11) and these that do not (components #10 and #12), see Figure 4.12.

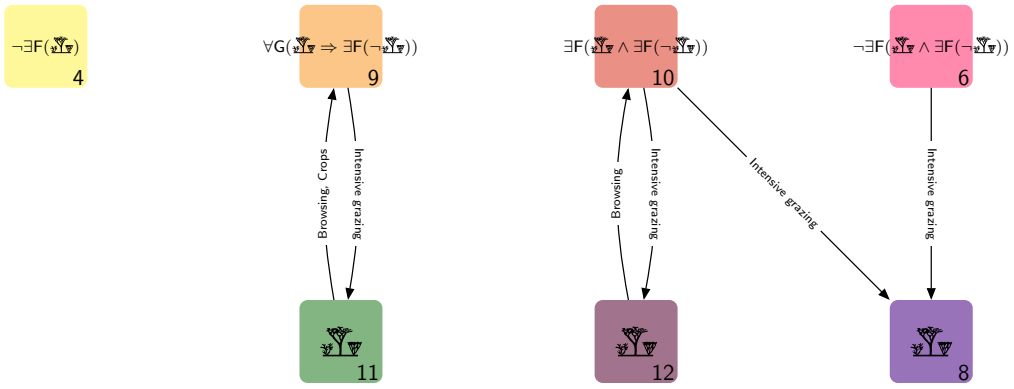


Figure 4.12: Final component graph.

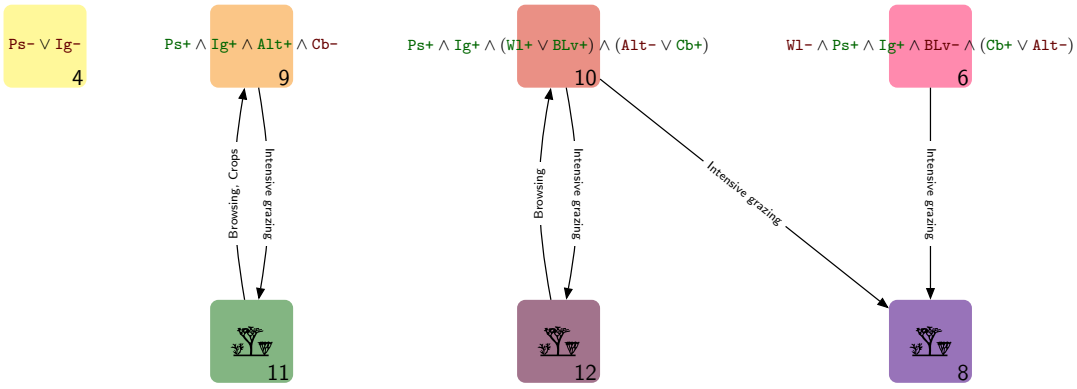


Figure 4.13: Final component graph with control values.

The component graph of Figure 4.12 answers both questions. Indeed, the management policies that prevent bush encroachment are these of component #4 $\neg\exists\mathrm{F}(🌳)$. Bush encroachment is always reversible under the management policies of component #9 $\forall\mathrm{G}(🌳 \Rightarrow \exists\mathrm{F}(\neg🌳))$, sometime reversible under the management policies of component #10 $\exists\mathrm{F}\left(🌳 \wedge \exists\mathrm{F}(\neg🌳)\right)$, and never reversible under the management policies of component #6 $\neg\exists\mathrm{F}\left(🌳 \wedge \exists\mathrm{F}(\neg🌳)\right)$. To get the specific management policies of a component, we extract exclusively the control variables from its decision diagram to get a Boolean formula that we transform into *canonical form* using SymPy [Meu+17]. A version of the component graph whose components are labelled with their control valuations is given in Figure 4.13.

We give an ecological interpretation of this result alongside other results exhibiting how the model-checking methodology could help better understand the Borana vegetation pathways and choose adequate management policies. We designed six CTL queries covering all five pattern types introduced in Table 2.1 and including the formulas used to partition the component graph. These queries are built upon the following state properties:

- Bush encroachment as defined previously:

$$\texttt{Encroachment} = (\texttt{Sh+} \vee \texttt{Tr+}) \wedge \texttt{Gr-} \wedge \texttt{Cr-}$$

- Closed canopy woodland is a vegetation class, see Table 4.1:

$$\texttt{ClosedCanopyWoodland} = \texttt{Gr-} \wedge \texttt{Sh-} \wedge \texttt{Tr+} \wedge \texttt{Cr-}$$

- Subsistence production [Wha69] corresponds to the states with crops or livestock:

$$\texttt{Subsistence} = \texttt{Cr+} \vee \texttt{Lv+}$$

We use model-checking to select the control valuations, i.e. scenarios, satisfying each query. For each query and scenario, the model-checker tests whether the initial state of the scenario exhibits the temporal behaviour specified by the query or not, returning a *yes/no* output. We selected the valuations of the controls for which the associated model-checking output is *yes*. The omitted controls have no impact on the model-checking output. Computing all the model-checking results took only a few seconds on a modern laptop (Linux 5.4 Mint/Ubuntu, 32G RAM, CPU Intel Core i7-7820HQ 2.9GHz).

The first two queries select the scenarios enabling bush encroachment. The answer to the first query shows that intensive grazing is the necessary condition for encroachment, as can be seen on the label of component #4 $\neg\exists\mathsf{F}(\text{🌳})$ in Figure 4.13. This may seem counter-intuitive because fire seems to prevent bush encroachment in Figure 4.2, yet bush encroachment has continued in Borana despite the lift of the fire ban in the 2000s [LCD18; Lia+20]. The answer to the second query shows that at least one of the following controls is additionally needed to reach closed canopy woodland: `Alt+`, `Fb-`, `Wl+`, or `BLv+`. Each of these controls enables one of the rules that removes shrubs without changing grasses nor trees (`R5`, `R1`, `R16`, or `R17` respectively, see Figure 4.4). Thus, when combined with intensive grazing (`R15`), grasses and shrubs are removed without removing trees, resulting in closed canopy woodland.

The third and fourth queries select the scenarios making bush encroachment reversible. The third query selects the scenarios where encroachment is always reversible (from any encroached state, there is a pathway toward an unencroached state), i.e. component #9 $\forall\mathsf{G}(\text{🌳} \Rightarrow \exists\mathsf{F}(\neg\text{🌳}))$ of the component graph. The answer to the third query shows that crop cultivation at high altitude (`Alt+` $\wedge$ `Cb-` corresponding to `R18`, `R19`) is the only management policy making bush encroachment always reversible. Although this phenomenon has been observed [LCD18], it is thought to be unfeasible at a large scale in the long term [Lia+20] due to the cost of the required inputs and the tensions between herders and farmers. The fourth query selects the scenarios where at least some encroachment pathways are reversible (from some encroached states, there is a pathway toward an unencroached state), i.e. component #5 $\exists\mathsf{F}\left(\text{🌳} \wedge \exists\mathsf{F}(\neg\text{🌳})\right)$ of the component graph of Figure 4.11. The answer to the fourth query shows that in addition to crop cultivation at high altitude (`Alt+` $\wedge$ `Cb-`), two management policies make some encroachment pathways reversible: the presence of wildlife `Wl+` or browsing livestock `BLv+`. Indeed, pastoralists in Borana have increased their holding of browsing livestock (goats and camels) to mitigate bush encroachment [LC18; Lia+20].

The fifth and sixth queries select the scenarios enabling subsistence. The fifth query selects the scenarios resulting in chronic subsistence (food is always reachable, thus can be reached regularly). The answer to the fifth query shows that three management policies result in chronic subsistence: (1) extensive pastoralism (`Ps+` $\wedge$ `Ig-`), (2) pastoralism with crop cultivation at high altitude (`Alt+` $\wedge$ `Cb-` $\wedge$ `Ps+`), and (3) crop cultivation with wildlife at high altitude (`Alt+`$\wedge$`Cb-`$\wedge$`Wl+`). The first management policy corresponds to the traditional management policy in the Borana zone (nomadic pastoralism [Lia+20]), while the second policy corresponds to one of the current management policies (mixed crop-livestock systems [Lia+20]), the third management policy correspond to crop cultivation with fallow

1) **Reachability pattern:**
$\exists \mathsf{F}\, \texttt{Encroachment}$
An encroached state can be reached.
☑ $\texttt{Ps+} \wedge \texttt{Ig+}$
Encroachment can only happen under the scenarios encompassing pastoralism Ps+ *with intensive grazing* Ig+*.*

2) **Reachability pattern:**
$\exists \mathsf{F}\, \texttt{ClosedCanopyWoodland}$
Closed Canopy Woodland can be reached.
☑ $\texttt{Ps+} \wedge \texttt{Ig+} \wedge (\texttt{Alt+} \vee \texttt{Fb-} \vee \texttt{Wl+} \vee \texttt{BLv+})$
Closed Canopy Woodland can only happen under pastoralism Ps+ *with intensive grazing* Ig+ *and with at least one of the following factors: high altitude* Alt+*, no fire ban* Fb−*, presence of wildlife* Wl+*, browsing livestock* BLv+*.*

3) **Reachability + Consequence pattern:**
$(\exists \mathsf{F}\, \texttt{Encroachment}) \wedge (\forall \mathsf{G}(\texttt{Encroachment} \Rightarrow \exists \mathsf{F}\, \neg \texttt{Encroachment}))$
An encroached state is reachable, and whenever such state is reached, it is possibly followed by an unencroached state.
☑ $\texttt{Ps+} \wedge \texttt{Ig+} \wedge \texttt{Alt+} \wedge \texttt{Cb-}$
If an encroached state is reachable (see output Ps+ ∧ Ig+ *from query 1) and if the system is at high altitude* Alt+ *with crops allowed* Cb−*, then whenever an encroached state is reached it is possibly followed by an unencroached state, i.e. bush encroachment is always reversible.*

4) **Sequence pattern:**
$\exists \mathsf{F}\, (\texttt{Encroachment} \wedge \exists \mathsf{F}\, \neg \texttt{Encroachment})$
An unencroached state is reachable and is possibly preceded at some time by an encroached state, i.e. at least some encroachment pathways are reversible.
☑ $\texttt{Ps+} \wedge \texttt{Ig+} \wedge (\texttt{BLv+} \vee \texttt{Wl+} \vee (\texttt{Alt+} \wedge \texttt{Cb-}))$
*If an encroached state is reachable (*Ps+ ∧ Ig+*, see query 1), there are three set of scenarios where at least some encroachment pathways are reversible: (1) with browsing livestock* BLv+*, (2) with wildlife* Wl+*, (3) at high altitude* Alt+ *with crops allowed* Cb−*.*

5) **Invariance pattern:**
$\forall \mathsf{G}(\exists \mathsf{F}\, \texttt{Subsistence})$
Subsistence states necessarily remain forever reachable.
☑ $(\texttt{Ps+} \wedge \texttt{Ig-}) \vee (\texttt{Alt+} \wedge \texttt{Cb-} \wedge \texttt{Ps+}) \vee (\texttt{Alt+} \wedge \texttt{Cb-} \wedge \texttt{Wl+})$
There are three sets of scenarios where subsistence remains reachable whatever happens : (1) under pastoralism Ps+ *without intensive grazing* Ig−*, (2) at high altitude* Alt+ *with crops allowed* Cb− *and with pastoralism* Ps+ *, or (3) at high altitude* Alt+ *with crops allowed* Cb− *and with wildlife* Wl+*.*

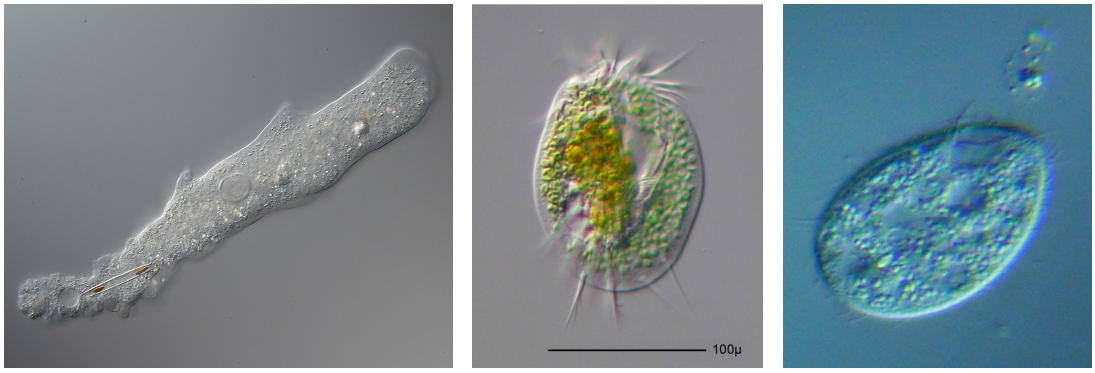
6) **Reachability & Invariance pattern:**
$\exists \mathsf{F}(\exists \mathsf{G}\, \texttt{Subsistence})$
It is possible to reach a state from which the subsistence can persist forever.
☑ $(\texttt{Ps+} \wedge \texttt{BLv+}) \vee (\texttt{Alt-} \wedge \texttt{Ps+}) \vee (\texttt{Fb+} \wedge \texttt{Cb+} \wedge \texttt{Ps+} \wedge \texttt{Ig-})$
There are three sets of scenarios where it is possible to reach a state from which subsistence can persist forever: (1) under pastoralism Ps+ *with browsing livestock* BLv+*, (2) at low altitude* Alt− *with pastoralism* Ps+*, or (3) with fire banned* Fb+ *as well as crops* Cb+ *and with pastoralism* Ps+ *but without intensive grazing* Ig−*.*

Table 4.2: Scenario selection by model-checking. For each of the six queries we show: (1) its pattern type, (2) its CTL formula, (3) its translation into English, (4) ☑ the scenario selection (i.e. control valuations) for which the associated model-checking output of the query is *yes*, (5) an English interpretation of this scenario selection.

periods (which is thought to be unfeasible in the long term in drylands [Lia+20]). The sixth query selects the scenarios enabling continuous subsistence (there is a maximal path along which food is constantly available). The answer to the sixth query shows that three management policies enable continuous subsistence: (1) pastoralism with browsing livestock (Ps+ ∧ BLv+), (2) pastoralism at low altitude (Alt− ∧ Ps+), and (3) extensive pastoralism without crop nor fire (Fb+ ∧ Cb+ ∧ Ps+ ∧ Ig−). This last result should be considered with caution as continuous subsistence may be restricted to a single maximal path, yet uncontrolled events may prevent humans to fully enforce this desired trajectory in a real system.

4.2 Protists assembly model

Protists are microscopic unicellular eukaryotes, see Figure 4.14, that are neither plant, animal nor fungi. From a pool of six protists species: *Amoeba proteus*, *Blepharisma japonicum*, *Colpidium striatum*, *Euplotes patella*, *Paramecium caudatum* and *Tetrahymena pyriformis* (named here A, B, C, E, P, T), Law, Warren and Weatherby studied in laboratory how they can assemble to form a microcosm. In a first experiment [WWL98], they recorded the fate of each of the 2^6 possible combination of species (i.e. community), with six replicates for each combination. In a second experiment [WLW03], they recorded how each stable community, found in the first experiment, responds to an invasion by another species from the pool. Data were recorded from monthly censuses of the replicated communities, documenting the changes in species composition as communities moved along their pathways towards the final persistent communities. This involved systematically scanning the entire microcosm with a microscope and recording the presence/absence of each species. These time series contain useful information about the order in which species were lost, including random variation across replicates. To our knowledge, these experiments are unique in being the only replicated analysis which systematically explores the fate of all possible communities that can be built from a pool of species.



(a) *Amoeba proteus* **(b)** *Euplotes patella* **(c)** *Tetrahymena pyriformis*

Figure 4.14: Pictures of protist species. (a) taken from Wikipedia authored by SmallRex under (CC BY-SA 4.0). (b) taken from Wikipedia authored by Picturepest under (CC BY 2.0). (c) taken from Wikipedia authored by Picturepest under (CC BY 2.0).

Experimental communities were constructed from combinations of the six species of protists from the pool plus a mixed bacterial flora feeding them. For each combination, approximately 100 individuals of each species were introduced into the microcosm, along

with a mixture of bacteria. Protist species were chosen to exhibit a range of sizes and trophic strategies, and on the basis that they were all able to persist when grown under the same environmental and nutrient medium conditions in the laboratory. The interaction network of these species (trophic and competition relations), see Figure 4.15, was inferred by [Her+22] from the trophic network of [WWL98, Fig.1] and from the outcomes of the single species and two-species replicates of the first experiment [WWL98].

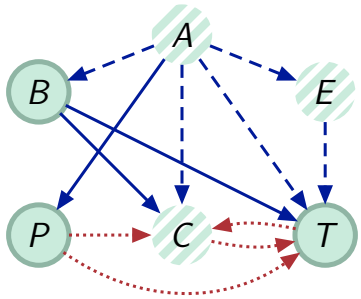


Figure 4.15: Protists interaction network. Solid blue arrows are predations (dashed if the predator cannot sustain by feeding only on this prey, and thus need other prey to survive). Dotted red arrows are competitions. Species in a striped borderless circle are unable to survive alone by feeding only on bacteria. Trophic links, i.e. predations, were inferred from the outcome of the two-species replicates and from the trophic network of [WWL98, Fig.1]. Trophic link are dashed if the predator disappeared before its prey in some replicates. Competition links were inferred from the remaining outcomes of the two-species replicates once the predation links had been inferred. Species unable to survive alone were inferred from the outcome of single species replicates. Taken from [Her+22].

4.2.1 Modelling

> *"Can the behaviour of the system be characterized by a simple set of rules? To what extent does knowledge of the results from the pairwise species combinations allow prediction of the outcomes of the more species-rich sets?" [WWL98]*

Based on the protists pairwise interaction network, see Figure 4.15, we built an RR description of the protists' community pathways recorded in the first experiment [Her+22]. The protists model consists of 6 variables (one for each protist species) and 15 rules, see Figure 4.16. The protists model's ruleset was derived from the protists interaction network upon the following scheme:

- Predation: if P predates N (solid blue arrow from P to N), then we add the rule: `[predation]P+>>N-`.
- Predation: if P predates secondarily N_s (dashed blue arrow from P to N_s) and if P predates $N_1, \ldots, N_k$ (solid blue arrows from P to $N1, \ldots, Nk$), then we add the rule: `[predation]P+`, `N1-`, $\ldots$, `Nk->>Ns-`.
- Starvation: if P predates $N_1, \ldots, N_k$ (solid blue arrows from P to $N_1, \ldots, N_k$) and if P cannot survive by feeding on bacteria (P in a striped borderless circle), then we add the rule: `[starvation]N1-`, $\ldots$, `Nk->>P-`.
- Starvation: if P does not predate any prey (no solid blue arrow from P) and if P

cannot survive by feeding on bacteria (P in a striped borderless circle), then we add the rule: `[starvation]P+>>P-`.

- Competition: if C_1 competes with C_2 (dotted red arrow from C_1 to C_2), then we add the rule: `[competition]C1+>>C2-`.

Each state of the protists model represents a community (i.e. a combination of species). Like in the first experiment [WWL98], every possible community is taken as an initial state, hence the `*` initial value of the variables. In the first experiment, species can only go extinct, the rules therefore define the conditions in which they can disappear.

```
variables:
    A*: Amoeba proteus
    B*: Blepharisma japonicum
    C*: Colpidium striatum
    E*: Euplotes patella
    P*: Paramecium caudatum
    T*: Tetrahymena pyriformis
```

```
rules:
  [predation] A+ >> P-          # R1
  [predation] B+ >> C-          # R2
  [predation] B+ >> T-          # R3
  [predation] A+, P- >> B-      # R4
  [predation] A+, P- >> C-      # R5
  [predation] A+, P- >> T-      # R6
  [predation] A+, P- >> E-      # R7
  [predation] E+ >> T-          # R8
  [starvation] P- >> A-         # R9
  [starvation] E+ >> E-         # R10
  [starvation] C+ >> C-         # R11
  [competition] P+ >> C-        # R12
  [competition] P+ >> T-        # R13
  [competition] C+ >> T-        # R14
  [competition] T+ >> C-        # R15
```

Figure 4.16: RR protists model.

The STG computed from the protists model using `ecco` [PTG22a], see Figure 4.17, has $2^6 = 64$ states and 135 transitions, 71 of them (53%) were recorded in the first experiment [WWL98]. The protists model predicts all experimentally recorded transitions, which is a necessary condition for validating the model. Note that this STG consists of three disconnected subgraphs, meaning that the system cannot shift from one subgraph to another without species invasion.

Two facts may explain why some of the predicted transitions were never recorded in the first experiment. First, transitions with very low probabilities may not be observed due to the finite number of replicates. In particular, as the states with many initial species always collapse towards states with fewer species; transitions concerning the latter are therefore observed more often. For example, the state $\{A, B\}$ was recorded 77 times along the community pathways of the first experiment. Starting from this state, the transition $\{A, B\} \to \{B\}$ was observed 73 times and the transition $\{A, B\} \to \{A\}$ was observed only 3 times. Such an infrequent transition would probably not have been recorded if $\{A, B\}$ had been recorded only 6 times like $\{A, B, C, E, P, T\}$ for example. Secondly, some transitions were too brief to be recorded in the experiment. For example, all replicates starting from $\{A, B, C, E, P, T\}$ had only 4 species left at the first census. The transitions to the 5-species states were therefore not recorded even if they must have happened (assuming that two species do not disappear exactly at the same time).

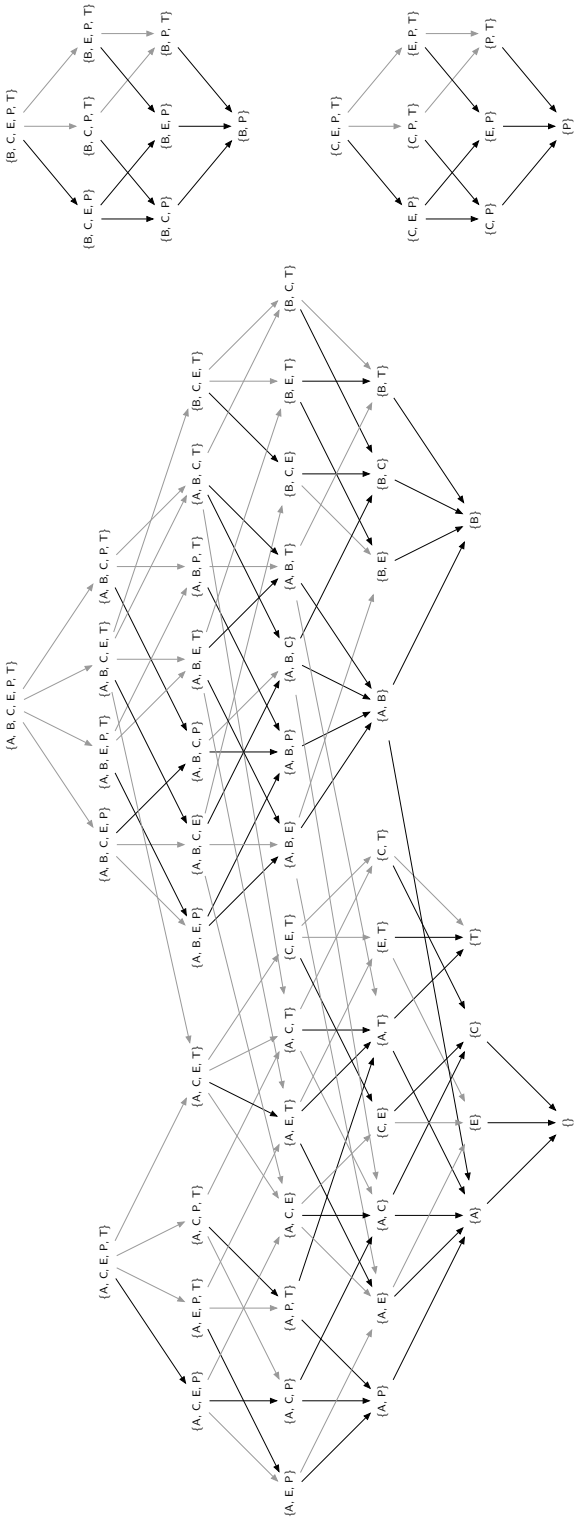


Figure 4.17: Protists model STG. The states represent the 64 protist communities, labelled by the initials of the present species between brackets. Black transitions were recorded during the first experiment [WWL98], while grey transitions were predicted by the model but were not observed in the first experiment. Adapted from [Her+22].

The STG of the protists model, see Figure 4.17, has 5 dead-ends denoting stable communities: $\{\}$, $\{B\}$, $\{P\}$, $\{T\}$ and $\{B,P\}$. The first experiment [WWL98] found 3 additional stable communities: $\{C\}$, $\{C,P\}$ and $\{P,T\}$. Indeed $\{C\} \not\to$ was often experimentally observed, yet in some other replicates $\{C\} \to \{\}$. Similarly in some replicates $\{C,P\} \not\to$ or $\{P,T\} \not\to$, yet in others $\{C,P\} \to \{P\}$ or $\{P,T\} \to \{P\}$ was observed. Thus $\{C\}$, $\{C,P\}$ and $\{P,T\}$ should not be considered as dead-ends in the protists model STG, yet we would like to be able to represent the fact that the system can stop in these states for some maximal paths. We will discuss in Chapter 7 how we could achieve this goal.

A model of the second experiment need to have rules representing invasions: `[invasion]X->>X+`. Seeing the STG of the protists model (Figure 4.17) as a kind of "assembly pinball" in which the ball/community is put in an initial state before rolling down toward one of the bottom states, invasion rules are the flippers making the ball bounce up toward the top of the STG. But the use of such "flippers" need to be restricted, otherwise they would create paths from any state to many upper states, blurring the assembly dynamics. We will see in the next chapter how to properly manage such "flippers" such that the occurrences of invasion events are controlled along the dynamics.

4.2.2 Results

In this section, we progressively build a component graph aiming to provide an answer to two questions proposed by [WWL98]:

- *From how many initial states are each of the stable communities obtained?*
- *What are the impacts of individual species on community collapse?*

These two questions have already been answered experimentally in [WWL98]. Our goal is to show how formal modelling and analysis are fitted to answer the questions raised by community assembly graphs, and could complement experimental studies with a modelling framework able to forecast the results of a particular community assembly from an interaction network.

Starting from the whole state space $\mathcal{S}$ of the protists model, we incrementally refine a partition answering the first question. The first step is to split the stable communities apart, i.e. the dead-ends $s \not\to$. To do so, we use the topological property "dead-end", or equivalently the CTL formula $\neg\exists \mathsf{X} \top$. Using this property, we can partition $\mathcal{S}$ between the states that are dead-ends (component #2) and these that are not (component #1), see Figure 4.18. We see that the system can move from component #1 to component #2 (and as expected, not the other way around because the system cannot escape a dead-end).

We are interested in each stable community individually. To highlight the disparities between the stable communities, we explicit component #2 `dead-end`, i.e. we split its states individually, see Figure 4.19. The resulting component graph has 5 new components corresponding to the 5 dead-ends of the system: $\{\}$, $\{B\}$, $\{P\}$, $\{T\}$ and $\{B,P\}$.

The final step is to partition the states of component #1 $\neg$`dead-end` with respect to the stable communities they lead to. To do so, we split component #1 $\neg$`dead-end` into the topological basins of the 5 dead-ends: $\{\}$, $\{B\}$, $\{P\}$, $\{T\}$ and $\{B,P\}$, and merge each

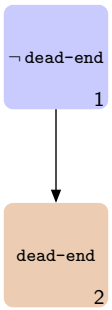


Figure 4.18: First splitting step.

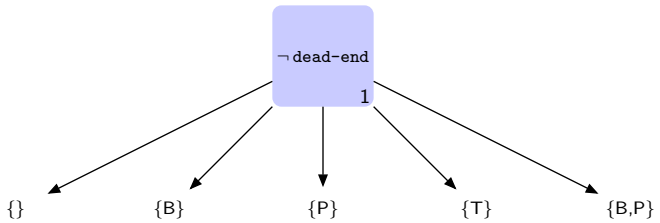


Figure 4.19: Second splitting step.

dead-end with the basin leading solely to it. This procedure results in the final component graph of the protists model [Her+22], see Figure 4.20.

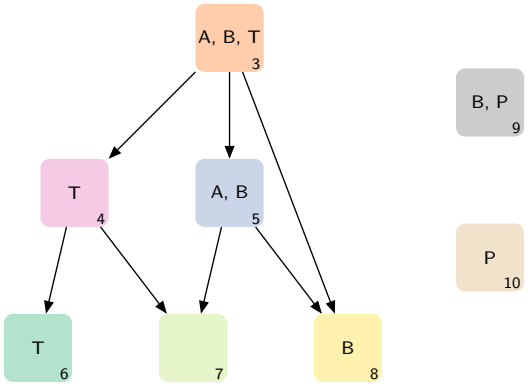
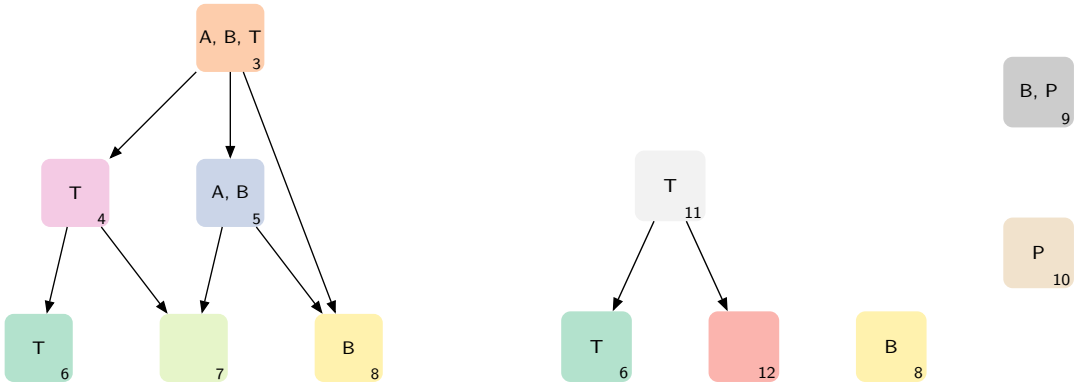


Figure 4.20: Final component graph. Each component is labelled with the species present in every state of the component. Adapted from [Her+22].

This final component graph answers the first question: "*From how many initial states are each of the stable communities obtained?*", because the number of initial states leading to each stable community is exactly the size of the components leading to the corresponding dead-ends. Indeed, each dead-end is merged with the states leading solely to it, so for example the states leading solely to the dead-end $\{B\}$ are the states of component #8 labelled B. When a state can lead to distinct dead-ends, then it is merged with the other states leading exactly to the same dead-ends, for example the states leading exactly to the dead-ends $\{\}$ and $\{B\}$ are the states of component #5 labelled A, B. While the states leading toward dead-end $\{P\}$ or the states leading toward dead-end $\{B, P\}$ cannot lead to any other dead-end, i.e. the end point of their dynamics is deterministic, the states of component #3 labelled A, B, T can lead to either dead-ends $\{\}$, $\{B\}$ or $\{T\}$. A version of the protists model STG where each state is coloured with the colour of the component it belongs to in the component graph is given in Figure 4.22. The component graph of

Figure 4.20 can be seen as a condensed version of this STG where the information about the dead-ends reachability has been summed up.

In order to answer the second question: "*What are the impacts of individual species on community collapse?*", we can compare two component graphs built following the same steps as earlier but from two versions of the protists model where a focused species is either always present or always absent initially. For example, as A is the top predator, it should strongly influence the system's dynamics. Thus focusing on A we can build two versions of the protists model where A is either always present initially `A+` or always absent initially `A-`, and build two component graphs following the same step as earlier, see Figure 4.21. When A is always present initially, see Figure 4.21a, the branching dynamics is rediscovered where the reached stable community is not determined initially, i.e. in components #3, #4 and #5. Conversely, when A is always absent initially, see Figure 4.21b, the reached stable community is mainly determined initially, except in component #11. Thus A seems to influence the community collapse by adding non-determinism to the reached stable community.



(a) A always present initially **(b)** A always absent initially

Figure 4.21: Component graphs for which A is either initially present or absent. Each component is labelled with the species present in every state of the component. Recall that when several components of distinct component graphs encompass the exact same set of states, they are all given the same number and the same colour across all component graphs.

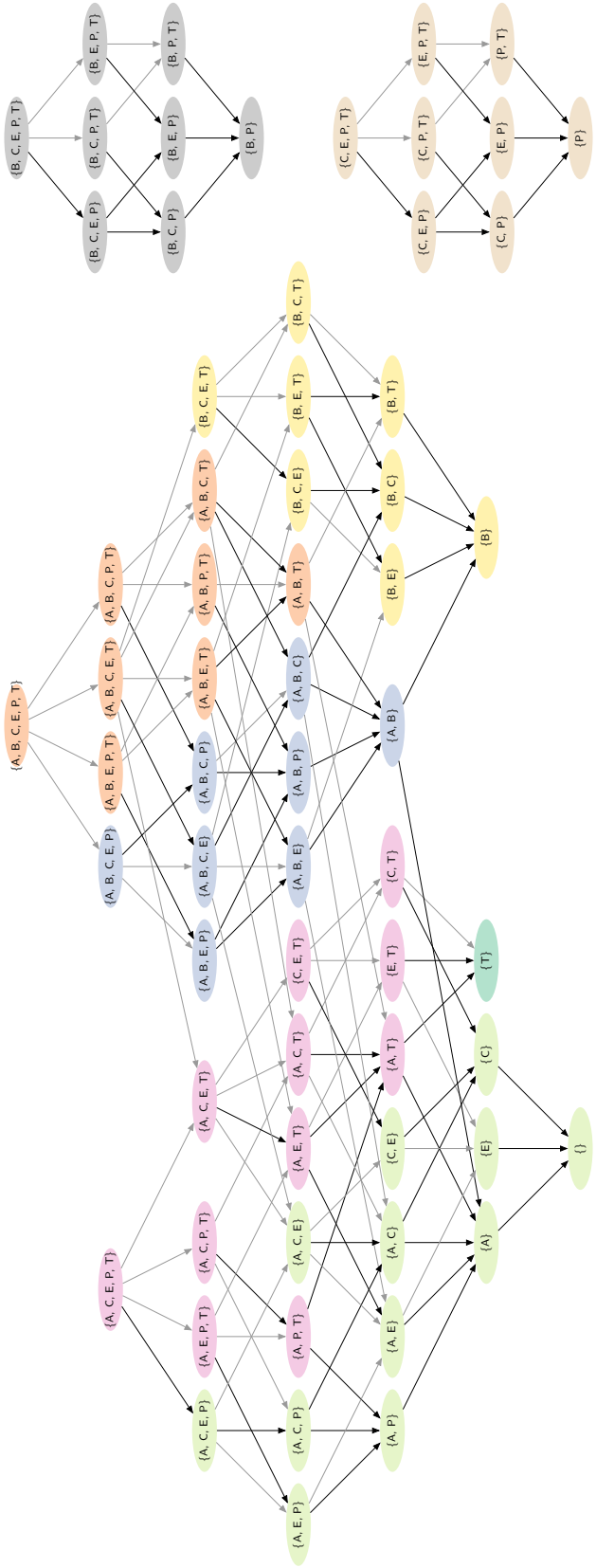


Figure 4.22: Protists model STG colored by component. Adapted from [Her+22].

5 Symbolic model-checking of Fair ARCTL

Summary

This chapter presents the Action-Restricted Computation Tree Logic (ARCTL), an extension of CTL allowing to restrict the maximal paths to a subset of enabled actions along each quantifier of a formula, i.e. $\exists$ or $\forall$. We then extend ARCTL with fairness restrictions on quantifiers, i.e. "realism" constraints upon order and occurrence rate of events along a path, resulting in Fair ARCTL (FARCTL). Consequently, FARCTL quantifiers allow to restrict maximal paths with respect to both enabled actions and fairness constraints. Such double restrictions can be used to model ecosystem management scenarios for example. FARCTL can thus be used to to shift between them in a controlled manner at the level of its quantifiers. Finally, we provide a symbolic model-checking algorithm for FARCTL that is implemented inside ecco.

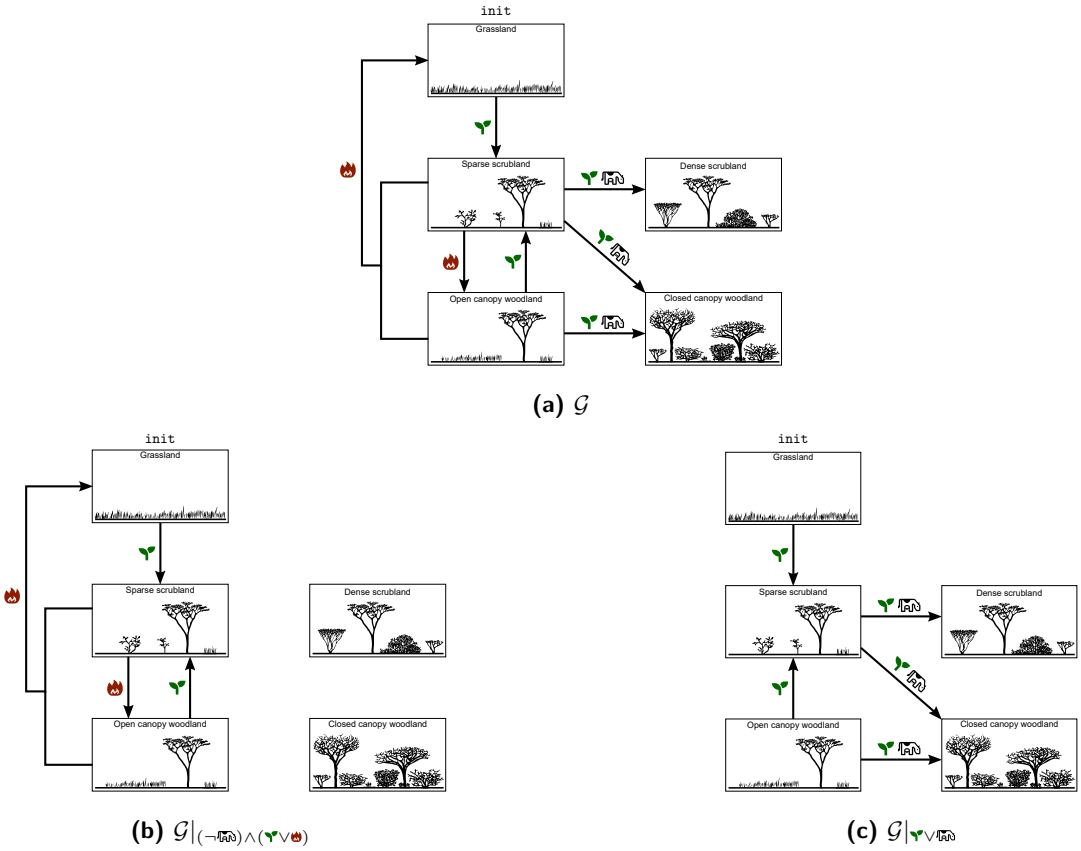
In the previous chapter, we saw that in ecological applications, we often want to restrict the set of maximal paths to represent particular scenarios (for example, ecosystem management policies or invasion events). Scenarios consist mainly of: (1) a list of enabled events, and (2) a list of disabled events; for example intensive grazing with fire banned. One way to achieve this is to build disconnected STGs with distinct sets of maximal paths (as we did in the *Borana model* in Chapter 4). The drawbacks of this method are that it increases the size of the state space $\mathcal{S}$ (because large portions of the disconnected STGs are just replicated between them), and more importantly that the system cannot shift between scenarios along the dynamics. Another way is to create a single STG encompassing all the scenarios, that is then *restricted* to particular scenarios by disabling some actions. This method does not duplicate the state space, and more importantly the restriction can change along the dynamics as the system shifts between scenarios.

Definition 5.1 ($\mathcal{G}|_\alpha, \Pi|_\alpha$)**.** Given an action formula $\alpha : 2^{\mathcal{P}_\mathcal{A}} \mapsto \mathbb{B}$ and an STG $\mathcal{G} = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \to, \mathcal{V}_\mathcal{S}, \mathcal{V}_\mathcal{A})$, the *$\alpha$-restriction of $\mathcal{G}$* is the STG $\mathcal{G}|_\alpha = (\mathcal{S}, \mathcal{S}_0, \mathcal{A}, \to\,|_\alpha, \mathcal{V}_\mathcal{S}, \mathcal{V}_\mathcal{A})$, where:

$$\to\,|_\alpha \stackrel{\text{def}}{=} \{(s, a, s') \in \to \mid a \models \alpha\}$$

The set of maximal α-restricted paths of $\mathcal{G}$ is noted $\Pi|_\alpha(\mathcal{G}) = \Pi(\mathcal{G}|_\alpha)$.

Example 5.1. Examples of α-restrictions are given in Figure 5.1. Note that Figure 5.1b is exactly the STM before livestock introduction (Figure 1.4a), and Figure 5.1c is exactly the STM with livestock and fire ban (Figure 1.4b). Figure 5.1b and Figure 5.1c are α-restrictions of Figure 5.1a, thus there is no replication and they all share the same states. By changing the α-restriction along the dynamics, we can for example test if a path travelled in one α-restriction can be reversed in another α'-restriction. For example, we can test if the encroachment reached with intensive grazing can be reversed by enabling fire.



(a) $\mathcal{G}$

(b) $\mathcal{G}|_{(\neg \text{🔥}) \wedge (\text{🌱} \vee \text{🔥})}$

(c) $\mathcal{G}|_{\text{🌱} \vee \text{🔥}}$

Figure 5.1: Examples of α-restrictions. Adapted from [LC18].

Ecological scenarios sometimes also encompass the concept of *fairness* [BK08]. For example, we may want to enforce that a path cannot infinitely carry on without fire happening. In this case, we say that the path is *fair* with respect to fire, meaning that fire happens infinitely often if the path if infinite. Thus, we may also want to restrict the set of maximal paths to fair paths.

In this chapter, we first present the extension of CTL using the concept of α-restriction: Action-Restricted CTL (ARCTL) [PR07], and then we extend it with the concept of fairness: Fair ARCTL (FARCTL). In FARCTL, a restriction upon actions and fairness is added to each quantifier ($\exists, \forall$), enabling the system to shift between scenarios along the formula. Lastly, we provide a symbolic algorithm to compute the set of states satisfying an FARCTL formula, that is implemented inside `ecco`. FARCTL model-checking will be applied in Chapter 6 to the case studies of Chapter 4.

5.1 ARCTL

Action-Restricted Computation Tree Logic (ARCTL) [PR07] is the extension of CTL dealing with α-restrictions. The quantifiers of CTL ($\exists, \forall$) are extended with a restriction upon actions, meaning that they quantify over the maximal paths of $\mathcal{G}|_\alpha$ and not of $\mathcal{G}$. Thus the α-restriction can change along the formula, corresponding to shifts between scenarios dur-

ing the dynamics. ARCTL has been used in systems biology, for example to assess lymphocyte differentiation pathways [Abo+15], but to our knowledge, it has never been used for ecological applications.

5.1.1 Syntax and semantics

The syntax of ARCTL extends the syntax of CTL with action formulas, i.e. Boolean formulas over action properties, and action restrictions on quantifiers.

Definition 5.2 (ARCTL syntax)**.** The syntax of *ARCTL* is given by the following grammar over state, action and path formulas:

- state formulas ($ARCTL_{\mathcal{S}}$): $\varphi \stackrel{\text{def}}{=} \top_{\mathcal{S}} \mid p_{\mathcal{S}} \in \mathcal{P}_{\mathcal{S}} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists_\alpha \gamma \mid \forall_\alpha \gamma$
- action formulas ($ARCTL_{\mathcal{A}}$): $\alpha \stackrel{\text{def}}{=} \top_{\mathcal{A}} \mid p_{\mathcal{A}} \in \mathcal{P}_{\mathcal{A}} \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2$
- path formulas ($ARCTL_{\Pi}$): $\gamma \stackrel{\text{def}}{=} \mathsf{X}\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi$

An ARCTL formula is a state formula $\varphi \in ARCTL_{\mathcal{S}}$.

The semantics of ARCTL extends the semantics of CTL by quantifying over maximal α-restricted paths instead of just maximal paths.

Definition 5.3 (ARCTL semantics)**.** Let $s \in \mathcal{S}$ a state:

- $s \models \top_{\mathcal{S}}$
- $s \models p_{\mathcal{S}}$ iff $p_{\mathcal{S}} \in \mathcal{V}_{\mathcal{S}}(s)$
- $s \models \neg\varphi$ iff $s \not\models \varphi$
- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$
- $s \models \varphi_1 \vee \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$
- $s \models \exists_\alpha \gamma$ iff $\exists \pi \in \Pi|_\alpha(s)$ such that $\pi \models \gamma$
- $s \models \forall_\alpha \gamma$ iff $\forall \pi \in \Pi|_\alpha(s)$ we have $\pi \models \gamma$

Let $a \in \mathcal{A}$ an action:

- $a \models \top$
- $a \models p_{\mathcal{A}}$ iff $p_{\mathcal{A}} \in \mathcal{V}_{\mathcal{A}}(a)$
- $a \models \neg\alpha$ iff $a \not\models \alpha$
- $a \models \alpha_1 \wedge \alpha_2$ iff $a \models \alpha_1$ and $a \models \alpha_2$
- $a \models \alpha_1 \vee \alpha_2$ iff $a \models \alpha_1$ or $a \models \alpha_2$

Let $\pi \in \Pi|_\alpha(\mathcal{G})$ a maximal path:

- $\pi \models \mathsf{X}\varphi$ iff $|\pi| \geqslant 1$ and $\pi[1]_{\mathcal{S}} \models \varphi$
- $\pi \models \mathsf{F}\varphi$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi|$ and $\pi[i]_{\mathcal{S}} \models \varphi$
- $\pi \models \mathsf{G}\varphi$ iff $\forall i \leqslant |\pi|$ we have $i \in \mathbb{N} \Rightarrow \pi[i]_{\mathcal{S}} \models \varphi$
- $\pi \models \varphi_1 \mathsf{U} \varphi_2$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi| \quad \pi[i]_{\mathcal{S}} \models \varphi_2$ and $\forall 0 \leqslant j < i \quad \pi[j]_{\mathcal{S}} \models \varphi_1$

Note that the ARCTL formulas $\exists_\top\gamma$ and $\forall_\top\gamma$ are strictly equivalent to the CTL formulas $\exists\gamma$ and $\forall\gamma$ respectively. Hence why ARCTL is an extension of CTL. In the following we will write $\exists\gamma$ for $\exists_\top\gamma$, and $\forall\gamma$ for $\forall_\top\gamma$, when convenient.

Definition 5.4. Given an STG $\mathcal{G}$ and an ARCTL formula φ, the *ARCTL model-checking problem* consists in determining if $\mathcal{S}_0 \subseteq \{s \in \mathcal{S} \mid s \models \varphi\}$, noted $\mathcal{G} \models \varphi$.

Example 5.2. For example in the STG of Figure 5.1, using the invariance pattern $\forall\mathsf{G}(\exists\mathsf{F}x)$, i.e. x *states necessarily remain forever reachable*:

- In the $(\neg$🏘$)\wedge($🌱$\vee$🔥$)$ scenario, a state without tree remains reachable whatever happens : $\mathcal{G} \models \forall_{(\neg🏘)\wedge(🌱\vee🔥)}\mathsf{G}(\exists_{(\neg🏘)\wedge(🌱\vee🔥)}\mathsf{F}\,\neg$🌳$)$.
- It is not the case in the 🌱$\vee$🏘 scenario: $\mathcal{G} \not\models \forall_{🌱\vee🏘}\mathsf{G}(\exists_{🌱\vee🏘}\mathsf{F}\,\neg$🌳$)$. Indeed, in $\mathcal{G}|_{🌱\vee🏘}$, *Dense scrubland* and *Closed canopy woodland* can be reached from which there is no path removing trees.
- It is still not the case even if we allow to switch from 🌱$\vee$🏘 to $(\neg$🏘$)\wedge($🌱$\vee$🔥$)$ in order to try to reach a state without tree: $\mathcal{G} \models \forall_{🌱\vee🏘}\mathsf{G}(\exists_{(\neg🏘)\wedge(🌱\vee🔥)}\mathsf{F}\,\neg$🌳$)$. Indeed, in $\mathcal{G}|_{🌱\vee🏘}$, *Dense scrubland* or *Closed canopy woodland* can be reached from which there is no path in $\mathcal{G}|_{(\neg🏘)\wedge(🌱\vee🔥)}$ removing trees. Thus the $(\neg$🏘$)\wedge($🌱$\vee$🔥$)$ scenario can remove trees, but it cannot revert the tree encroachment caused by the 🌱$\vee$🏘 scenario.

Technical remark

Note that $\exists_\alpha\mathsf{G}\varphi$ holds in particular if there is a finite maximal α-path where φ holds. In that regard, an α-dead-end $s \not\to|_\alpha$ is considered a self-loop, i.e. an infinite path remaining in the same state ($s \xrightarrow{\perp_\mathcal{A}} s \xrightarrow{\perp_\mathcal{A}} s \xrightarrow{\perp_\mathcal{A}} \ldots$), just like we did for CTL. In contrast, we can define $\exists_\alpha\mathsf{G}^\infty\varphi$ that holds only for infinite maximal α-paths: $\exists_\alpha\mathsf{G}^\infty\varphi \overset{\text{def}}{=} \exists_\alpha\mathsf{G}(\varphi \wedge \exists_\alpha\mathsf{X}\top)$.

Note also that we could have added a real $\perp_\mathcal{A}$-labelled self-loop to each dead-end of $\mathcal{G}$ beforehand to make every maximal path of $\mathcal{G}$ infinite. But this procedure cannot be extended to ARCTL as each α-restriction along the formula may produce new α-dead-ends that cannot be complemented with $\perp_\mathcal{A}$-labelled self-loops beforehand. For that reason, we chose to define these self-loops semantically, i.e. without adding real transitions to the STG, both for CTL and ARCTL.

5.1.2 ARCTL symbolic model-checking

Like with CTL, there is a symbolic perspective with ARCTL. Instead of defining the semantics of ARCTL formulas over states and paths individually, it can be given over sets of states:

$$\varphi = \{s \in \mathcal{S} \mid s \models \varphi\} \subseteq \mathcal{S}$$

Like CTL operators, ARCTL operators can be defined using state properties, set operations, fixed points and α-restrictions of the predecessor function $Pred_\alpha$:

$$Pred_\alpha : Z \in \mathcal{P}(\mathcal{S}) \mapsto \{s \in \mathcal{S} \mid \exists s' \in Z \quad s \xrightarrow{\alpha} s'\} \in \mathcal{P}(\mathcal{S})$$

The symbolic semantics of ARCTL is the symbolic semantics of CTL in $\mathcal{G}|_\alpha$, thus the definitions of its operators are exactly the same as the symbolic semantics of CTL operators (Table 2.2). This set-based perspective provides a symbolic algorithm to compute the set of states satisfying an ARCTL formula, see Table 5.1. Just like with CTL, component graphs can be built by partitioning the state space $\mathcal{S}$ using ARCTL formulas.

	$\exists_\alpha$	$\forall_\alpha$
X	$\exists_\alpha \mathsf{X}\varphi = Pred_\alpha(\varphi)$	$\forall_\alpha \mathsf{X}\varphi = \exists_\alpha \mathsf{X}\top \wedge \neg\exists_\alpha \mathsf{X}\neg\varphi$
U	$\exists_\alpha(\varphi_1 \mathsf{U} \varphi_2) = \mu Z.\varphi_2 \vee (\varphi_1 \wedge \exists_\alpha \mathsf{X} Z)$	$\forall_\alpha(\varphi_1 \mathsf{U} \varphi_2) = \mu Z.\varphi_2 \vee (\varphi_1 \wedge \forall_\alpha \mathsf{X} Z)$
F	$\exists_\alpha \mathsf{F}\varphi = \mu Z.\varphi \vee \exists_\alpha \mathsf{X} Z$	$\forall_\alpha \mathsf{F}\varphi = \mu Z.\varphi \vee \forall_\alpha \mathsf{X} Z$
G	$\exists_\alpha \mathsf{G}\varphi = \nu Z.\varphi \wedge (\exists_\alpha \mathsf{X} Z \vee \neg\exists_\alpha \mathsf{X}\top)$	$\forall_\alpha \mathsf{G}\varphi = \nu Z.\varphi \wedge (\forall_\alpha \mathsf{X} Z \vee \neg\exists_\alpha \mathsf{X}\top)$

Table 5.1: Fixed point definitions of ARCTL operators.

5.2 Fairness and FARCTL

In this section, we extend ARCTL with the notion of fairness, meaning that we restrict the set of maximal paths to these considered realistic. A path is considered realistic if some given events happen regularly, such a path is then called *fair* towards these events. From a set of fairness constraints concerning distinct events, we build a fairness assumption that merges all these constraints. The set of maximal paths can then be restricted with respect to this fairness assumption. Finally, we extend ARCTL quantifiers ($\exists, \forall$) to make use of the restricted set of maximal fair paths, resulting in Fair ARCTL. FARCTL thus provides restrictions of the set of maximal paths based both upon action constraints and fairness constraints.

5.2.1 Fairness constraint

A fair path is characterized by the fact that various fairness constraints are fullfilled [BK08]. Fairness constraints deal with the order and occurrence rate of certain events, i.e. states properties (e.g. 🍸, $\exists\mathsf{F}\neg$🍸, etc...) or actions properties (e.g. 🔥, 🌱$\vee$🚒, etc...), along maximal paths. Fairness constraints are classically divided into three classes:

- Unconditional fairness: “event e happens infinitely often (with arbitrary long breaks between occurrences)”
- Weak fairness (justice): “if event e_1 happens continuously (i.e. without breaks) from a certain time on, then event e_2 happens infinitely often”
- Strong fairness (compassion): “if event e_1 happens infinitely often, then event e_2 happens infinitely often”

Example 5.3. Examples of fairness constraints:

- Unconditional fairness: “fire happens infinitely often”.

- Weak fairness: "if cattle are present continuously, then events related to intensive grazing happen infinitely often".
- Strong fairness "if herbaceous fuel is present infinitely often, then fires happen infinitely often".

Maximal fair paths are classically defined to be infinite. We extend this notion to finite maximal fair paths. Note that α-restrictions can produce α-dead-ends, i.e. $s \in \mathcal{S}$ where $s \not\rightarrow|_\alpha$ noted $s \overset{\alpha}{\not\rightarrow}$, and thus finite maximal α-restricted paths. As for the $\exists_\alpha$G operator, we consider dead-ends as self-loops, i.e. infinite paths remaining in the same state ($s \xrightarrow{\perp_\mathcal{A}} s \xrightarrow{\perp_\mathcal{A}} s \xrightarrow{\perp_\mathcal{A}} \ldots$) without any action happening (hence the $\perp_\mathcal{A}$ labelling the transitions). Thus a finite maximal path π is unconditionally fair iff event e happens infinitely often along the self-loop. Similarly a finite maximal path π is weakly (resp. strongly) fair iff event e_1 does not happen continuously (resp. infinitely often) along the self-loop, or event e_2 happens infinitely often along the self-loop. Note that action-events cannot happen continuously, nor infinitely often, along a self-loop. Indeed no action happens at all along a self-loop (hence the $\perp_\mathcal{A}$ labelling the transitions). Based on this idea, we now give a formal definition of events and fairness constraints.

Definition 5.5. A *state-event* $e_\mathcal{S}$ is an ARCTL state formula $e_\mathcal{S} \in ARCTL_\mathcal{S}$ representing a set of states. An *action-event* $e_\mathcal{A}$ is an ARCTL action formula $e_\mathcal{A} \in ARCTL_\mathcal{A}$ representing a set of actions.

Technical remark

A standard way of handling action-events is by adding the last action into the current state [BK08; PR07], i.e. $s \xrightarrow{a} s'$ becomes $(s, *) \rightarrow (s', a)$. The STG $\mathcal{G}$ is transformed into a Kripke Structure by transforming $\mathcal{S}$ into $\mathcal{S}' = (\mathcal{S} \times \mathcal{A})$, and $\rightarrow$ into $\rightarrow' = \{((a, s), (a', s')) \mid (s, a', s') \in \rightarrow\}$. This transformation leads to a blow-up of the size of the state space by the size of the set of actions. We will avoid this blow-up by handling directly the $\xrightarrow{a}$ transition relations, as in the ARCTL semantics ($s \xrightarrow{\alpha} s'$ iff $s' \models \exists_\alpha X(\{s\})$).

We define two operators, $\overset{\infty}{\exists}$ and $\overset{\infty}{\forall}$, meaning that an event happens either "infinitely often" or "continuously from a certain time on" along a maximal path. Their semantics is defined to consider dead-ends as self-loops, with subtle differences between state-events that can happen infinitely often or continuously along self-loops, and action-events that cannot.

Definition 5.6 ($\overset{\infty}{\exists}, \overset{\infty}{\forall}$)**.** Let π be a maximal path and $e_\mathcal{S} \in ARCTL_\mathcal{S}$ a state-event:

- $\pi \models \overset{\infty}{\exists}\, e_\mathcal{S}$ iff $\forall i \in \mathbb{N} \quad i \leqslant |\pi| \Rightarrow \exists j \in \mathbb{N}$ such that $i \leqslant j \leqslant |\pi|$ and $\pi[j]_\mathcal{S} \models e_\mathcal{S}$
- $\pi \models \overset{\infty}{\forall}\, e_\mathcal{S}$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi|$ and $\forall j \in \mathbb{N} \quad i \leqslant j \leqslant |\pi| \Rightarrow \pi[j]_\mathcal{S} \models e_\mathcal{S}$

Let π be a maximal path and $e_\mathcal{A} \in ARCTL_\mathcal{A}$ an action-event:

- $\pi \models \overset{\infty}{\exists}\, e_\mathcal{A}$ iff $|\pi| = \infty$ and $\forall i \in \mathbb{N} \quad \exists j \geqslant i$ such that $\pi[j]_\mathcal{A} \models e_\mathcal{A}$
- $\pi \models \overset{\infty}{\forall}\, e_\mathcal{A}$ iff $|\pi| = \infty$ and $\exists i \in \mathbb{N}$ such that $\forall j \geqslant i \quad \pi[j]_\mathcal{A} \models e_\mathcal{A}$

We can now give a concise formal definition of fairness constraints encompassing either state-events, action-events, or mixing them.

Definition 5.7 ($\mathcal{F}_U, \mathcal{F}_W, \mathcal{F}_S$)**.** Let e_1, e_2 be state-events or action-events, and π a maximal path. *Fairness constraints* are divided into three classes of path properties:

- Unconditional fairness: $\pi \models \mathcal{F}_U(e_1)$ iff $\pi \models^{\infty}_{\exists} e_1$
- Weak fairness: $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(\pi \models^{\infty}_{\forall} e_1\right) \Rightarrow \left(\pi \models^{\infty}_{\exists} e_2\right)$
- Strong fairness: $\pi \models \mathcal{F}_S(e_1, e_2)$ iff $\left(\pi \models^{\infty}_{\exists} e_1\right) \Rightarrow \left(\pi \models^{\infty}_{\exists} e_2\right)$

Proposition 5.1.

$$\mathcal{F}_U(e_1) \Rightarrow \mathcal{F}_S(e_2, e_1) \Rightarrow \mathcal{F}_W(e_2, e_1)$$

Proof.

- $\mathcal{F}_U(e_1) \Rightarrow \mathcal{F}_S(e_2, e_1)$:

$$a \Rightarrow b \text{ means } \neg a \vee b \text{ thus } \mathcal{F}_S(e_2, e_1) = \neg\left(\pi \models^{\infty}_{\exists} e_2\right) \vee \mathcal{F}_U(e_1)$$

- $\mathcal{F}_S(e_2, e_1) \Rightarrow \mathcal{F}_W(e_2, e_1)$:

$$\neg\left(\pi \models^{\infty}_{\exists} e_2\right) \Rightarrow \neg\left(\pi \models^{\infty}_{\forall} e_2\right)$$

□

Example 5.4. If we restrict fairness constraints to CTL state-events and infinite maximal paths, then the presented semantics is equivalent to the classical semantics of CTL fairness [BK08, chap. 6.5] that can be expressed as LTL formulas:

- Unconditional fairness: $\mathcal{F}_U(e_1) = \mathsf{GF}(e_1)$
- Weak fairness: $\mathcal{F}_W(e_1, e_2) = \mathsf{FG}(e_1) \Rightarrow \mathsf{GF}(e_2)$
- Strong fairness: $\mathcal{F}_S(e_1, e_2) = \mathsf{GF}(e_1) \Rightarrow \mathsf{GF}(e_2)$

Example 5.5. Classical action-based fairness [BK08, chap. 3.5] can also be expressed by mixing state-events ($\exists_\alpha \mathsf{X}\top$, i.e. an α-action is enabled) and action-events:

- Unconditional fairness: $\mathcal{F}_U(\alpha)$ i.e. "α-actions happen infinitely often"
- Weak fairness: $\mathcal{F}_W(\exists_\alpha \mathsf{X}\top, \alpha)$ i.e. "if α-actions are enabled continuously, then α-actions happen infinitely often"
- Strong fairness: $\mathcal{F}_S(\exists_\alpha \mathsf{X}\top, \alpha)$ i.e. "if α-actions are enabled infinitely often, then α-actions happen infinitely often"

5.2.2 Fairness assumption

Fair paths are often defined by several independent fairness constraints. Fairness is then defined by a fairness assumption $\mathcal{F}$ gathering these various fairness constraints [BK08].

Definition 5.8 ($\mathcal{F}$)**.** A *fairness assumption* is a set $\mathcal{F}$ of fairness constraints.

Definition 5.9 ($\Pi|^{\mathcal{F}}$)**.** Given a fairness assumption $\mathcal{F}$, *maximal fair paths* are these that satisfy every fairness constraint in $\mathcal{F}$. The set of all maximal fair paths starting from a state $s \in \mathcal{S}$ is noted $\Pi|^{\mathcal{F}}(s) = \{\pi \in \Pi(s) \mid \forall f \in \mathcal{F} \quad \pi \models f\}$, the set of all maximal fair paths of $\mathcal{G}$ is noted $\Pi|^{\mathcal{F}}(\mathcal{G})$.

Definition 5.10 ($\mathcal{G}|^{\mathcal{F}}$)**.** Given a fairness assumption $\mathcal{F}$ and an STG $\mathcal{G}$, the *restriction of $\mathcal{G}$ by $\mathcal{F}$* is the STG $\mathcal{G}|^{\mathcal{F}} = (\mathcal{S}|^{\mathcal{F}}, \mathcal{A}, \rightarrow, \mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{A}})$, where:

$$\mathcal{S}|^{\mathcal{F}} = \{s \in \mathcal{S} \mid \Pi|^{\mathcal{F}}(s) \neq \emptyset\}$$

Proposition 5.2. $\Pi|^{\mathcal{F}}(\mathcal{G}) = \Pi|^{\mathcal{F}}(\mathcal{G}|^{\mathcal{F}}) \neq \Pi(\mathcal{G}|^{\mathcal{F}})$

Proof. $\Pi|^{\mathcal{F}}(\mathcal{G}) = \bigcup_{s \in \mathcal{S}} \Pi|^{\mathcal{F}}(s) = \bigcup_{s \in \mathcal{S} \mid \Pi|^{\mathcal{F}}(s) \neq \emptyset} \Pi|^{\mathcal{F}}(s) = \bigcup_{s \in \mathcal{S}|^{\mathcal{F}}} \Pi|^{\mathcal{F}}(s) = \Pi|^{\mathcal{F}}(\mathcal{G}|^{\mathcal{F}})$

But $\Pi(\mathcal{G}|^{\mathcal{F}}) = \bigcup_{s \in \mathcal{S}|^{\mathcal{F}}} \Pi(s) \neq \Pi|^{\mathcal{F}}(\mathcal{G})$ □

A fairness assumption $\mathcal{F}$ restricts directly the set of maximal path $\Pi|^{\mathcal{F}}(\mathcal{G})$, resulting in a restricted STG $\mathcal{G}|^{\mathcal{F}}$ whose states without fair paths have been removed. Conversely an α-restriction acts on the STG itself $\mathcal{G}|_\alpha$, resulting in a restriction of its set of maximal paths $\Pi|_\alpha(\mathcal{G})$. We now combine both restrictions to extend ARCTL with fairness assumptions.

Definition 5.11 ($\Pi|^{\mathcal{F}}_\alpha$)**.** Given a fairness assumption $\mathcal{F}$, an action formula α, and an STG $\mathcal{G}$, *α-restricted maximal fair paths* of $\mathcal{G}$ are the maximal paths of $\mathcal{G}|_\alpha$ that satisfy every fairness constraint in $\mathcal{F}$. The set of all α-restricted maximal fair paths starting from a state $s \in \mathcal{S}$ is noted $\Pi|^{\mathcal{F}}_\alpha(s) = \{\pi \in \Pi|_\alpha(s) \mid \forall f \in \mathcal{F} \quad \pi \models f\}$, the set of all α-restricted maximal fair paths of $\mathcal{G}$ is noted $\Pi|^{\mathcal{F}}_\alpha(\mathcal{G})$.

Definition 5.12 ($\mathcal{G}|^{\mathcal{F}}_\alpha$)**.** Given a fairness assumption $\mathcal{F}$, an action formula α, and an STG $\mathcal{G}$, the restriction of $\mathcal{G}$ by α and $\mathcal{F}$ is the STG $\mathcal{G}|^{\mathcal{F}}_\alpha = (\mathcal{S}|^{\mathcal{F}}_\alpha, \mathcal{A}, \rightarrow|_\alpha, \mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{A}})$, where:

$$\mathcal{S}|^{\mathcal{F}}_\alpha = \{s \in \mathcal{S} \mid \Pi|^{\mathcal{F}}_\alpha(s) \neq \emptyset\}$$

Technical remark

$\Pi|^{\mathcal{F}}_\alpha(s)$ cannot be defined by starting from $\Pi|^{\mathcal{F}}(s)$ because the α-restriction may produce new dead-ends, and thus add fair paths. Therefore, the α-restriction must be enforced first and, in a second step, the fairness constraints can be enforced on the maximal paths of the α-restricted model $\mathcal{G}|_\alpha$.

5.2.3 Fair ARCTL

In the literature [Cla+18a, chap. 2.2], the fairness assumption $\mathcal{F}$ is classically defined at the level of the STG $\mathcal{G}$. Here, we define the fairness assumption at the level of the quantifiers $(\exists, \forall)$ of an ARCTL formula, as it had already been proposed for CTL [Hau+20]. Thus the fairness assumption may change along an FARCTL formula. Of course, a fairness assumption defined at the level of the model can be applied recursively upon every quantifier, resulting in the same semantics.

The syntax and semantics of *Fair ARCTL (FARCTL)* extend these of ARCTL with fairness restrictions on quantifiers that now quantify over maximal α-restricted fair paths:

Definition 5.13 (Syntax)**.** The syntax of FARCTL is given by the following grammar over state, action and path formulas (with new fairness restrictions on quantifiers):

- state formulas: $\varphi \overset{\text{def}}{=} \top_{\mathcal{S}} \mid p_{\mathcal{S}} \in \mathcal{P}_{\mathcal{S}} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists^{\mathcal{F}}_{\alpha}\gamma \mid \forall^{\mathcal{F}}_{\alpha}\gamma$
- action formulas: $\alpha \overset{\text{def}}{=} \top_{\mathcal{A}} \mid p_{\mathcal{A}} \in \mathcal{P}_{\mathcal{A}} \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \mid \alpha_1 \vee \alpha_2$
- path formulas: $\gamma \overset{\text{def}}{=} \mathsf{X}\varphi \mid \varphi_1 \mathsf{U} \varphi_2 \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi$

An ARCTL formula is a state formula.

Definition 5.14 (Semantics)**.** The semantics of FARCTL is the same as the semantics of ARCTL without fairness, except for the quantifiers. Let $s \in \mathcal{S}|^{\mathcal{F}}_{\alpha}$ a state:

- $s \models \exists^{\mathcal{F}}_{\alpha}\gamma$ iff $\exists \pi \in \Pi|^{\mathcal{F}}_{\alpha}(s)$ such that $\pi \models \gamma$
- $s \models \forall^{\mathcal{F}}_{\alpha}\gamma$ iff $\forall \pi \in \Pi|^{\mathcal{F}}_{\alpha}(s)$ we have $\pi \models \gamma$

Example 5.6. In the STG of Figure 5.1, using the invariance pattern $\exists\mathsf{G}(\exists\mathsf{F}x)$, i.e. x *states possibly remain forever reachable*. We have $\mathcal{G} \models \exists\mathsf{G}(\exists\mathsf{F}\neg$  $)$, indeed the system can loop between *Grassland*, *Sparse scrubland* and *Open canopy woodland*, which all allow reaching a state without trees. But $\mathcal{G} \not\models \exists^{\mathcal{F}_S(}$  $^{)}\mathsf{G}(\exists\mathsf{F}\neg$$)$ because the fairness constraint $\mathcal{F}_S($$)$ forces the system to reach *Dense scrubland* or *Closed canopy woodland* from which there is no path removing trees.

5.3 Symbolic model-checking algorithm for FARCTL

In this section, we provide a symbolic algorithm computing the set of states satisfying an FARCTL formula. Similarly to the symbolic algorithms for CTL with fairness [McM93; Hau+20], this algorithm relies on the $\exists^{\mathcal{F}}_{\alpha}\mathsf{G}$ case. Indeed fairness relates to the infinite suffix of a path, and thus to the temporal modality G. Let us first define some functions over sets of states that will help us define the algorithm for $\exists^{\mathcal{F}}_{\alpha}\mathsf{G}$.

Definition 5.15. Let $e_{\mathcal{S}} \in ARCTL_{\mathcal{S}}$ a state-event, $e_{\mathcal{A}} \in ARCTL_{\mathcal{A}}$ an action-event, and $Z \subseteq \mathcal{S}$:

- $\exists_{\alpha}{}^{\vec{\mathsf{x}}}{}_{\models e_{\mathcal{S}}}(Z) = e_{\mathcal{S}} \wedge (\exists_{\alpha}\mathsf{X}Z \vee \neg\exists_{\alpha}\mathsf{X}\top)$
- $\exists_{\alpha}{}^{\vec{\mathsf{x}}}{}_{\not\models e_{\mathcal{S}}}(Z) = \neg e_{\mathcal{S}} \wedge (\exists_{\alpha}\mathsf{X}Z \vee \neg\exists_{\alpha}\mathsf{X}\top)$

- $\exists_\alpha \vec{\mathsf{X}}_{\models e_\mathcal{A}}(Z) = \exists_{\alpha \wedge e_\mathcal{A}} \mathsf{X} Z$
- $\exists_\alpha \vec{\mathsf{X}}_{\not\models e_\mathcal{A}}(Z) = \exists_{\alpha \wedge \neg e_\mathcal{A}} \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top$

Intuitively, $\exists_\alpha \vec{\mathsf{X}}_{\models e}(Z)$ is the set of states allowing to start either: (1) an α-path toward Z, or (2) a self-loop induced by the α-restriction; along which event e happens during either the first state or the first action, depending if e is a state-event or an action-event. Thus, $\exists_\alpha \vec{\mathsf{X}}_{\models e}(Z)$ implements for $\exists_\alpha \mathsf{X}$ the semantics of α-dead-ends as self-loops of $\bot_\mathcal{A}$ actions, hence the $\nrightarrow$ on top of the X. Similarly, $\exists_\alpha \vec{\mathsf{X}}_{\not\models e}(Z)$ is the set of states allowing to start either: (1) an α-path toward Z, or (2) a self-loop induced by the α-restriction; along which event e *does not* happen during either the first state or the first action, depending if e is a state-event or an action-event.

Proposition 5.3.

$$\exists_\alpha \vec{\mathsf{X}}_{\not\models e_\mathcal{S}}(Z) = \exists_\alpha \vec{\mathsf{X}}_{\models \neg e_\mathcal{S}}(Z)$$

But:

$$\exists_\alpha \vec{\mathsf{X}}_{\not\models e_\mathcal{A}}(Z) = \exists_\alpha \vec{\mathsf{X}}_{\models \neg e_\mathcal{A}}(Z) \vee \neg \exists_\alpha \mathsf{X} \top \neq \exists_\alpha \vec{\mathsf{X}}_{\models \neg e_\mathcal{A}}(Z)$$

Which matches the intuition because no action happens along self-loops: $\bot_\mathcal{A} \not\models \neg e_\mathcal{A}$.

We can now provide an algorithm to compute $\exists_\alpha^\mathcal{F} \mathsf{G}$ for each kind of fairness constraint.

5.3.1 Strong fairness

Definition 5.16 ($\tau_\alpha^{\mathcal{F}_S}$)**.** Let $\mathcal{F}_S = \{\mathcal{F}_S(e_1, e_2) \mid e_1 \in ARCTL_\mathcal{S}\}$ a fairness assumption composed exclusively of strong fairness constraints whose first events are state-events $e_1 \in ARCTL_\mathcal{S}$. We define $\tau_\alpha^{\mathcal{F}_S}$ as:

$$\tau_\alpha^{\mathcal{F}_S}(Z) = \bigwedge_{\mathcal{F}_S(e_1,e_2) \in \mathcal{F}_S} \left(\exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(Z) \vee \exists_\alpha (Z \mathsf{U} Z \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(Z)) \right)$$

Intuitively, $s \in \nu\, Z.\tau_\alpha^{\mathcal{F}_S}(Z)$ means that from s we can reach either an α-dead-end satisfying every fairness constraints, or a strongly connected component such that $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$ if event e_1 happens in the SCC then event e_2 happens as well. In such SCC, when event e_1 happens, we can always extend the path in the SCC to reach e_2, thus satisfying the strong fairness constraint.

Lemma 5.1. $\left(s \in \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))\right) \Rightarrow \left(s \models \exists_\alpha^{\mathcal{F}_S} \mathsf{G}(\varphi)\right)$

Proof. We give here the sketch of the proof, the fully detailed proof is given in Annex A.3.

Let $S' = \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))$. For every state $s \in S'$, we have $s \models \varphi$ and $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$ at least one of the following is true:

1. $s \models \exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(S')$, meaning that s is either (1) the beginning in S' of a maximal α-path, or (2) of a self-loop induced by the α-restriction; along which e_1 does not happen during the first state/transition.

2. $s \models \exists_\alpha(S' \mathsf{U} S' \wedge \exists_\alpha \vec{\mathrm{x}}_{\models e_2}(S'))$, meaning that s allows to α-reach either (1) the beginning in S' of a maximal α-path, or (2) a self-loop induced by the α-restriction; along which e_2 happens during the first state/transition.

Thus from any $s \in S'$, one can build a maximal α-path in S' satisfying continuously φ and either: (1) ending in a dead-end satisfying every strong fairness constraint, or (2) infinitely carrying on while $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, if e_1 happens then it is eventually followed by e_2, thus if e_1 happens infinitely often then e_2 happens infinitely often as well. Hence $s \models \exists_\alpha^{\mathcal{F}_S} \mathsf{G}(\varphi)$. □

Technical remark

Note that we restricted the first event of each strong fairness constraint to be a state-event: $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S \quad e_1 \in ARCTL_{\mathcal{S}}$. Indeed, we need to check that if e_1 happens, then it can be followed by e_2. Thus, if e_1 does not happen ($\exists_\alpha \vec{\mathrm{x}}_{\not\models e_1}(Z)$) we can carry on without checking for e_2. But, when several strong fairness constraints have action-events as e_1, then we must synchronize the action chosen in the several $\exists_\alpha \vec{\mathrm{x}}_{\not\models e_1}(Z)$. Indeed, we can carry on without checking for any e_2 iff we can choose an action where neither e_1 happens for every strong fairness constraint. Similarly, if only some e_1 happen during the chosen action, then we must check that their respective e_2 can be reached. Thus, $\tau_\alpha^{\mathcal{F}_S}$ does not produce the correct result when more than one strong fairness constraint have action-events as e_1.

We chose to restrict e_1 to state-events because we did not find applications where action-events were needed as first events of strong fairness constraints. But note that the lemma is still true if only a single strong fairness constraint has an action-event as e_1. Moreover, the algorithm could be adapted to several first action-events by checking, for every combination of these action-events, if an action can be taken where neither e_1 of the combination happens and that allows reaching every e_2 not in the combination. Instead of enumerating over fairness constraints, we would enumerate over combinations of fairness constraints, thus increasing greatly the complexity of the algorithm. Another adaptation of the algorithm could be to add the last action taken in the states themselves in order to synchronize the chosen action in the several $\exists_\alpha \vec{\mathrm{x}}_{\not\models e_1}(Z)$. Yet, this approach would increase the complexity of the algorithm as well by increasing the size of the state space.

The preceding lemma is one-way only, in fact the opposite direction is false in general. Indeed $\nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))$ requires that every occurrence of e_1 is able to α-reach an occurrence of e_2. But this requirement is too strong: e_1 is allowed to happen a finite number of times along a maximal α-restricted fair path without requiring the occurrence of e_2. To take this fact into account, we add the operator $\exists_\alpha(\ldots \mathsf{U} \ldots)$ to append the finite prefixes where e_1 happens only a finite number of times.

Theorem 5.1. $\exists_\alpha^{\mathcal{F}_S} \mathsf{G}(\varphi) = \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$

Proof. We give here the sketch of the proof, the fully detailed proof is given in Annex A.3.

Let us prove that $\exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}}(Z))) \subseteq \exists_\alpha^{\mathcal{F}}\mathsf{G}(\varphi)$. Let $s \in \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}}(Z)))$ and take $\pi \in \Pi|_\alpha(s)$ such that $\pi \models (\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}}(Z)))$. Thus $\exists i \in \mathbb{N}$ such that $\pi[i]_{\mathcal{S}} \models \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}}(Z))$ and $\forall j < i, \pi[j]_{\mathcal{S}} \models \varphi$. Now use Lemma 5.1 to extend π from $\pi[i]_{\mathcal{S}}$ into a maximal α-restricted fair path satisfying φ.

Let us prove that $\exists_\alpha^{\mathcal{F}_S}\mathsf{G}(\varphi) \subseteq \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$. Let $s \in \exists_\alpha^{\mathcal{F}_S}\mathsf{G}(\varphi)$ and take $\pi \in \Pi|_\alpha^{\mathcal{F}_S}(s)$ such that $\pi \models \mathsf{G}\varphi$. We will use the fact that $\nu\, Z.\tau(Z) \stackrel{\text{def}}{=} \cup\{S \subseteq \mathcal{S} \mid S \subseteq \tau(S)\}$ [BS07a] to show that $s \in \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$.

1. If $|\pi| \in \mathbb{N}$, then we prove that $s' = \pi[|\pi|]_{\mathcal{S}}$, the α-dead-end ending π, is in the greatest fixed point of $\varphi \wedge \tau_\alpha^{\mathcal{F}}$, i.e. that $\{s'\} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_S}(\{s'\})$. Indeed $s' \models \varphi$ because $\pi \models \mathsf{G}\varphi$. As π is fair, for every $\mathcal{F}_S(e_1, e_2) \in \mathcal{F}$, either e_2 happens along the self-loop, or e_1 does not happen, thus:

$$s' \models \bigwedge_{\mathcal{F}_S(e_1,e_2)\in\mathcal{F}_S} \left(\exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(\{s'\}) \vee \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(\{s'\})\right) \subseteq \tau_\alpha^{\mathcal{F}_S}(\{s'\})$$

   Consequently $s \in \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$.

2. If $|\pi| = \infty$ then take $\pi[i \ldots]$ the longest infinite suffix of π such that $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}$, if e_1 only happen a finite number of times along π, then i is strictly bigger than the last occurrence of e_1. Thus along $\pi[i \ldots]$, either e_2 happens infinitely often or e_1 does not happen at all. Let us prove that the states $\pi[i \ldots]_{\mathcal{S}}$ of this suffix are in the greatest fixed point of $\varphi \wedge \tau_\alpha^{\mathcal{F}_S}$, i.e. that $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_S}(\pi[i \ldots]_{\mathcal{S}})$. As $\pi \models \mathsf{G}\varphi$ we have $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi$. For every $\mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, either e_2 happens infinitely often along $\pi[i \ldots]$ or e_1 does not happen at all, thus $\forall j \geqslant i$:

$$\pi[j]_{\mathcal{S}} \models \bigwedge_{\mathcal{F}_S(e_1,e_2)\in\mathcal{F}_S} \Big(\exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(\pi[i \ldots]_{\mathcal{S}}) \vee \exists_\alpha(\pi[i \ldots]_{\mathcal{S}} \mathsf{U} \pi[i \ldots]_{\mathcal{S}} \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(\pi[i \ldots]_{\mathcal{S}}))\Big)$$

   Thus $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}}(\pi[i \ldots]_{\mathcal{S}})$, and consequently $s \in \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}}(Z)))$.

□

We gave here intuitive proofs of Lemma 5.1 and Theorem 5.1, exposing the structure of the proofs and their main arguments. Detailed proofs, based on the formal definitions of the fairness constraints and breaking down the cases depending on whether e_2 are state-events or action-events, are given in Annex A.3.

Example 5.7. If we restrict the semantics to CTL state-events and infinite maximal paths, we can recognize the fixed point algorithm for CTL strong fairness [Hau+20]:

$$\exists^{\mathcal{F}}\mathsf{G}\varphi = \exists\left(\varphi \mathsf{U} \nu\, Z.\varphi \wedge \bigwedge_{\mathcal{F}_S(e_1,e_2)\in\mathcal{F}} (\neg e_1 \wedge \exists\mathsf{X}Z) \vee \exists\mathsf{X}(\exists(Z \mathsf{U} Z \wedge e_2))\right)$$

Note the absence of $\neg\exists\mathsf{X}\top$ because of the restriction to infinite maximal paths, and the order inversion between $\exists\mathsf{X}$ and $\exists\mathsf{U}$ in the e_2 case in order to be consistent between state-events and action-events. Indeed swapping $\exists\mathsf{X}$ and $\exists\mathsf{U}$ is irrelevant along infinite

looping ν because we only need to check that we move forward in Z at some point, i.e. $\exists \mathsf{X} Z$, but checking action-events requires to nest $\exists \mathsf{X}$ inside $\exists \mathsf{U}$.

5.3.2 Unconditional fairness

The algorithm to compute $\exists_\alpha^{\mathcal{F}_U} \mathsf{G}$ for the unconditional fairness can be derived from the algorithm of the strong fairness. Indeed unconditional fairness can be considered as a special case of strong fairness.

Lemma 5.2. For all maximal paths π, $\pi \models^{\infty}_{\exists} \top_{\mathcal{S}}$.

Proof. Recall that $\pi \models^{\infty}_{\exists} \top_{\mathcal{S}}$ iff $\forall i \in \mathbb{N}, i \leqslant |\pi| \Rightarrow \exists j \in \mathbb{N}$ such that $i \leqslant j \leqslant |\pi|$ and $\pi[j]_{\mathcal{S}} \models \top_{\mathcal{S}}$. For all maximal path π and for all $i \leqslant |\pi|$ take $j = i$, we have $i \leqslant j, j \leqslant |\pi|$ and $\pi[i]_{\mathcal{S}} \models \top_{\mathcal{S}}$. Thus for all maximal paths π, $\pi \models^{\infty}_{\exists} \top_{\mathcal{S}}$. □

Corollary 5.1. Let e be a state-event or an action-event, $\mathcal{F}_U(e) = \mathcal{F}_S(\top_{\mathcal{S}}, e)$.

Proof.

$$\begin{aligned}
\mathcal{F}_S(\top_{\mathcal{S}}, e) \text{ iff } & \left(\pi \models^{\infty}_{\exists} \top_{\mathcal{S}}\right) \Rightarrow \left(\pi \models^{\infty}_{\exists} e\right) \\
\text{iff } & (\top) \Rightarrow \left(\pi \models^{\infty}_{\exists} e\right) \\
\text{iff } & \left(\pi \models^{\infty}_{\exists} e\right) \\
\text{iff } & \mathcal{F}_U(e)
\end{aligned}$$

□

Corollary 5.2. Let $\mathcal{F}_U$ a fairness assumption composed exclusively of unconditional fairness constraints, we define $\tau_\alpha^{\mathcal{F}_U}$ as:

$$\tau_\alpha^{\mathcal{F}_U}(Z) = \bigwedge_{\mathcal{F}_W(e) \in \mathcal{F}_W} \left(\exists_\alpha (Z \mathsf{U} Z \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e}(Z))\right)$$

Then: $\exists_\alpha^{\mathcal{F}_U} \mathsf{G}(\varphi) = \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_U}(Z))$

Proof. $\exists_\alpha \vec{\mathsf{X}}_{\not\models \top_{\mathcal{S}}}(Z) = \neg \top_{\mathcal{S}} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top) = \bot_{\mathcal{S}}$

Moreover nesting $\nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_U}(Z))$ inside $\exists_\alpha(\varphi \mathsf{U} \ldots)$ is not required. Indeed $\top_{\mathcal{S}}$ cannot happen only finitely often along a maximal α-path, as it happens in every state. □

Example 5.8. If we restrict the semantics to CTL state-events and infinite maximal paths, then we recognize the fixed point algorithm for CTL unconditional fairness [McM93] can be recognized:

$$\exists^{\mathcal{F}}\mathsf{G}\varphi = \nu Z.\varphi \wedge \exists\mathsf{X} \bigwedge_{\mathcal{F}_U(e)\in\mathcal{F}} \exists(Z\mathsf{U}Z \wedge e)$$

As for the strong fairness case, note the absence of $\neg\exists\mathsf{X}\top$ because of the restriction to infinite maximal paths, and the order inversion between $\exists\mathsf{X}$ and $\exists\mathsf{U}$.

5.3.3 Weak fairness

The algorithm to compute $\exists_\alpha^{\mathcal{F}_W}\mathsf{G}$ for the weak fairness is inspired from the algorithm of the unconditional fairness.

Lemma 5.3. Let π a maximal path, $e_{\mathcal{S}} \in ARCTL_{\mathcal{S}}$ a state-event, and $e_{\mathcal{A}} \in ARCTL_{\mathcal{A}}$ an action-event:

$$\pi \not\models \overset{\infty}{\forall}\, e_{\mathcal{S}} \text{ iff } \pi \models \overset{\infty}{\exists}\, \neg e_{\mathcal{S}}$$
$$\pi \not\models \overset{\infty}{\forall}\, e_{\mathcal{A}} \text{ iff } |\pi| \neq \infty \text{ or } \pi \models \overset{\infty}{\exists}\, \neg e_{\mathcal{A}}$$

Proof.

$$\pi \not\models \overset{\infty}{\forall}\, e_{\mathcal{S}} \text{ iff } \forall i \in \mathbb{N}, i \leqslant |\pi| \Rightarrow \exists j \geqslant i \text{ such that } \pi[j]_{\mathcal{S}} \not\models e_{\mathcal{S}}$$
$$\text{iff } \pi \models \overset{\infty}{\exists}\, \neg e_{\mathcal{S}}$$

$$\pi \not\models \overset{\infty}{\forall}\, e_{\mathcal{A}} \text{ iff } |\pi| \neq \infty \text{ or } \forall i \in \mathbb{N}, \exists j \geqslant i \text{ such that } \pi[j]_{\mathcal{A}} \not\models e_{\mathcal{A}}$$
$$\text{iff } |\pi| \neq \infty \text{ or } \pi \models \overset{\infty}{\exists}\, \neg e_{\mathcal{A}}$$

□

Corollary 5.3. Let π a maximal path, and e_2 a state-event or an action-event:

- Let e_1 a state-event, $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(\pi \models \overset{\infty}{\exists}\, \neg e_1 \text{ or } \pi \models \overset{\infty}{\exists}\, e_2\right)$
- Let e_1 an action-event, $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(|\pi| \neq \infty \text{ or } \pi \models \overset{\infty}{\exists}\, \neg e_1 \text{ or } \pi \models \overset{\infty}{\exists}\, e_2\right)$

Proof.

$$\pi \models \mathcal{F}_W(e_1, e_2) \text{ iff } \left(\pi \models \overset{\infty}{\forall}\, e_1\right) \Rightarrow \left(\pi \models \overset{\infty}{\exists}\, e_2\right)$$
$$\text{iff } \neg\left(\pi \models \overset{\infty}{\forall}\, e_1\right) \vee \left(\pi \models \overset{\infty}{\exists}\, e_2\right)$$
$$\text{iff } \left(\pi \not\models \overset{\infty}{\forall}\, e_1\right) \vee \left(\pi \models \overset{\infty}{\exists}\, e_2\right)$$

□

Example 5.9. If we restrict the semantics to state-events and infinite maximal paths, then a weak fairness constraint can be expressed as an LTL formula [BK08, chap. 6.5] and reformulated into an unconditional fairness constraint:

$$\mathsf{FG}e_1 \Rightarrow \mathsf{GF}e_2 = (\neg\mathsf{FG}e_1) \vee (\mathsf{GF}e_2) = (\mathsf{GF}\neg e_1) \vee (\mathsf{GF}e_2) = \mathsf{GF}(\neg e_1 \vee e_2)$$

The algorithm for the weak fairness is based upon this resemblance with unconditional fairness:

Theorem 5.2. Let $\mathcal{F}_W$ a fairness assumption composed exclusively of weak fairness constraints, we define $\tau_\alpha^{\mathcal{F}_W}$ as:

$$\tau_\alpha^{\mathcal{F}_W}(Z) = \bigwedge_{\mathcal{F}_W(e_1,e_2)\in\mathcal{F}} \left(\exists_\alpha(Z\mathsf{U}Z \wedge (\exists_\alpha \vec{\mathsf{x}}_{\not\models e_1}(Z) \vee \exists_\alpha \vec{\mathsf{x}}_{\models e_2}(Z)))\right)$$

Then: $\exists_\alpha^{\mathcal{F}_W}\mathsf{G}(\varphi) = \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$

A detailed proof is given in Annex A.3.

5.3.4 Complete symbolic algorithm for FARCTL model-checking

The complete algorithm for the symbolic model-checking of FARCTL relies on the algorithm for $\exists_\alpha^{\mathcal{F}}\mathsf{G}$ that merges the three algorithms defined previously.

Theorem 5.3. Let $\mathcal{F}$ a fairness assumption composed of a set of strong fairness constraints $\mathcal{F}_S$, a set of unconditional fairness constraints $\mathcal{F}_U$ and, a set of weak fairness constraints $\mathcal{F}_W$:

$$\exists_\alpha^{\mathcal{F}}\mathsf{G}(\varphi) = \exists_\alpha(\varphi\mathsf{U}\nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S\subseteq\mathcal{F}}(Z) \wedge \tau_\alpha^{\mathcal{F}_U\subseteq\mathcal{F}}(Z) \wedge \tau_\alpha^{\mathcal{F}_W\subseteq\mathcal{F}}(Z)))$$

Proof. Maximal α-restricted fair paths are stable by finite prefix concatenation, thus even if $\exists(\varphi\mathsf{U}\ldots)$ is not required for unconditional and weak fairness, it has the correct semantics. □

$\exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}}$ selects the states with at least one fair maximal path, i.e. $\exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}} = \{s \in \mathcal{S} \mid \Pi|_\alpha^{\mathcal{F}}(s) \neq \emptyset\} = \mathcal{S}|_\alpha^{\mathcal{F}}$. Similarly to CTL with fairness [McM93], the FARCTL operators dealing with prefixes can be rewritten using $\exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}}$:

- $\exists_\alpha^{\mathcal{F}}\mathsf{X}\varphi = \exists_\alpha\mathsf{X}(\varphi \wedge \exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}})$
- $\forall_\alpha^{\mathcal{F}}\mathsf{X}\varphi = \exists_\alpha\mathsf{X}\top_{\mathcal{S}} \wedge \neg\exists_\alpha\mathsf{X}\neg\varphi \wedge \exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}}$
- $\exists_\alpha^{\mathcal{F}}\mathsf{F}\varphi = \exists_\alpha\mathsf{F}(\varphi \wedge \exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}})$
- $\exists_\alpha^{\mathcal{F}}\varphi_1\mathsf{U}\varphi_2 = \exists_\alpha(\varphi_1\mathsf{U}(\varphi_2 \wedge \exists_\alpha^{\mathcal{F}}\mathsf{G}\top_{\mathcal{S}}))$

Technical remark

Under an action restriction α, every state $s \in \mathcal{S}$ has at least one maximal path (the path $\pi = s \overset{\alpha}{\not\to}$ of length 0 if $s \overset{\alpha}{\not\to}$). But under a fairness restriction $\mathcal{F}$, some states may not have any fair path at all, i.e. $s \notin \exists^{\mathcal{F}}_{\alpha} \mathsf{G} \top_{\mathcal{S}}$, thus $\Pi|^{\mathcal{F}}_{\alpha}(s) = \emptyset$. Removing such a state does not change the semantics, as it cannot be part of any maximal α-restricted fair path π (otherwise the suffix of π starting from s would be a maximal α-restricted fair path and thus $\Pi|^{\mathcal{F}}_{\alpha}(s) \neq \emptyset$). Note that this is exactly how we defined $\mathcal{S}|^{\mathcal{F}}_{\alpha} = \{s \in \mathcal{S} \mid \Pi|^{\mathcal{F}}_{\alpha}(s) \neq \emptyset\}$.

Once we restrict $\mathcal{S}$ to $\mathcal{S}|^{\mathcal{F}}_{\alpha} = \exists^{\mathcal{F}}_{\alpha} \mathsf{G} \top_{\mathcal{S}}$, the FARCTL operators dealing with prefixes are exactly the same as for ARCTL:

- $\exists^{\mathcal{F}}_{\alpha} \mathsf{X} \varphi = \exists_{\alpha} \mathsf{X} \varphi$
- $\forall^{\mathcal{F}}_{\alpha} \mathsf{X} \varphi = \forall_{\alpha} \mathsf{X} \varphi$
- $\exists^{\mathcal{F}}_{\alpha} \mathsf{F} \varphi = \exists_{\alpha} \mathsf{F} \varphi$
- $\exists^{\mathcal{F}}_{\alpha} \varphi_1 \mathsf{U} \varphi_2 = \exists_{\alpha} \varphi_1 \mathsf{U} \varphi_2$

The FARCTL operators dealing with suffixes are derived using the usual dual equivalences:

- $\forall^{\mathcal{F}}_{\alpha} \mathsf{F} \varphi = \neg \exists^{\mathcal{F}}_{\alpha} \mathsf{G} \neg \varphi$
- $\forall^{\mathcal{F}}_{\alpha} \mathsf{G} \varphi = \neg \exists^{\mathcal{F}}_{\alpha} \mathsf{F} \neg \varphi$
- $\forall^{\mathcal{F}}_{\alpha} \varphi_1 \mathsf{U} \varphi_2 = \neg (\exists^{\mathcal{F}}_{\alpha} \neg \varphi_2 \mathsf{U} \neg \varphi_1 \wedge \neg \varphi_2) \wedge \neg \exists^{\mathcal{F}}_{\alpha} \mathsf{G} \neg \varphi_2$

This symbolic FARCTL model-checking algorithm is implemented inside `ecco`.

6 Applications of FARCTL

Summary

In this chapter, we apply FARCTL to the two case studies presented in Chapter 4. First, we exemplify on the Borana model how ecosystem management scenarios can be modelled by action and fairness restrictions. FARCTL is used to investigate the consequences of shifting between scenarios during the model exploration, and to describe realistic paths using fairness constraints. Second, we exemplify on the protists model how two distinct kinds of dynamics can be combined in a controlled manner using FARCTL. We use action-restrictions to separate invasions from the rest of the dynamics, and alternate between them using an ARCTL formula to look for specific invasion behaviours.

The ecco notebooks encompassing the case studies presented in this chapter can be found in [Tho22]. Both the RR model file and the python analysis notebook are given for each case study, along with a static html preview of the notebook. Computing all the results took only a few seconds on a modern laptop (Linux 5.4 Mint/Ubuntu, 32G RAM, CPU Intel Core i7–7820HQ 2.9GHz).

6.1 Borana vegetation model

Chapter 4 presented the *Borana model* representing the vegetation dynamics of the Borana Zone in Ethiopia [LC18; LCD18; Lia+20]. CTL model-checking was used both to build a component graph representing the differences of bush encroachment behaviours depending upon fixed management scenarios, and to extract the scenarios making encroachment reversible. In this chapter, we explore with FARCTL how shifts between management scenarios impact the system dynamics. First, the *Borana model* is revised with action labels, then the question "*Can a change in management policy reverse the bush encroachment induced by the preceding one?*" is answered using FARCTL, and finally fairness is embedded into scenarios to get a more fitting model.

6.1.1 Modelling

The *Borana model* is revised to make use of FARCTL, see Figure 6.1. Controls, apart from altitude, are removed from the variables declaration and converted into action labels, i.e. controls are moved from the condition of the rules towards their label. Altitude remains a control, as it cannot be reasonably changed during the dynamics (we assume that the

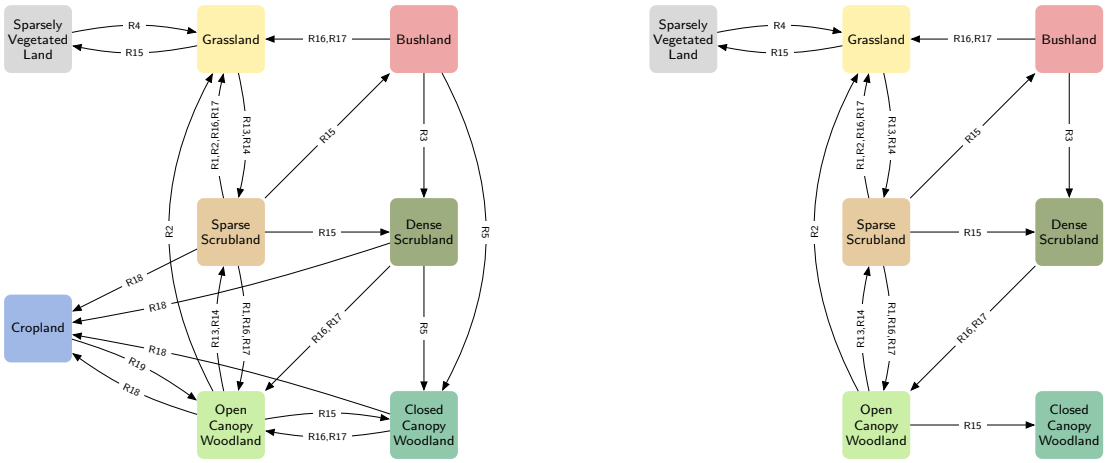
altitude of the studied ecosystem cannot shift, but note that global warming could produce a similar effect). Instead of being an initial value of the control variables, a scenario is now an action restriction. For example, the scenarios where fire is banned are no longer defined as the subset of the STG where `Fb+`, but as action-restrictions of the STG where the actions labelled by `[Fb-]` are disabled. Thus there are no longer distinct and disconnected STGs for distinct scenarios, but two disconnected STGs, corresponding to the system at low or at high altitude, with distinct action-restrictions corresponding to distinct scenarios.

```
variables:
    Gr+: Grasses
    Sh-: Shrubs
    Tr-: Savanna trees
    Sa-: Tree saplings
    Cr-: Crops
    Lv-: Livestock (cattle)
    Gz-: Wild grazers
    Bw-: Wild browsers

controls:
    Alt*: Altitude

rules:
  [Fb-] Gr+ >> Sh-, Sa-, Lv-, Gz-, Bw-                  # R1
  [Fb-] Gr+ >> Sh-, Tr-, Sa-, Lv-, Gz-, Bw-             # R2
  Sa+ >> Tr+                                             # R3
  Sh-, Tr-, Sa-, Cr- >> Gr+                              # R4
  [Fb+] Alt+, Gr-, Sa+ >> Sh-, Tr+                       # R5
  Alt-, Sh+, Tr- >> Sa-                                  # R6
  [Wl+] Gr+, Lv- >> Gz+                                  # R7
  [Wl+] Sh+, Lv- >> Bw+                                  # R8
  [Wl+] Sa+, Lv- >> Bw+                                  # R9
  [Ps+] Gr+ >> Lv+, Gz-, Bw-                             # R10
  [Ps+, BLv+] Sh+ >> Lv+, Gz-, Bw-                       # R11
  [Ps+, BLv+] Sa+ >> Lv+, Gz-, Bw-                       # R12
  Gr+, Lv+ >> Sh+, Sa+                                   # R13
  Gr+, Gz+ >> Sh+, Sa+                                   # R14
  [Ig+] Lv+ >> Gr-, Lv-                                  # R15
  Bw+ >> Gr+, Sh-, Sa-, Bw-                              # R16
  [BLv+] Lv+ >> Gr+, Sh-, Sa-, Bw-                       # R17
  [Cb-] Alt+, Tr+ >> Gr-, Sh-, Sa-, Cr+, Lv-, Gz-, Bw-   # R18
  Cr+ >> Gr+, Cr-                                        # R19
```

Figure 6.1: Revised *Borana model*.

The STG computed from the revised *Borana model* has only 50 states, way less than the 1185 states spread over the 128 disconnected subgraphs of its initial version. The component graphs corresponding to the partition between vegetation classes of the STGs at low and at high altitudes are depicted in Figure 6.2. Note that some rules should not be enabled together in the same scenario, for example `R1` and `R5` have conflicting labels:

[Fb−] and [Fb+]. This fact has to be ensured at the level of the action-restrictions defining the scenarios, thus some paths in the unrestricted STG may involve conflicting rules that implicitly involve a scenario shift in between their occurrences.



(a) At high altitude Alt+ **(b)** At low altitude Alt−

Figure 6.2: Component graphs of the revised *Borana model* partitioned by vegetation classes.

Technical remark

Note that we cannot design an action-restriction defining a set of scenarios. Indeed, we could build an action-restriction α as the logical or ($\vee$) of the distinct action-restrictions corresponding to each element of a set of scenarios. Nevertheless, the paths resulting from α are not the union of the paths resulting from each action-restriction, but the paths shifting arbitrarily often between these action-restrictions. Thus when doing $\exists_\alpha$ we are not exploring the maximal paths of the various scenarios of the set, but the maximal paths shifting arbitrarily often between these scenarios. Although such action-restriction α can be useful, there is no way to explore the maximal paths of a set of scenarios without either explicitly building distinct action-restrictions linked to distinct quantifiers, or representing the scenarios as control variables.

6.1.2 Results

The benefits of analysing the *Borana model* with FARCTL are outlined using the state property bush encroachment = $(\texttt{Sh+}\vee\texttt{Tr+})\wedge\texttt{Gr−}\wedge\texttt{Cr−}$, and the following action-restrictions corresponding to the 3 STMs of Liao, see Figure 4.2:

- WildScenario $= \neg([\texttt{Fb+}]\vee[\texttt{Cb−}]\vee[\texttt{Wl−}]\vee[\texttt{Ps+}]\vee[\texttt{Ig+}]\vee[\texttt{BLv+}])$ corresponding to the wild scenario of the Borana zone: wildlife with fire but without livestock nor crop, see Figure 4.2a (Alt∗, Fb−, Cb+, Wl+, Ps−, Ig−, BLv− in the original *Borana model*);
- TraditionalScenario $= \neg([\texttt{Fb+}]\vee[\texttt{Cb−}]\vee[\texttt{Wl+}]\vee[\texttt{Ps−}]\vee[\texttt{Ig+}]\vee[\texttt{BLv+}])$ corresponding to the traditional management policy of the Borana zone: extensive grazing with fire but without wildlife nor crop, see Figure 4.2b (Alt∗, Fb−, Cb+, Wl−, Ps+, Ig−, BLv− in the original *Borana model*);

- $\texttt{CurrentScenario} = \neg(\texttt{[Fb-]} \vee \texttt{[Cb-]} \vee \texttt{[Wl+]} \vee \texttt{[Ps-]} \vee \texttt{[Ig-]} \vee \texttt{[BLv+]})$ corresponding to the current management policy of the Borana zone: intensive grazing without fire nor wildlife nor crops, see Figure 4.2c (`Alt*`, `Fb+`, `Cb+`, `Wl-`, `Ps+`, `Ig+`, `BLv-` in the original *Borana model*).

First, we can retrieve the fact, established in Chapter 4, that intensive grazing is required for bush encroachment. To do so, we check that bush encroachment can only happen in the `CurrentScenario`:

- $\mathcal{S}_0 \not\models \exists_{\texttt{WildScenario}} \mathsf{F}(\text{🌳})$
- $\mathcal{S}_0 \not\models \exists_{\texttt{TraditionalScenario}} \mathsf{F}(\text{🌳})$
- $\mathcal{S}_0 \models \exists_{\texttt{CurrentScenario}} \mathsf{F}(\text{🌳})$

The only satisfied formula is indeed the one restricting the actions to the `CurrentScenario`. Next, we show how to provide an answer to the following question: "*Can a change in management policy reverse the bush encroachment induced by the preceding one?*" To do so, we adapt to FARCTL the sequence and invariance patterns used in Chapter 4. Let us check if at least some bush encroachment is reversible by a change in management policy:

- $\mathcal{S}_0 \models \exists_{\texttt{CurrentScenario}} \mathsf{F}(\text{🌳} \wedge \exists_{\texttt{WildScenario}} \mathsf{F}(\neg\text{🌳}))$
- $\mathcal{S}_0 \not\models \exists_{\texttt{CurrentScenario}} \mathsf{F}(\text{🌳} \wedge \exists_{\texttt{TraditionalScenario}} \mathsf{F}(\neg\text{🌳}))$

Thus the bush encroachment induced in the Borana zone by the current management policy cannot be reversed by shifting back toward the traditional management policy. However, at least some bush encroachment is reversible by letting the ecosystem shift back to its wild behaviour. Let us check if all bush encroachment is reversible by a change in management policy:

- $\mathcal{S}_0 \models \forall_{\texttt{CurrentScenario}} \mathsf{G}(\text{🌳} \Rightarrow \exists_{\texttt{WildScenario}} \mathsf{F}(\neg\text{🌳}))$
- $\mathcal{S}_0 \not\models \forall_{\texttt{CurrentScenario}} \mathsf{G}(\text{🌳} \Rightarrow \exists_{\texttt{TraditionalScenario}} \mathsf{F}(\neg\text{🌳}))$

The traditional management policy cannot reverse all bush encroachment induced by the current management policy, as expected because we just saw that it could not even reverse it partially. However, all bush encroachment induced by the current management policy can be reversed by letting the ecosystem shift back to its wild behaviour. The fact that the `TraditionalScenario` prevents encroachment but cannot reverse it (at least when induced by the `CurrentScenario`) in the *Borana model* may seem counter-intuitive. Yet, such behaviour has been reported before [LCD18; Lia+20] and was one of the motivations behind the STM framework [WWN89].

We could test other scenarios, both inducing encroachment or trying to reverse it, but we limit ourselves here to the scenarios of Liao's STMs in order to validate the model. Note that if we change the scenario inducing encroachment, then the reachable encroached state may change as well, thus encroachment may not be reversible by the `WildScenario` anymore. In addition, we could design more complex formulas, shifting between more than two scenarios for example.

Finally, we outline the use of fairness assumptions in FARCTL formulas to characterize realistic paths using the following action fairness constraint:

- $\mathcal{F}_W(\texttt{[Ig+]}) = \mathcal{F}_W(\exists_{\texttt{[Ig+]}} \mathsf{X}\top, \texttt{[Ig+]})$ meaning that if rules related to intensive grazing are enabled continuously, then they must happen infinitely often

- $\mathcal{F}_S(\texttt{[Fb-]}) = \mathcal{F}_S(\exists_{\texttt{[Fb-]}}\mathsf{X}\top, \texttt{[Fb-]})$ meaning that if rules related to fire are enabled infinitely often, then they must happen infinitely often

In the `CurrentScenario`, grasses do not always disappear: $\mathcal{S}_0 \not\models \forall_{\texttt{CurrentScenario}}\mathsf{F}(\texttt{Gr-})$. We can wonder what would happen if we force the intensive grazing rule `R15` to happen if it is continuously enabled, i.e. if we assume that the continuous presence of livestock under the intensive grazing policy shall trigger the intensive grazing rule infinitely often. Indeed, the continuous presence of livestock without consequence is not realistic. To do so, we use the fairness constraint $\mathcal{F}_W(\texttt{[Ig+]})$ and get a new model-checking result: $\mathcal{S}_0 \models \forall^{\mathcal{F}_W(\texttt{[Ig+]})}_{\texttt{CurrentScenario}}\mathsf{F}(\texttt{Gr-})$. Thus grasses always disappear in the `CurrentScenario` with weak fairness on the intensive grazing rule `R15`, indeed `R15` is the only rule removing `Gr` enabled in the `CurrentScenario`.

We got a similar result in the `WildScenario`, where shrubs and saplings do not always disappear: $\mathcal{S}_0 \not\models \forall_{\texttt{WildScenario}}\mathsf{F}(\texttt{Sh-} \wedge \texttt{Sa-})$. We can wonder what would happen if we force the fire rules `R1` and `R2` to happen if they are enabled infinitely often, i.e. if we assume that if herbaceous fuel is present infinitely often then the fire rules shall trigger infinitely often as well. To do so, we use the fairness constraint $\mathcal{F}_S(\texttt{[Fb-]})$ and get a new model-checking result: $\mathcal{S}_0 \models \forall^{\mathcal{F}_S(\texttt{[Fb-]})}_{\texttt{WildScenario}}\mathsf{F}(\texttt{Sh-} \wedge \texttt{Sa-})$. Thus shrubs and saplings always disappear in the `WildScenario` with strong fairness on the fire rules `R1` and `R2`.

6.2 Protists model

Chapter 4 presented the protists model representing the first protists experiment [WWL98] studying the fate of species combinations. A component graph was built representing the various collapsing behaviours leading to stable communities. In this chapter, we show how to model with FARCTL the second protists experiment [WLW03] studying the invasion of stable communities. First, the protists model is revised with invasion rules and action labels, then the question "*Are there catalytic species, i.e. that invade, change the community, and then go extinct?*" is answered using FARCTL.

6.2.1 Modelling

The protists model is revised to make use of ARCTL, see Figure 6.3. While the variables section remains unchanged, 6 new rules were added modelling the invasion by each species. Rules are labelled by their related process `[process]` as in the original protists model, and in addition by their affecting species `[sp:l]` (appearing positively on the left-hand side of the rule) as well as by their affected species `[sp:r]` (appearing on the right-hand side of the rule). Thus an `[invasion]` rule is also labelled by the invading species `[sp:r]`, and the other rules are also labelled by both the acting species `[sp:l]` and the disappearing species `[sp:r]`.

The STG computed from the revised Protists model has 64 states, just like its initial version. But it has additional transitions representing the invasions that blur the dynamics. We will restrict the occurrences of these additional transitions using action-restrictions.

```
variables:
    A*: Amoeba proteus
    B*: Blepharisma japonicum
    C*: Colpidium striatum
    E*: Euplotes patella
    P*: Paramecium caudatum
    T*: Tetrahymena pyriformis

rules:
  [predation, A:l, P:r] A+ >> P-      # R1
  [predation, B:l, C:r] B+ >> C-      # R2
  [predation, B:l, T:r] B+ >> T-      # R3
  [predation, A:l, B:r] A+, P- >> B-  # R4
  [predation, A:l, C:r] A+, P- >> C-  # R5
  [predation, A:l, T:r] A+, P- >> T-  # R6
  [predation, A:l, E:r] A+, P- >> E-  # R7
  [predation, E:l, T:r] E+ >> T-      # R8
  [starvation, A:r] P- >> A-          # R9
  [starvation, E:l, E:r] E+ >> E-     # R10
  [starvation, C:l, C:r] C+ >> C-     # R11
  [competition, P:l, C:r] P+ >> C-    # R12
  [competition, P:l, T:r] P+ >> T-    # R13
  [competition, C:l, T:r] C+ >> T-    # R14
  [competition, T:l, C:r] T+ >> C-    # R15
  [invasion, A:r] A- >> A+            # R16
  [invasion, B:r] B- >> B+            # R17
  [invasion, C:r] C- >> C+            # R18
  [invasion, E:r] E- >> E+            # R19
  [invasion, P:r] P- >> P+            # R20
  [invasion, T:r] T- >> T+            # R21
```

Figure 6.3: Revised protists model.

6.2.2 Results

In this secction, we answer the following question taken from the second protists experiment article [WLW03]: *"Are there catalytic species, i.e. that invade, change the community, and then go extinct?"*. To do so, we design an FARCTL formula checking if species $\texttt{sp} \in \{\texttt{A},\texttt{B},\texttt{C},\texttt{E},\texttt{P},\texttt{T}\}$ is catalytic (with `[inv.]` for `[invasion]`):

$$\underbrace{\exists_{\texttt{[inv.]}\wedge\texttt{[sp:r]}}}_{\texttt{sp invades}}\mathsf{X}(\underbrace{\exists_{\neg\texttt{[inv.]}}}_{\substack{\texttt{dynamics}\\\texttt{goes on}}}\mathsf{F}(\underbrace{\exists_{\neg\texttt{[inv.]}\wedge\texttt{[sp:l]}\wedge\neg\texttt{[sp:r]}}}_{\texttt{sp change the community}}\mathsf{X}(\underbrace{\exists_{\neg\texttt{[inv.]}}}_{\substack{\texttt{dynamics}\\\texttt{goes on}}}\mathsf{F}(\underbrace{\exists_{\neg\texttt{[inv.]}\wedge\texttt{[sp:r]}}}_{\texttt{sp goes extinct}}\mathsf{X}\top))))$$

For each species $\texttt{sp} \in \{\texttt{A},\texttt{B},\texttt{C},\texttt{E},\texttt{P},\texttt{T}\}$, this formula is satisfied by some states of the STG. Thus every species is classified as catalytic. In [WLW03], only A, B, E and T were found catalytic experimentally. But in the second experiment, invasion was only tested in so-called "persistent communities". When checking the satisfaction of the formula only on these persistent communities, we find that P is not catalytic anymore, which is more in

line with the experimental result. The additional catalytic species found by the model may arise from the additional transitions predicted by the model, see Figure 4.17, or from the fact that invasions were not always successful in the experiment [LWW00] while the model assumes that every species can always invade.

7 Conclusion and discussion

7.1 Conclusion

In this thesis, we presented formal methods based on model-checking for the modelling and the analysis of ecological state-transition graphs. First, we introduced the concept of ecological *state-transition graph (STG)* that, while being a novelty, captures a long history of disparate representations of the dynamics of an ecosystem as a graph. Computer science has developed a wide range of methods for analysing such graphs, some of which have been used in systems biology. In this thesis, we focused on methods based on *model-checking*, which checks whether an STG satisfies a dynamical property given as a *temporal logic formula*. We presented the *Computation Tree Logic (CTL)* that expresses properties about branching dynamics with alternative pathways, a feature often found in ecological STGs. CTL model-checking can be used to partition the state space of the STG depending on the satisfaction of a given set of CTL formulas. Such partition can be depicted as a graph, called a *component graph*, that offers a visual representation of whether the STG dynamics fulfills or not the behaviour described by the given formulas.

This approach was implemented in `ecco` [PTG22a], a Python toolbox for the formal modelling and analysis of ecosystems. Models in `ecco` compute into STGs, whose states are defined by Boolean variables and whose transitions are derived from if-then rules. The model analysis in `ecco` is performed by interactively refining a partition of the state space, using either topological or CTL state properties, that is depicted as a component graph. We used `ecco` on two case studies exemplifying this approach. In the first instance, we modelled the vegetation changes in the Borana Zone in Ethiopia under diverse management scenarios [Tho+22]. We then built a component graph representing bush encroachment using CTL formulas, and selected management policies preventing it or making it reversible. In the second instance, we modelled protists community assembly based on the results of laboratory experiments [Her+22]. We then built a component graph partitioning the states with respect to the stable communities they lead to using topological properties.

Both case studies were limited by the fact that we want some specific events to occur but only in a controlled manner, for example changes of management scenarios or species invasion. To overcome these limitations, we presented *Action-Restricted CTL (ARCTL)* [PR07], an extension of CTL that allows to restrict the set of enabled actions along a formula. We then extended ARCTL with *fairness constraints*, i.e. "realism" constraints upon the order and happening rate of events along a path. This resulted in *Fair ARCTL (FARCTL)*, for which we provided a symbolic model-checking algorithm that is implemented inside `ecco`. FARCTL quantifiers allow to restrict maximal paths with respect to both enabled actions and fairness constraints, for example to model ecosystem management scenarios. An

FARCTL formula can thus shift between them in a controlled manner at the level of its quantifiers. Finally, we applied FARCTL to both case studies. In the first instance, we defined management scenarios as action and fairness restrictions, and we used FARCTL to investigate the consequences of shifting between scenarios. In the second instance, we used action-restrictions to separate invasions from the rest of the dynamics, and alternated between them using a FARCTL formula to look for specific invasion behaviours.

7.2 Implementation choices

The approach presented in this thesis is primarily independent of the presented implementation based on `ecco`. Indeed, as long as the ecosystem dynamics can be represented as an STG, the methods we presented, such as component graphs or FARCTL, can be applied. In this section, we discuss the implementation choices made in this thesis, namely Reaction Rules for modelling and (FAR)CTL for analysis, and give some perspective on envisioned improvements. Furthermore, we also list some other possible implementations, along with example studies in biology operating them.

7.2.1 Reaction Rules as modelling language

In this thesis, we computed STGs from Reaction Rules (RR) models [GP19; PTG22a], i.e. system descriptions based upon Boolean variables and if-then rules. We chose this computational model to cope with one particular specificity of ecological data: observations of ecosystems are very *sparse*. For example, the protist experiments [WWL98; WLW03] may seem quite limited (only 6 species in a microcosm with very few quantitative recordings). Yet, to our knowledge, they are the most advanced laboratory experiments ever performed to record community assembly exhaustively. In ecosystems of bigger scale, data is even sparser, studies may use conflicting variables (for example, species, genre or functional groups), and the influence of a variable on another is often studied in abstraction from the rest of the system. Indeed quantitatively recording the behaviour of an ecosystem is incredibly costly, and often cannot be replicated. Boolean variables can capture very sparse information about the system [Cos22], and if-then rules are expressive enough to encode every qualitative knowledge about the system's behaviour.

Indeed, the knowledge about the dynamics of an ecosystem can often be summarized by sentences such as "*if x happens then y may happen*" which is precisely the semantics of if-then rules. For example, most rules of the Borana model encode this kind of sentences found in the literature [LCD18; LC18; Lia+20; Lia16], see Annex A.2. If-then rules also have the benefit of being *user-friendly*, thanks to their similarity with everyday language even people unfamiliar with modelling are able to understand and employ them. For example, local stakeholders often have a deep understanding of the ecosystem dynamics that can be used to mitigate the data sparsity problem. But only if they are able to take part in the modelling process. Indeed, one of the main assets of the STM framework is that it enables collaborative modelling between scientists and local stakeholders [Bes+17]. If-then rule modelling with qualitative variables had already been proposed before to model

ecosystem dynamics precisely for its ability to capture sparse information and for its user-friendliness [Ryk89; Sta90].

In addition, RR is expressive enough to encode any existing STG. Most STGs found in ecology are directly drawn from empirical observations (e.g. field studies or laboratory experiments), for example the Borana STMs [LC18] or the protists assembly graph [WLW03], and not computed from a formal model. Yet, analysis of such empirical STGs would still benefit from the methods presented in this thesis. To this end, they need to be encoded as a computational model coupled with an analysis toolbox. Any STG can be elementarily encoded as an RR model by creating a variable for every state $s \in \mathcal{S}$, and creating a rule `[a]s+>>s'+, s-` for every $(s, a, s') \in \rightarrow$. Any state label $p \in \mathcal{P}_{\mathcal{S}}$ can additionally be encoded as a state formula: $p = \bigvee_{\{s \in \mathcal{S} | p \in \mathcal{V}_{\mathcal{S}}(s)\}} s$. Thus `ecco` can be used not only as a modelling framework, but also as an analysis toolbox for empirical STGs.

Future extensions of RR

The most immediate extension of RR is undoubtedly to use *multivalued qualitative variables*. Such variables are typically used in biology to model phenomena where a reactant regulates distinct reactions that occur at distinct thresholds. Although data is sparse in ecology, such information is often available for at least some variables or processes, and thus could be included in the modelling. For example in the Borana ecosystem, fire is rare when woody plant cover is above a threshold of 40% [Arc+09; Lia+20], thus trees could be more precisely described as multivalued rather than as Boolean: $\texttt{Tr} \in \{\texttt{none}, \texttt{low}, \texttt{high}\}$ corresponding respectively to 0%, $< 40\%$ and $\geqslant 40\%$. In RR, the multiple values of a variable and their potential order would be defined during variable declaration along with its initial value(s). If the values of a variable are ordered, rules guards could use this ordering with expressions such as $\texttt{Tr} \geqslant \texttt{low}$.

Priority rules is a feature that was included in previous presentations of RR [Cos+22; PTG22a], but that is missing in this thesis. The semantics of priority rules is that low-priority rules can only be fired if no high-priority rule is enabled, thus high-priority rules are always preferred over low-priority ones. High-priority rules are typically used to model cascading effects that need to be resolved before any other low-priority transition is fired. For example, priority rules could be used in the *Borana model* to split the removal of animals from the removal of their food in the rules. To do so, new priority rules of the form "`food- >> animal-`" should be added, and `animal-` should be removed from the effect of the already existing rules causing `food-`. Priority rules make the semantics more convoluted, for example it is unclear what is the semantics of a cycle of high-priority rules, especially as modellers often want to hide high-priority transitions in the STG. Yet, when high-priority rules cannot cycle, they can be incorporated inside low-priority rules, that may have to be duplicated if high-priority rules imply branching, by chaining their effects. Moreover, neither the Borana model nor the protists model has any priority rule. Hence, we chose to skip priority rules in this thesis, but the methods we presented could be extended to include them.

Other modelling formalisms

Computer science provides with a large range of modelling formalisms computing STGs [BČŠ13; Nal+15; BL16], each fitted to specific features. Most of the methods presented in this thesis are independent of the modelling formalism, as long as it computes an STG, and thus may be used with other modelling languages than RR. As an exhaustive inventory of these modelling formalisms would be tedious, we limit ourselves to features found in existing ecology studies.

In ecology, STGs are often computed from interaction networks, such as differential equations [SA21] or Boolean networks [Cam+11; LaB+13], with possible bridges between them [RM16]. Modelling frameworks based on biological interaction networks have been designed in the field of systems biology [CGR12; Nal+15], for example GINsim [CNT12] handling Boolean networks. An example of this implementation in systems biology is given in [Abo+15], modelling the differentiation of T-helper cells and analysing the model through a partition of the state space based on ARCTL. As the protists model is based on such an interaction network, modelling it with a Boolean network may help to build a bridge between ecology and the formal toolbox developed in systems biology.

When the duration of the transitions between states are available, for example through life history characteristics [Cat+79], they can be incorporated into the STG as transition labels. Timed automata [Pet99; AD94] is a modelling formalism computing such STGs, that can be implemented with the software Uppaal [BDL04] incorporating a model-checker. A concrete example of this implementation in ecology is given in [Lar+12], modelling scenarios of a coral reef ecosystem with timed automata.

Lastly, when the probabilities of the transitions between states are available, they can also be incorporated into the STG as transition labels. Markov chain is a modelling formalism able to produce such STGs. A concrete example of probabilistic STG in ecology can be found in the State-and-Transition Simulation Models [Dan+16], a distinctive approach of the STM framework [Bes+21] focusing on simulation.

7.2.2 FARCTL as temporal logic

In this thesis, we used *Computation Tree Logic (CTL)* [Cla+18b], and its extensions *Action-Restricted Computation Tree Logic (ARCTL)* [PR07] and *Fair ARCTL (FARCTL)*, both (1) to compute sets of states satisfying some properties and (2) to partition the STG with respect to these properties. CTL and (F)ARCTL are state-based temporal logics that can be easily expressed in terms of sets of states in the symbolic perspective [Bur+92; PR07; Thi16], a feature required to build component graphs efficiently. (FAR)CTL expresses properties about branching dynamics between alternative pathways, a feature often found in ecological STGs as illustrated by the Borana and protists examples. Nevertheless, designing (FAR)CTL formulas often turns out to be a difficult exercise for non-expert users [DAC99]. In order to ease this task, we proposed a catalogue of patterns[Mon+08; Lar+12; Tho+22], mapping properties written in English to their CTL translation, see Table 2.1. Defining such a catalogue for FARCTL would be useful future work.

Yet, if ecologists got acquainted with formal definitions such as temporal logics, we believe that it would help them clarify and compare various concepts used in the fields of ecology. Indeed, many STGs found in ecology were drawn as graphical summaries of the knowledge about the dynamics of the studied system [Ber+14; LC18], rather than as actual data [Lon74; WLW03; LCD18], without having a formal definition in mind. However, if STGs can be found repeatedly in so many separate fields of ecology along its history, then they must be a relevant representation of ecosystem behaviour. Such representation certainly deserves a formal conceptual framework, such as the one developed in computer science to analyse Kripke structures and labelled transition systems [BK08; Cla+18b].

The fairness concept was an early request from ecologists during this thesis. The automated systems studied in computer science can be trapped in cycles where only a small portion of the system takes action, and finding such cycles may be the goal of the analysis when looking for bugs. On the contrary, it is often supposed in ecology that all parts of the ecosystem are able to take action in turn more or less fairly. For example, if a model represents seasonal change, we want to exclude from the analysis the maximal paths trapped in a single climatic season, i.e. the maximal paths that are not fair with seasonal change events. Recurring events are commonplace in ecosystems, for example seasonal changes, fires, tides, invasions, or even anthropogenic events such as harvests or taxes. A fairness assumption can encompass multiple distinct fairness constraints [BK08], for example restricting the maximal paths to the ones that are fair toward multiple distinct events. Thus a modelling framework for ecological STGs should let the modeller define a fairness assumption alongside the system description.

ARCTL [PR07] allows to restrict the dynamics of the system to a delimited subset of its actions. As illustrated by the Borana example, the objective behind STMs [Bes+17] is often to find an ecosystem management policy achieving some goal, in this case preventing bush encroachment to preserve livelihood. Such management policies can be represented as action-restrictions, i.e. subsets of enabled actions, or in other words subsets of disabled ones. Instead of building distinct STGs representing distinct management scenarios, we can build a single more intricate STG encompassing every scenario, that we can then restrict to a specific scenario using an action-restriction. Instead of replicating the states between the distinct STGs representing the several scenarios, every state makes a unique appearance in the single STG encompassing every scenario. In addition to the shrinkage of the state-space, ARCTL also has the benefit that we can easily shift between scenarios by changing the action-restriction. Yet, as mentioned in Chapter 6, ARCTL does not enable to select scenarios satisfying some property without enumerating them explicitly.

We could have added directly the transitions allowing to navigate between the distinct STGs representing the distinct management scenarios, i.e. transition changing the scenarios. But then the system would be able to shift between the scenarios uncontrollably, the worst case being that the system is only bumping between scenarios without performing any action. It would blur the dynamics just like uncontrolled invasion transitions blur the dynamics of the protists model. ARCTL solves this issue by specifying the action-restriction at the level of quantifiers. For example in the case studies of Chapter 6, it is the quantifiers of the formulas that are shifting in a controlled manner between scenarios, or that are enabling invasions temporarily. Thus, we can specify in the formula itself if and when the system shifts between scenarios, or if and when invasions happen. As a

consequence, we can check for example if a scenario can reverse the effect of another by shifting between them along the formula.

Fair ARCTL pushes a little further this definition of scenario by appending a fairness restriction to it. A scenario therefore delimits not only a set of enabled actions, but also a set of fairness constraints shaping the maximal paths. Indeed while some fairness constraints may be global across all scenarios, such as the changing seasons, others may be related to particular scenarios, such as fire happening infinitely often in ecosystems not managed by humans. As a fairness assumption can be specified in a formula at the level of the quantifiers, just like with action-restriction, we can still shift between scenarios along a formula.

An FARCTL model-checker has been implemented inside `ecco`, using the symbolic algorithm presented in Chapter 5. This algorithm computes the set of states satisfying a given FARCTL formula in the symbolic perspective, thus we can use it to efficiently partition the state-space in order to build component graphs. All three fairness categories, namely *unconditional*, *weak*, and *strong*, are available, using both state-events or action-events, apart from strong fairness constraints with an action-event as first event. Indeed this specific case involves a jump in algorithmic complexity, because it requires checking these action-events in a synchronized way when taking a transition. Nevertheless, most usual fairness constraints are available, for example the traditional action fairness [BK08, chap. 3.5] only involves action-events as second events, and our applications did not require these problematic fairness constraints so far.

Perspectives

As noted just above, the FARCTL symbolic model-checking algorithm presented in Chapter 5 does not manage strong fairness constraints with action-event as first event. But, as mentioned in Chapter 5, the algorithm could already handle a single strong fairness with action-event as first event. If an application requires multiple such fairness constraints, then the algorithm would need to be revised. For example, if only a few of such fairness constraints are needed, a straightforward enumeration of the combination of synchronized action-events may provide a solution with a tractable complexity.

On another topic, we mentioned in the protists case studies of Chapter 4 that the model was not able to handle states that are nondeterministically stable, i.e. that are stable in some maximal paths but not in others. Indeed, the following axiom of temporal logics may be perturbing for ecologists: the maximal paths must always move forward if at least one transition is available. Yet, it is not always the case in ecosystem dynamics: the ecosystem may remain forever in a given state even though some transitions are available. For example, in the first protists experiment [WWL98], in half of the replicates starting from $\{P, C\}$ there is a transition towards $\{P\}$, while in the other half $\{P, C\}$ is stable. In a similar fashion, the STM framework [WWN89] distinguishes *"transient states"* in which the ecosystem cannot persist indefinitely from *"persistent states"* where it can persist, even though some particular events may bring the system out of these persistent states. This behaviour could be modelled with fair actions and an alternative semantics of maximal paths. Suppose that the actions are partitioned between unfair and fair ac-

tions [BK08, chap. 3.5], i.e. we can partition the actions between these that *can* happen and these that *will* happen. Then, we can redefine maximal paths as infinite paths satisfying every fairness constraints, or finite paths ending in a any state s whose self-loop, i.e. the infinite path $s \xrightarrow{\perp_{\mathcal{A}}} s \xrightarrow{\perp_{\mathcal{A}}} \ldots$, satisfies every fairness constraints. Thus, if there are no unconditionally fair actions, a state where only actions that *can* happen are available will be considered stable for some maximal paths, even though other maximal paths that move forward from this state are also considered. Note that this approach is more subtle than just adding self-loops in these new potentially stable states, because it also ensures that all infinite maximal paths are fair. The symbolic algorithm for FARCTL presented in Chapter 5 can easily be adapted to this new semantics by expanding the finite case of $\exists_\alpha \vec{\mathrm{x}}_{\models e}(Z)$ and $\exists_\alpha \vec{\mathrm{x}}_{\not\models e}(Z)$ to the states where only actions that *can* happen are available: $\neg\exists_{\alpha \wedge must}\mathsf{X}\top$.

If the STG includes transition durations or probabilities, then these features can be leveraged using appropriate CTL extensions. Timed CTL [ACD93] extends CTL with duration restrictions on modalities $\mathsf{F}, \mathsf{G}, \mathsf{U}$. An example of the use of TCTL to analyze the scenarios of a coral reef fishery is given in [Lar+12] alongside a catalogue of pattern mapping properties in English with TCTL formulas. Note also that fairness can be seen as a form of qualitative temporality. In contrast to the crisp quantitative temporal properties specified by TCTL, FARCTL can express looser temporal properties about pathways such as the infinite occurrence of an event. On the other hand, Probabilistic CTL [HJ94] extends CTL with quantifiers allowing to query about the probability of some paths. An example of the use of PCTL to analyze the stability of frog populations is given in [Bar+15].

Another temporal logic worth considering is Alternating-time Temporal Logic (ATL) [AHK02] that expresses properties of an STG that describes a game between players, for example, one or several ecosystem managers and the environment. In such an STG, every state is a state of the ecosystem, and every transition is a possible move in the game between the players. ATL quantifiers allow querying if a given player (or set of players) has a strategy to enforce/prevent a certain behaviour of the system. The ATL symbolic model-checking algorithm [AHK02] is similar to the FARCTL symbolic model-checking algorithm presented in Chapter 5. ATL seems particularly fitted to the STM framework, as exemplified by the following citation taken from the STM foundational article:

> *"Under the state-and-transition model, [ecosystem] management would not see itself as establishing a permanent equilibrium. Rather, it would see itself as engaged in a continuing game, the object of which is to seize opportunities and to evade hazards, so far as possible. [...] Transitions between states are triggered by natural 'events' (e.g., weather, fire) or by management 'actions' (change in stocking rate, burning, destruction or introduction of plant populations, fertilization). Very often a combination of the two may be needed." [WWN89]*

Yet, in order to make use of ATL, the system description has to be formulated as a game between players. Thus the RR modelling language would need to be extended to describe the players and their moves, for example by tagging the rules with the players that can perform the associated move.

7.3 Interactive component graphs with ecco

At the inception of this thesis, ecco was based upon *explicit STGs* [GP19; Mao+21; Cos+22; Her+22], i.e. enumerating the states individually, that users would compact using topological components such as strongly connected components or dead-ends. For example, the states forming a strongly connected component could be merged into a single node representing the SCC as a whole. As ecco's models often yielded huge STGs, from hundreds of states [Cos+22] to thousands [Mao+21] or even millions in unpublished exploratory work, users were often overwhelmed by the explicit STGs (when ecco managed to display them, which was not always easy for the biggest models). Starting from a large explicit STG and incrementally compacting it proved to be troublesome, for example merging every SCCs sometimes produced a graph enclosing still hundreds of nodes representing each a distinct SCC. Users were often lost in front of the complexity of their model's STG, see Figure 7.1, and confined themselves to looking for dead-ends or global properties of the STG such as strong connectivity or freedom of dead-ends.

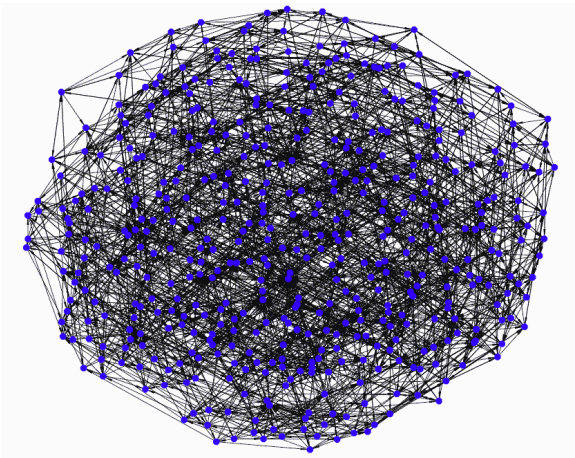


Figure 7.1: Example explicit STG built with ecco. 482 nodes and 2640 transitions, from [Cos+22].

Component graphs [PTG22a] take the opposite view compared to explicit STGs: instead of increasingly compacting the STG they propose to increasingly refine a partition. Starting from the simplest partition composed of a single enclosing component ($\mathcal{C} = \{\mathcal{S}\}$), the user incrementally splits the components with respect to some properties, see the examples of Chapter 4. As a consequence the number of components can be controlled as every individual split only divide a component into two sub-components. Instead of being overwhelmed by a huge number of individual states and transitions, the user progressively sharpens their understanding of the system at each split, new questions arising as the partition is refined. ecco proposes to interactively explore the information collected about each component, helping the user to design further questions and splits. The user has incrementally refined the partition from the start, and thus is able to fully understand it even if it ends up being quite complex. The partitioning of a given STG can be refined in many distinct ways, resulting in distinct component graphs answering distinct questions. We hope that this workflow is intuitive for ecologists, indeed it seems to match the following citation taken from the STM foundational article:

> *"As a general rule, one would distinguish two states only if the difference between*

> *them represented an important change in the land from the point of view of management. [...] It follows that a given [ecosystem] could be described in terms of a greater or lesser number of states and transitions, depending on the nature and objectives of management and on the state of existing knowledge. There would not be a single correct description." [WWN89]*

The first use of ecco's component graph workflow by ecologists is promising [Cos22]. Although explicit STGs are easier to understand thanks to their proximity to the semantics, and although temporal logics form a difficult learning step, ecologist users were able to articulate more complex questions and to produce more refined analyses using the component graph workflow. Yet we believe that the explicit perspective and the component graph perspective are complementary. Indeed explicit STGs are a better introduction to formal analysis and may provide interesting information about moderate-size STGs such as the protists model, see Figure 4.22.

7.3.1 Perspectives

The most immediate extension of ecco's component graph perspective would be to display the successive splits hierarchically. When splitting a component into two parts, in addition to drawing these two new sub-components, we could also draw the initial components as a box enclosing them, see Figure 7.2. Thus the component graph would fully display the history of the partition refinement, i.e. the successive splitting steps. This may help the interactive exploration of the system's dynamics by making the interpretation of the component graph more straightforward. It may also help ecologists to grasp the component graph perspective faster by making it more self-explanatory.

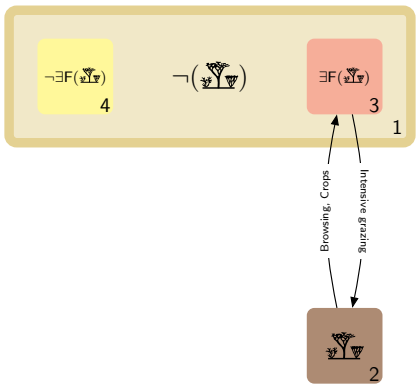


Figure 7.2: Hierarchical component graph. The component #1 has been split into components #3 and #4, and is drawn as a box enclosing both sub-components. Adapted from Figure 4.10.

Another extension of ecco's component graph perspective would be to allow to restrict its transition relation: $\rightarrow_{\mathcal{C}}|_{\alpha}$. Indeed, when using FARCTL to split components, the component graph may be confusing as we used a restricted transition relation to explore the system's behaviour, yet the unrestricted transition relation is displayed in the component graph. The restriction of the component graph's transition relation needs to be user tuned as a single FARCTL formula may use several distinct action-restrictions. Thus, in such a case, there is no obvious restriction for the component graph's transition relation itself.

Using simultaneously distinct restricted transition relations in the component graph may also prove valuable. For example, in the protists model we could display simultaneously on the one hand the invasions and on the other hand the rest of the transitions as distinct restricted transition relations, see Figure 7.3. In addition, the component graph's transitions could be defined by FARCTL formulas, as in [Abo+15, Fig.3] where the transitions are defined by $s_1 \xrightarrow{\alpha} s_2$ iff $s_1 \models \exists_\alpha \mathsf{F}(s_2 \wedge \forall_\alpha \mathsf{G} s_2)$.

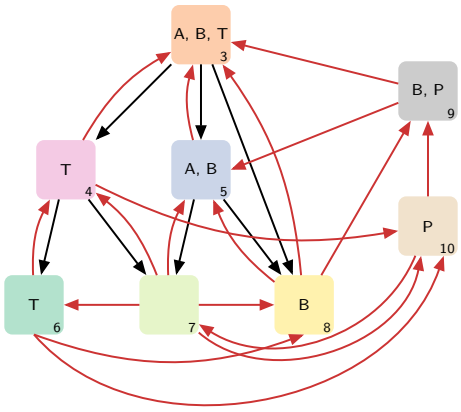
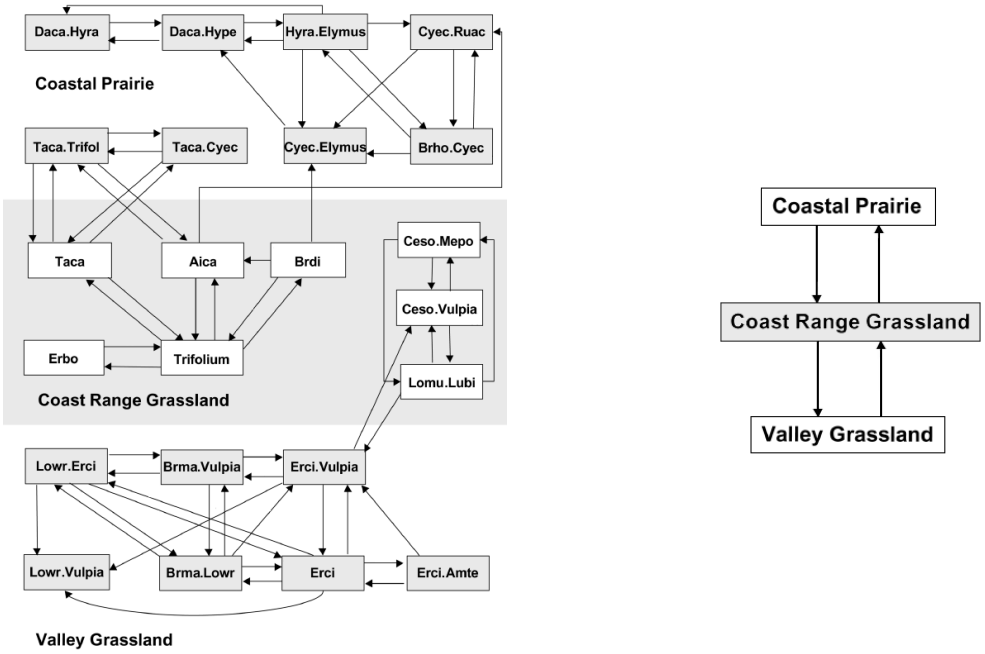


Figure 7.3: Component graph with distinct restricted transition relations. The invasion transitions are displayed with red arrows, while the rest of the transitions are displayed with black arrows. Adapted from Figure 4.20.

Furthermore, component graphs can sometimes be misleading because every path in a component graph does not always match a path in the STG. For example, if a component incorporates disconnected portions of the STG, then a path going through this component may not match a real path in the STG, see Figure 7.4. To ensure that the component graph is not misleading, we would like to provide splitting blueprints ensuring that some properties are preserved in the component graph, for example that any path in the component graph matches a path in the STG.

Instead of being user-driven, the partition refinement could be an automatic procedure driven by an equivalence relation between states acting as a splitting blueprint and resulting in a quotient graph. For example, Figure 7.5a shows such splitting blueprint [Ver22], partitioning any STG with respect to the reachability of a *stable* state property φ, i.e. such that $\varphi \Rightarrow \forall \mathsf{G}\varphi$ meaning that the property that cannot be left once reached. This blueprint goes exhaustively through all possible cases, for example that we will always reach φ (component #5) or that we cannot reach φ at all (component #2). When using this blueprint to partition a given STG, some components may be empty because no state in the STG satisfies the corresponding case, resulting in a quotient graph that is a subgraph of the blueprint, see Figure 7.5b. Because we have the blueprint, we know which components are missing in the quotient graph and we can display this information, see Figure 7.5. Knowing which behaviours are missing in the quotient graph can prove to be valuable information for the understanding of the system dynamics. Conversely, if we had built Figure 7.5b incrementally without the blueprint, we would not have known which cases were missing in the component graph. For example, equivalent component graphs were built without the blueprint in [Cos22] and in [PTG22a, Fig.9]. Such blueprints can also ensure properties between the STG and the component graph, like path equivalence or bisimilarity [Ver22]. For example, the splitting blueprint of Figure 7.5 ensures that every



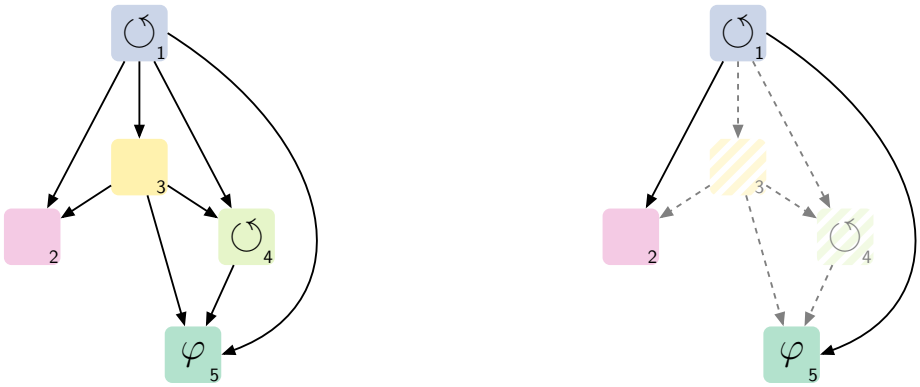
(a) STG **(b)** Component graph

Figure 7.4: Example of a component graph's path not matching any path in the STG. Vegetation succession of Californian grasslands from [JB02, Fig.2]. The path of the component graph: *Coastal Prairie* $\rightarrow$ *Coast Range Grassland* $\rightarrow$ *Valley Grassland* does not match any path in the STG.

path of the quotient graph matches a path of the STG. This method could be generalized into a catalogue mapping questions, such as the reachability of stable property, to splitting blueprints, such as Figure 7.5a, ensuring properties between the STG and the component graph built from the blueprint.

A comparable method would be to build the *hybrid product* of an STG and a *Büchi automaton* [Dur+11], resulting in an automaton whose nodes include sets of the STG's states. Building the product of an STG and a Büchi automaton is a classical procedure for the model-checking of LTL formulas [Cla+18b], a very popular temporal logic expressing properties of individual maximal paths. Such a hybrid product automaton is very similar to a component graph, as it is a hybrid structure mixing symbolic sets of states and explicit transitions between them. In contrast with component graphs, the "components" of a hybrid product automaton do not form a partition of $\mathcal{S}$ in general. Indeed, the same state of the STG may be found in distinct nodes of the hybrid product automaton. In addition, each "component" of a hybrid product automaton includes a node of the original automaton, and the hybrid product automaton itself includes an acceptance condition. All this information can be used to interpret the hybrid product automaton. For example, an LTL formula can be translated into a generalized Büchi automaton using the tableau method [Dur11], resulting in an Büchi automaton whose nodes and edges can be labelled by the section of the formula they check, see Figure 7.6. Such a Büchi automaton can be seen as another kind of splitting blueprint, not resulting in a quotient graph but in a hybrid product automaton.

These various kinds of splitting blueprints may be gathered inside a catalogue map-



(a) Splitting blueprint **(b)** Example quotient graph

Figure 7.5: Example splitting blueprint and quotient graph. Component #5 consists of the states satisfying φ and of the states that necessarily lead to φ, i.e. satisfying $\forall F\varphi$. Component #2 consists of the states that cannot lead to φ, i.e. satisfying $\forall G\neg\varphi$. Component #4 consists of the states that lead to φ, but that can delay infinitely reaching it. Component #3 consists of the states that lead to components #2, #4 and #5 but that cannot delay infinitely reaching them. Component #1 consists of the states that lead to all other components, and that can delay infinitely reaching them. Missing components and transitions in the example quotient graph are displayed striped and dashed. Adapted from [Ver22].

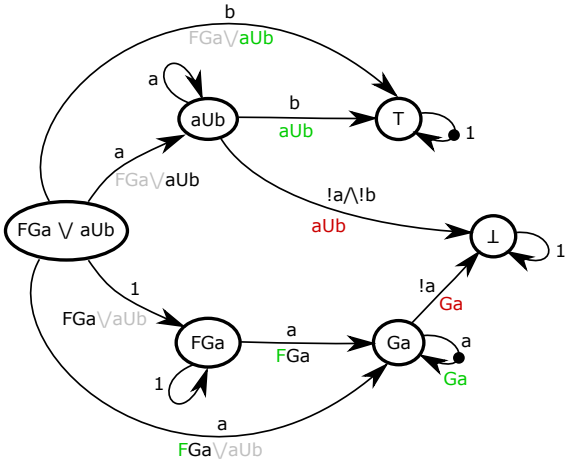


Figure 7.6: Generalized Büchi automaton built with the tableau method.

ping question patterns to splitting blueprints (equivalence relation if the question can be phrased in CTL or Büchi automaton if the question can be phrased in LTL). These splitting blueprints could be explained in detail as they form generic patterns for the analysis of STGs. Properties, such as path equivalence or bisimilarity between the STG and the component graphs could also be ensured directly by splitting blueprints themselves. We believe that such a catalogue of generic analyses could help ecologists getting familiar with the component graph framework.

7.4 Case studies in ecology

We presented two case studies based on ecology research: the vegetation successions of the Borana zone in Ethiopia and the laboratory assembly of protists communities. Even though both case studies produce STGs, they differ in many aspects. For example, the *Borana model* was built from observed transitions, while the protists model was built upon an interaction network inferred from the experiment. The questions answered also differ between both case studies. The questions asked by the Borana study deal with practical management problems (focusing on special variable values, such as bush encroachment), while the protists study asks theoretical questions about the presence of generic patterns in the dynamics (independently of the variables instantiating them).

Yet, both studies can be gathered inside the same modelling framework, using the same formal tools to analyse them. Indeed, the two case studies represent two sides of the same problem: "*how does an ecosystem behave through time?*" The Borana study is on the more practical side, while the protists study is on the more theoretical side. These two sides come from historically separated fields in ecology, community succession and community assembly, which have been increasingly integrated together in recent times but still differ in some aspects because of their legacy [YCH01]. We believe that the logical modelling framework presented in this thesis, and developed in computer science and systems biology, can help join together these two sides that still too often tend to overlook each other.

7.4.1 Borana vegetation community successions

The *Borana model* [Tho+22] is built upon the STMs developed by Liao et al. [LC18; LCD18; Lia+20; Lia16] using plant survey, cattle tracking and classification of satellite imagery. From the description of the vegetation states and the description of the transitions between them, we extracted a set of variables and a set of if-then rules driving the Borana vegetation successions. The complete modelling methodology is described in Annex A.2. The STM framework promotes participatory workshops between ecosystem managers and scientists to build models collectively. Our proposed modelling methodology shares this concern of remaining as user-friendly as possible, so that ecology experts or professionals untrained in computer science can manage to build their own model [Mao+21; Cos22]. Indeed, if-then rules are easily understood even by inexperienced users and match the available data in ecology that often remains sparse and/or qualitative, especially in practical studies. As demonstrated by the *Borana model*, modelling tools like `ecco` are able both to encode experimentally observed behaviours (such as the transitions observed by satellite imagery) and to forecast novel behaviours based upon well founded knowledge about processes and transitions [WW20] (such as the incorporation of browsing livestock in the model).

The questions raised by STMs are often down-to-earth. They can mostly be summarised as "*how can we enforce/prevent this particular behaviour of the ecosystem?*". This question is very close to the search for new drugs in systems biology [Flo+15; Bia18], except that management policies translate into action-restrictions while drugs translate into variable

forcing. The methods designed in systems biology could surely be adapted to look for ecosystem management policies. Ecologists and managers need to be able to discuss the analysis results in order to choose which particular policy they want to implement. Visual results, for example in the form of component graphs, can be user-friendly enough to be understood even by untrained users. The visual and intuitive aspect of STMs is one of their most important assets, and it should be kept in mind when designing a mathematical framework embracing them.

The STM framework is increasingly used, for example by the USA administration to catalogue its ecosystems [Cau13], yet a mathematical setting to compare the behaviours of several STMs is still lacking [Bes+16; WW20]. Computer science could provide such a formal framework, for example by comparing component graphs that are abstractions of such STMs in a unifying description setting. Another way of comparing the behaviours of several STMs is to use model-checking to test some dynamical properties of interest. FARCTL fit with STM analysis as it allows the management policy to shift between scenarios during the dynamics, which underlies many of the ecosystem management questions. Another good candidate temporal logic would be the Alternating Time Logic (ATL) [AHK02] that represents the system's behaviour as the result of a game between players, for example as a game between one or several managers and the environment.

7.4.2 Protists community assembly

The protists model [Her+22] is built upon the interaction network inferred from the protists experiments [WWL98; WLW03]. Each variable represents a protist species, and interactions are translated into if-then rules. One of the most interesting questions raised by the protists case study is *model synthesis*, i.e. the automatic design of a model based upon knowledge about the system's structure and behaviour. Indeed, 47% of the transitions predicted by the protists model were not recorded during the experiment. We can wonder if we could have designed a model more fitted to the observations, i.e. with less unobserved transitions, but based on the same knowledge about the system's structure, i.e. its interaction network. Note that this problem was already raised in [WWL98] in the form of two questions: "*can the behaviour of the system be characterized by a simple set of rules?*" and "*to what extent does knowledge of the results from the pairwise species combinations allow prediction of the outcomes of the more species-rich sets?*". This problem is an active question in computer science, see for example [Che+19] for the model synthesis of a Boolean network based upon biological constraints. From the community assembly perspective, the goal would be to automatically design assembly models based on interaction networks. This would enable *in silico* assembly experiments based upon the knowledge about the interaction network [SA21], a knowledge way cheaper to obtain than performing a replicated laboratory experiment such as the protists ones [WWL98; WLW03].

The questions raised by assembly graphs are often theoretical, not focusing on specific species but on the presence or absence of generic patterns in the dynamics. These questions may be answered by iterating upon the possible instances of these patterns, as we performed in our case studies. Yet some questions are tedious to solve this way, for example: "*Are there stable communities that cannot be put back together by sequential assembly*

using just the species they contain?" [WLW03]. This question would require iterating over all stable communities and then nesting an iteration upon every possible invasion sequence for every stable community. These kinds of questions cannot directly be formulated in FARCTL or other temporal logic. An appropriate language to do so remained to be defined (or identified if one already exists).

8 Logbook of a journey across disciplinary borders

During the elaboration of this thesis, I moved back and forth between computer science and ecology. It was sometimes confusing; for example, many separate concepts of state-transition graphs exist in ecology, as presented in Chapter 1, but they were not gathered inside a unifying framework. Once gathered inside the STG framework, these ecological graphs can be compared to the similar concepts of Kripke structure or labelled transition system in computer science. Yet, the meaning of these concepts slightly differs between the two disciplines. For example, every maximal path of a Kripke structure is a concrete execution of an automated system, even degenerated paths that cycle in a minor subpart of the graph. But conversely, every maximal path of an STG does not always match a realistic trajectory of an ecosystem, because of implicit realism constraints such as the seasonal cycle. The concept of infinite maximal path proved to be particularly puzzling for ecologists because of these implicit realism constraints. The introduction of fairness in Chapter 5 aims at providing a formal framework to articulate these realism constraints. Defining precisely the realism constraints may help ecologists produce more deliberate modelling. For example, we could relax the constraint on seasonal change if some area has a micro-climate without seasonal change, as it sometimes happens in coastal regions.

Similarly, model-checking techniques can be applied to both Kripke structures modelling automated systems and STGs modelling ecosystems. Yet, the goals and questions slightly differ between the two disciplines. In computer science, model-checking aims at proving that an automated system is bug-free, thus the main question is: "*is there any bug?*", and the desired output is "*no*". In ecology, the main question is: "*does that behaviour happen ? can it be enforced or prevented?*" and the expected output is: "*it can happen / be enforced / be prevented only in these specific cases*". In computer science, the outcome of a model-checking process is the modification of the automated system to remove faulty behaviours. In ecology, the desired outcome is the differentiation between the trajectories satisfying the targeted behaviour and these that do not, as well as the branching between them. Indeed, the ecosystem cannot be modified drastically. But, as humans are often part of it, they may influence it from inside in order to enforce or prevent targeted behaviours. In systems biology, which already uses model-checking techniques, the perspective also differs slightly from either computer science or ecology because the system is perceived as an unalterable chemical machine that can only be influenced using external forces such as drugs.

These divergent perspectives on the same concepts and techniques can cause misunderstandings between computer scientists and ecologists. We are forced to articulate these perspectives that otherwise often remain implicit. Once the ecological perspective about STG and model-checking techniques is made explicit, it can raise new questions for

computer scientists. For example, ecologists would like a visual output representing the partition between the maximal paths satisfying or not satisfying targeted behaviours, as well as the branching between them. I tried to tackle this challenge in this thesis using component graphs, but much work is still to be done on this subject.

Most existing ecological STGs have a limited size, often a few dozen of states at most, and thus may not require automated analysis. Yet, I believe that fact is likely due to the lack of or unawareness about existing automated analysis tools. For example, satellite imagery can produce massive empirical STGs, just like the Borana vegetation classification [LCD18] but with finer classes. However, even in the case of small STGs, a formal framework is lacking for the often handmade analysis of the dynamics. Computer science may provide such a formal framework, and the existing concepts for ecological STGs could thus be formally defined, compared, and investigated.

In conclusion, I believe there would be much to learn for both fields if computer science and ecology worked collaboratively. Making computer science concepts and techniques understandable for ecologists, and conversely, is challenging yet rewarding work because both fields must make an effort to understand the other side's perspective. It is the part of this thesis that interested me the most and that I am the proudest of. I hope that I achieved to draft a bridge between these two fields that too often overlook each other.

Bibliography

[Abo+15] Wassim Abou-Jaoudé, Pedro T. Monteiro, Aurélien Naldi, Maximilien Grandclaudon, Vassili Soumelis, Claudine Chaouiya, and Denis Thieffry. "Model Checking to Assess T-Helper Cell Plasticity". In: *Frontiers in Bioengineering and Biotechnology* 2 (2015), p. 86. issn: 2296-4185. doi: 10.3389/fbioe.2014.00086.

[Abo+16] Wassim Abou-Jaoudé, Pauline Traynard, Pedro T. Monteiro, Julio Saez-Rodriguez, Tomáš Helikar, Denis Thieffry, and Claudine Chaouiya. "Logical Modeling and Dynamical Analysis of Cellular Networks". In: *Frontiers in Genetics* 7 (2016). issn: 1664-8021. doi: 10.3389/fgene.2016.00094.

[ACD93] R. Alur, C. Courcoubetis, and D. Dill. "Model-Checking in Dense Real-Time". In: *Information and Computation* 104.1 (May 1993), pp. 2–34. issn: 0890-5401. doi: 10.1006/inco.1993.1024.

[AD94] Rajeev Alur and David L. Dill. "A Theory of Timed Automata". In: *Theoretical Computer Science* 126.2 (Apr. 1994), pp. 183–235. issn: 0304-3975. doi: 10.1016/0304-3975(94)90010-8.

[AHK02] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. "Alternating-Time Temporal Logic". In: *Journal of the ACM* 49.5 (Sept. 2002), pp. 672–713. issn: 0004-5411. doi: 10.1145/585265.585270.

[Arc+09] Sally Archibald, David P. Roy, Brian W. Van WILGEN, and Robert J. Scholes. "What Limits Fire? An Examination of Drivers of Burnt Area in Southern Africa". In: *Global Change Biology* 15.3 (2009), pp. 613–630. issn: 1365-2486. doi: 10.1111/j.1365-2486.2008.01754.x.

[Bar+15] Roberto Barbuti, Pasquale Bove, Paolo Milazzo, and Giovanni Pardini. "Minimal Probabilistic P Systems for Modelling Ecological Systems". In: *Theoretical Computer Science*. From Computer Science to Biology and Back 608 (Dec. 2015), pp. 36–56. issn: 0304-3975. doi: 10.1016/j.tcs.2015.07.035.

[Bar+18] Isabel C. Barrio, David S. Hik, Jóhann Thórsson, Kristín Svavarsdóttir, Bryndís Marteinsdóttir, and Ingibjörg Svala Jónsdóttir. "The Sheep in Wolf's Clothing? Recognizing Threats for Land Degradation in Iceland Using State-and-Transition Models". In: *Land Degradation & Development* 29.6 (2018), pp. 1714–1725. issn: 1099-145X. doi: 10.1002/ldr.2978.

[Bat+05] Grégory Batt, Delphine Ropers, Hidde de Jong, Johannes Geiselmann, Radu Mateescu, Michel Page, and Dominique Schneider. "Validation of Qualitative Models of Genetic Regulatory Networks by Model Checking: Analysis of the Nutritional Stress Response in Escherichia Coli". In: *Bioinformatics* 21.suppl_1 (June 2005), pp. i19–i28. issn: 1367-4803. doi: 10.1093/bioinformatics/bti1048.

[BČŠ13] Luboš Brim, Milan Češka, and David Šafránek. “Model Checking of Biological Systems”. In: *Formal Methods for Dynamical Systems. SFM 2013.* Ed. by Marco Bernardo, Erik de Vink, Alessandra Di Pierro, and Herbert Wiklicky. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 63–112. isbn: 978-3-642-38874-3. doi: 10.1007/978-3-642-38874-3_3.

[BDL04] Gerd Behrmann, Alexandre David, and Kim G. Larsen. “A Tutorial on Uppaal”. In: *Formal Methods for the Design of Real-Time Systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, Bertinora, Italy, September 13-18, 2004, Revised Lectures*. Ed. by Marco Bernardo and Flavio Corradini. Berlin, Heidelberg: Springer, 2004, pp. 200–236. isbn: 978-3-540-30080-9. doi: 10.1007/978-3-540-30080-9_7.

[Béa+21] Jonas Béal, Lorenzo Pantolini, Vincent Noël, Emmanuel Barillot, and Laurence Calzone. “Personalized Logical Models to Investigate Cancer Response to BRAF Treatments in Melanomas and Colorectal Cancers”. In: *PLOS Computational Biology* 17.1 (Jan. 2021), e1007900. issn: 1553-7358. doi: 10.1371/journal.pcbi.1007900.

[Bér+13] D. Bérenguier, C. Chaouiya, P. T. Monteiro, A. Naldi, E. Remy, D. Thieffry, and L. Tichit. “Dynamical Modeling and Analysis of Large Cellular Regulatory Networks”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 23.2 (June 2013), p. 025114. issn: 1054-1500. doi: 10.1063/1.4809783.

[Ber+14] Y. Bergeron, H. Y. H. Chen, N. C. Kenkel, A. L. Leduc, and S. E. Macdonald. “Boreal Mixedwood Stand Dynamics: Ecological Processes Underlying Multiple Pathways.” In: *Forestry Chronicle* 90.2 (2014), pp. 202–213. issn: 0015-7546.

[Bes+16] Brandon T. Bestelmeyer, Jeb C. Williamson, Curtis J. Talbot, Greg W. Cates, Michael C. Duniway, and Joel R. Brown. “Improving the Effectiveness of Ecological Site Descriptions: General State-and-Transition Models and the Ecosystem Dynamics Interpretive Tool (EDIT)”. In: *Rangelands*. Ecological Sites for Landscape Management 38.6 (Dec. 2016), pp. 329–335. issn: 0190-0528. doi: 10.1016/j.rala.2016.10.001.

[Bes+17] Brandon T. Bestelmeyer, Andrew Ash, Joel R. Brown, Bulgamaa Densambuu, María Fernández-Giménez, Jamin Johanson, Matthew Levi, Dardo Lopez, Raul Peinetti, Libby Rumpff, and Patrick Shaver. “State and Transition Models: Theory, Applications, and Challenges”. In: *Rangeland Systems: Processes, Management and Challenges*. Ed. by David D. Briske. Springer Series on Environmental Management. Cham: Springer International Publishing, 2017, pp. 303–345. isbn: 978-3-319-46709-2. doi: 10.1007/978-3-319-46709-2_9.

[Bes+21] Brandon Bestelmeyer, Maria Fernández-Giménez, Bulgamaa Densambuu, and Retta Bruegger. “State-and-Transition Modelling”. In: *The Routledge Handbook of Research Methods for Social-Ecological Systems*. Routledge, 2021. isbn: 978-1-00-302133-9.

[Bia18] Célia Biane. “Behavioral Reprogramming : Models, Algorithms and Application to Complex Diseases”. Theses. Université Paris-Saclay ; Université d’Evry-Val-d’Essonne, Nov. 2018.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, Apr. 2008. isbn: 978-0-262-02649-9.

[BL16] Ezio Bartocci and Pietro Lió. “Computational Modeling, Formal Analysis, and Tools for Systems Biology”. In: *PLOS Computational Biology* 12.1 (Jan. 2016), e1004591. issn: 1553-7358. doi: 10.1371/journal.pcbi.1004591.

[Boe15] Carl Boettiger. “An Introduction to Docker for Reproducible Research”. In: *ACM SIGOPS Operating Systems Review* 49.1 (Jan. 2015), pp. 71–79. issn: 0163-5980. doi: 10.1145/2723872.2723882.

[Bry18] Randal E. Bryant. “Binary Decision Diagrams”. In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Cham: Springer International Publishing, 2018, pp. 191–217. isbn: 978-3-319-10575-8. doi: 10.1007/978-3-319-10575-8_7.

[BS07a] Julian Bradfield and Colin Stirling. “12 Modal Mu-Calculi”. In: *Studies in Logic and Practical Reasoning*. Ed. by Patrick Blackburn, Johan Van Benthem, and Frank Wolter. Vol. 3. Handbook of Modal Logic. Elsevier, Jan. 2007, pp. 721–756. doi: 10.1016/S1570-2464(07)80015-2.

[BS07b] Julian Bradfield and Colin Stirling. “Modal Mu-Calculi”. In: *Handbook of modal logic* 3 (2007), pp. 721–756.

[Bur+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. “Symbolic Model Checking: 1020 States and Beyond”. In: *Information and Computation* 98.2 (June 1992), pp. 142–170. issn: 0890-5401. doi: 10.1016/0890-5401(92)90017-A.

[Cam+11] Colin Campbell, Suann Yang, Réka Albert, and Katriona Shea. “A Network Model for Plant–Pollinator Community Assembly”. In: *Proceedings of the National Academy of Sciences* 108.1 (Jan. 2011), pp. 197–202. issn: 0027-8424, 1091-6490. doi: 10.1073/pnas.1008204108.

[Cat+79] Peter J. Cattelino, Ian R. Noble, Ralph O. Slatyer, and Stephen R. Kessell. “Predicting the Multiple Pathways of Plant Succession”. In: *Environmental Management* 3.1 (Jan. 1979), pp. 41–50. issn: 1432-1009. doi: 10.1007/BF01867067.

[Cau13] Dan Caudle. *Interagency Ecological Site Handbook for Rangelands*. US Department of the Interior, Bureau of Land Management, 2013.

[CF03] Nathalie Chabrier and François Fages. “Symbolic Model Checking of Biochemical Networks”. In: *Computational Methods in Systems Biology*. Ed. by Corrado Priami. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, pp. 149–162. isbn: 978-3-540-36481-8. doi: 10.1007/3-540-36481-1_13.

[CGR12] Miguel Carrillo, Pedro A. Góngora, and David Rosenblueth. “An Overview of Existing Modeling Tools Making Use of Model Checking in the Analysis of Biochemical Networks”. In: *Frontiers in Plant Science* 3 (2012), p. 155. issn: 1664-462X. doi: 10.3389/fpls.2012.00155.

[CH16] Cynthia Chang and Janneke HilleRisLambers. “Integrating Succession and Community Assembly Perspectives”. In: *F1000Research* 5 (Sept. 2016), F1000 Faculty Rev–2294. issn: 2046-1402. doi: 10.12688/f1000research.8973.1.

[Che+19] Stéphanie Chevalier, Christine Froidevaux, Loïc Paulevé, and Andrei Zinovyev. “Synthesis of Boolean Networks from Biological Dynamical Constraints Using Answer-Set Programming”. In: *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. Nov. 2019, pp. 34–41. doi: 10.1109/ICTAI.2019.00014.

[CHV18] Edmund M. Clarke, Thomas A. Henzinger, and Helmut Veith. “Introduction to Model Checking”. In: *Handbook of Model Checking*. Cham: Springer International Publishing, 2018, pp. 1–26. isbn: 978-3-319-10575-8. doi: 10.1007/978-3-319-10575-8_1.

[Cla+18a] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of Model Checking*. Vol. 10. Springer, 2018.

[Cla+18b] Edmund M. Clarke, Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model Checking*. Second. Cyber Physical Systems Series. Cambridge, MA, USA: MIT Press, Dec. 2018. isbn: 978-0-262-03883-6.

[Cle16] Frederic Edward Clements. *Plant Succession: An Analysis of the Development of Vegetation*. 242. Carnegie Institution of Washington, 1916.

[CLZ14] Marie-Odile Cordier, Christine Largouët, and Yulong Zhao. “Model-Checking an Ecosystem Model for Decision-Aid”. In: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. Nov. 2014, pp. 539–543. doi: 10.1109/ICTAI.2014.87.

[CNT12] Claudine Chaouiya, Aurélien Naldi, and Denis Thieffry. “Logical Modelling of Gene Regulatory Networks with GINsim”. In: *Bacterial Molecular Networks: Methods and Protocols*. Ed. by Jacques van Helden, Ariane Toussaint, and Denis Thieffry. Methods in Molecular Biology. New York, NY: Springer, 2012, pp. 463–479. isbn: 978-1-61779-361-5. doi: 10.1007/978-1-61779-361-5_23.

[Cos+22] Maximilien Cosme, Christelle Hély, Franck Pommereau, Paolo Pasquariello, Christel Tiberi, Anna Treydte, and Cédric Gaucherel. “Qualitative Modeling for Bridging Expert-Knowledge and Social-Ecological Dynamics of an East African Savanna”. In: *Land* 11.1 (Jan. 2022), p. 42. issn: 2073-445X. doi: 10.3390/land11010042.

[Cos22] Maximilien Cosme. “Modélisation qualitative à événements discrets des dynamiques d’écosystèmes”. PhD thesis. Université de Montpellier, Mar. 2022.

[Cou+02] Jean-Michel Couvreur, Emmanuelle Encrenaz, Emmanuel Paviot-Adet, Denis Poitrenaud, and Pierre-André Wacrenier. “Data Decision Diagrams for Petri Net Analysis”. In: *Application and Theory of Petri Nets 2002*. Ed. by Javier Esparza and Charles Lakos. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 101–120. isbn: 978-3-540-48068-6. doi: 10.1007/3-540-48068-4_8.

[CP02] H YH Chen and R V Popadiouk. “Dynamics of North American Boreal Mixedwoods”. In: *Environmental Reviews* 10.3 (Sept. 2002), pp. 137–166. issn: 1181-8700. doi: 10.1139/a02-007.

[DAC99] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. “Patterns in Property Specifications for Finite-State Verification”. In: *Proceedings of the 21st International Conference on Software Engineering*. ICSE ’99. New York, NY, USA: Association for Computing Machinery, May 1999, pp. 411–420. isbn: 978-1-58113-074-4. doi: 10.1145/302405.302672.

[Dan+16] Colin J. Daniel, Leonardo Frid, Benjamin M. Sleeter, and Marie-Josée Fortin. “State-and-Transition Simulation Models: A Framework for Forecasting Landscape Change”. In: *Methods in Ecology and Evolution* 7.11 (2016), pp. 1413–1423. issn: 2041-210X. doi: 10.1111/2041-210X.12597.

[Di +20] Cinzia Di Giusto, Cédric Gaucherel, Hanna Klaudel, and Franck Pommereau. "Analysis of Discrete Models for Ecosystem Ecology". In: *Biomedical Engineering Systems and Technologies*. Ed. by Ana Roque, Arkadiusz Tomczyk, Elisabetta De Maria, Felix Putze, Roman Moucek, Ana Fred, and Hugo Gamboa. Cham: Springer International Publishing, 2020, pp. 242–264. isbn: 978-3-030-46970-2. doi: 10.1007/978-3-030-46970-2_12.

[Dur+11] Alexandre Duret-Lutz, Kais Klai, Denis Poitrenaud, and Yann Thierry-Mieg. "Combining Explicit and Symbolic Approaches for Better On-the-Fly LTL Model Checking". In: *arXiv preprint arXiv:1106.5700* (2011). arXiv: 1106.5700.

[Dur11] Alexandre Duret-Lutz. "LTL Translation Improvements in Spot". In: *Fifth International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS 2011) 5*. 2011, pp. 1–12.

[Flo+15] Åsmund Flobak, Anaïs Baudot, Elisabeth Remy, Liv Thommesen, Denis Thieffry, Martin Kuiper, and Astrid Lægreid. "Discovery of Drug Synergies in Gastric Cancer Cells Predicted by Logical Modeling". In: *PLOS Computational Biology* 11.8 (Aug. 2015), e1004426. issn: 1553-7358. doi: 10.1371/journal.pcbi.1004426.

[Gau+21] C. Gaucherel, C. Carpentier, I. R. Geijzendorffer, C. Noûs, and F. Pommereau. "Discrete-Event Models for Conservation Assessment of Integrated Ecosystems". In: *Ecological Informatics* 61 (Mar. 2021), p. 101205. issn: 1574-9541. doi: 10.1016/j.ecoinf.2020.101205.

[GP19] Cédric Gaucherel and Franck Pommereau. "Using Discrete Systems to Exhaustively Characterize the Dynamics of an Integrated Ecosystem". In: *Methods in Ecology and Evolution* 10.9 (2019), pp. 1615–1627. issn: 2041-210X. doi: 10.1111/2041-210X.13242.

[Hau+20] Daniel Hausmann, Tadeusz Litak, Christoph Rauch, and Matthias Zinner. "Cheap CTL Compassion in NuSMV". In: *Verification, Model Checking, and Abstract Interpretation*. Ed. by Dirk Beyer and Damien Zufferey. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 248–269. isbn: 978-3-030-39322-9. doi: 10.1007/978-3-030-39322-9_12.

[Her+22] Mathieu de Goër de Herve, Colin Thomas, Maximilien Cosme, Philip Warren, and Cédric Gaucherel. *Is a Community State Reachable, and Why?* Preprint. Jan. 2022. doi: 10.22541/au.164301558.85072720/v1.

[HJ94] Hans Hansson and Bengt Jonsson. "A Logic for Reasoning about Time and Reliability". In: *Formal Aspects of Computing* 6.5 (Sept. 1994), pp. 512–535. issn: 1433-299X. doi: 10.1007/BF01211866.

[HP93] Luh Hang-Kwang and Stuart L. Pimm. "The Assembly of Ecological Communities: A Minimalist Approach". In: *Journal of Animal Ecology* 62.4 (1993), pp. 749–765. issn: 0021-8790. doi: 10.2307/5394.

[JB02] Randall D. Jackson and James W. Bartolome. "A State-Transition Approach to Understanding Nonequilibrium Plant Community Dynamics in Californian Grasslands". In: *Plant Ecology* 162.1 (2002), pp. 49–65.

[LaB+13] Thomas LaBar, Colin Campbell, Suann Yang, Réka Albert, and Katriona Shea. "Global versus Local Extinction in a Network Model of Plant–Pollinator Communities". In: *Theoretical Ecology* 6.4 (Nov. 2013), pp. 495–503. issn: 1874-1746. doi: 10.1007/s12080-013-0182-8.

[Lar+12] Christine Largouët, Marie-Odile Cordier, Yves-Marie Bozec, Yulong Zhao, and Guy Fontenelle. "Use of Timed Automata and Model-Checking to Explore Scenarios on Ecosystem Models". In: *Environmental Modelling & Software* 30 (Apr. 2012), pp. 123–138. issn: 1364-8152. doi: 10.1016/j.envsoft.2011.08.005.

[LC18] Chuan Liao and Patrick E. Clark. "Rangeland Vegetation Diversity and Transition Pathways under Indigenous Pastoralist Management Regimes in Southern Ethiopia". In: *Agriculture, Ecosystems & Environment* 252 (Jan. 2018), pp. 105–113. issn: 0167-8809. doi: 10.1016/j.agee.2017.10.009.

[LCD18] Chuan Liao, Patrick E. Clark, and Stephen D. DeGloria. "Bush Encroachment Dynamics and Rangeland Management Implications in Southern Ethiopia". In: *Ecology and Evolution* 8.23 (2018), pp. 11694–11703. issn: 2045-7758. doi: 10.1002/ece3.4621.

[Lia+20] Chuan Liao, Arun Agrawal, Patrick E. Clark, Simon A. Levin, and Daniel I. Rubenstein. "Landscape Sustainability Science in the Drylands: Mobility, Rangelands and Livelihoods". In: *Landscape Ecology* 35.11 (Nov. 2020), pp. 2433–2447. issn: 1572-9761. doi: 10.1007/s10980-020-01068-8.

[Lia16] Chuan Liao. "Complexity In The Open Grazing System: Rangeland Ecology, Pastoral Mobility And Ethnobotanical Knowledge In Borana, Ethiopia". PhD thesis. Cornell University, 2016.

[LM93] Richard Law and R. Daniel Morton. "Alternative Permanent States of Ecological Communities". In: *Ecology* 74.5 (1993), pp. 1347–1361. issn: 1939-9170. doi: 10.2307/1940065.

[Lon74] G. Londo. "Successive Mapping of Dune Slack Vegetation". In: *Vegetatio* 29.1 (July 1974), pp. 51–61. issn: 1573-5052. doi: 10.1007/BF02390895.

[LWW00] Richard Law, Anita J. Weatherby, and Philip H. Warren. "On the Invasibility of Persistent Protist Communities". In: *Oikos* 88.2 (2000), pp. 319–326. issn: 1600-0706. doi: 10.1034/j.1600-0706.2000.880210.x.

[Mao+21] Zhun Mao, Julia Centanni, Franck Pommereau, Alexia Stokes, and Cédric Gaucherel. "Maintaining Biodiversity Promotes the Multifunctionality of Social-Ecological Systems: Holistic Modelling of a Mountain System". In: *Ecosystem Services* 47 (Feb. 2021), p. 101220. issn: 2212-0416. doi: 10.1016/j.ecoser.2020.101220.

[May06] Robert M. May. "Network Structure and the Biology of Populations". In: *Trends in Ecology & Evolution*. Twenty Years of TREE - Part 2 21.7 (July 2006), pp. 394–399. issn: 0169-5347. doi: 10.1016/j.tree.2006.03.013.

[McK11] Wes McKinney. "Pandas: A Foundational Python Library for Data Analysis and Statistics". In: *undefined* (2011).

[McM93] Kenneth L. McMillan. *Symbolic Model Checking*. Springer US, 1993. isbn: 978-1-4613-6399-6. doi: 10.1007/978-1-4615-3190-6.

[Meu+17] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz.

"SymPy: Symbolic Computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. issn: 2376-5992. doi: 10.7717/peerj-cs.103.

[Mil05] Millennium Ecosystem Assessment. *Ecosystems and Human Well-Being, Biodiversity Synthesis*. 2005.

[Mon+08] Pedro T. Monteiro, Delphine Ropers, Radu Mateescu, Ana T. Freitas, and Hidde de Jong. "Temporal Logic Patterns for Querying Dynamic Models of Cellular Interaction Networks". In: *Bioinformatics* 24.16 (Aug. 2008), pp. i227–i233. issn: 1367-4803. doi: 10.1093/bioinformatics/btn275.

[Nal+15] Aurélien Naldi, Pedro T. Monteiro, Christoph Müssel, the Consortium for Logical Models and Tools, Hans A. Kestler, Denis Thieffry, Ioannis Xenarios, Julio Saez-Rodriguez, Tomas Helikar, and Claudine Chaouiya. "Cooperative Development of Logical Modelling Standards and Tools with CoLoMoTo". In: *Bioinformatics* 31.7 (Apr. 2015), pp. 1154–1159. issn: 1367-4803. doi: 10.1093/bioinformatics/btv013.

[NS77] I. R. Noble and R. O. Slatyer. "Post-Fire Succession of Plants in Mediterranean Ecosystems". In: *Symposium on Environmental Consequences of Fire and Fuel Management in Mediterranean Ecosystems, Palo Alto, CA, USA*. 1977, pp. 27–36.

[Pat71] BERNARD C. Patten. "A Primer for Ecological Modeling and Simulation with Analog and Digital Computers". In: *Systems Analysis and Simulation in Ecology (Ed. BC Patten) Vol. I*. Vol. 1. Academic Press, 1971, pp. 3–121. isbn: 978-1-4832-7751-6.

[Per18] Jeffrey M. Perkel. "Why Jupyter Is Data Scientists' Computational Notebook of Choice". In: *Nature* 563.7729 (Oct. 2018), pp. 145–146. doi: 10.1038/d41586-018-07196-1.

[Pet99] Paul Pettersson. "Modelling and Verification of Real-Time Systems Using Timed Automata : Theory and Practice". PhD thesis. Uppsala University, Department of Computer Systems, 1999. isbn: 99-2900552-8.

[Phi11] Jonathan D. Phillips. "The Structure of Ecological State Transitions: Amplification, Synchronization, and Constraints in Responses to Environmental Change". In: *Ecological Complexity*. Special Section: Complexity of Coupled Human and Natural Systems 8.4 (Dec. 2011), pp. 336–346. issn: 1476-945X. doi: 10.1016/j.ecocom.2011.07.004.

[Pil+17] Shai Pilosof, Mason A. Porter, Mercedes Pascual, and Sonia Kéfi. "The Multilayer Nature of Ecological Networks". In: *Nature Ecology & Evolution* 1.4 (Mar. 2017), pp. 1–9. issn: 2397-334X. doi: 10.1038/s41559-017-0101.

[PR07] Charles Pecheur and Franco Raimondi. "Symbolic Model Checking of Logics with Actions". In: *Model Checking and Artificial Intelligence*. Ed. by Stefan Edelkamp and Alessio Lomuscio. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 113–128. isbn: 978-3-540-74128-2. doi: 10.1007/978-3-540-74128-2_8.

[PTG22a] Franck Pommereau, Colin Thomas, and Cédric Gaucherel. "EDEN Framework for Interactive Analysis of Ecosystems Models". In: *International Workshop on Petri Nets and Software Engineering (PNSE 2022)*. Vol. 3170. June 2022, p. 119.

[PTG22b] Franck Pommereau, Colin Thomas, and Cédric Gaucherel. “Petri Nets Semantics of Reaction Rules (RR)”. In: *Application and Theory of Petri Nets and Concurrency*. Ed. by Luca Bernardinello and Laure Petrucci. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 175–194. isbn: 978-3-031-06653-5. doi: 10.1007/978-3-031-06653-5_10.

[PV17] Jonathan D. Phillips and Chris Van Dyke. “State-and-Transition Models in Geomorphology”. In: *CATENA* 153 (June 2017), pp. 168–181. issn: 0341-8162. doi: 10.1016/j.catena.2017.02.009.

[Pyt] Python Software Foundation. *The Python Language*.

[RM16] Raina Robeva and David Murrugarra. “The Spruce Budworm and Forest: A Qualitative Comparison of ODE and Boolean Models”. In: *Letters in Biomathematics* 3.1 (Jan. 2016), pp. 75–92. doi: 10.1080/23737867.2016.1197804.

[Ryk89] Edward J. Rykiel. “Artificial Intelligence and Expert Systems in Ecology and Natural Resource Management”. In: *Ecological Modelling* 46.1 (July 1989), pp. 3–8. issn: 0304-3800. doi: 10.1016/0304-3800(89)90066-5.

[SA21] Carlos A. Serván and Stefano Allesina. “Tractable Models of Ecological Assembly”. In: *Ecology Letters* 24.5 (2021), pp. 1029–1037. issn: 1461-0248. doi: 10.1111/ele.13702.

[SB06] Paulo Salles and Bert Bredeweg. “Modelling Population and Community Dynamics with Qualitative Reasoning”. In: *Ecological Modelling*. Selected Papers from the Third Conference of the International Society for Ecological Informatics (ISEI), August 26–30, 2002, Grottaferrata, Rome, Italy 195.1 (May 2006), pp. 114–128. issn: 0304-3800. doi: 10.1016/j.ecolmodel.2005.11.014.

[SFS21] Chuliang Song, Tadashi Fukami, and Serguei Saavedra. “Untangling the Complexity of Priority Effects in Multispecies Communities”. In: *Ecology Letters* 24.11 (2021), pp. 2301–2313. issn: 1461-0248. doi: 10.1111/ele.13870.

[Sta90] Anthony M. Starfield. “Qualitative, Rule-Based Modeling”. In: *BioScience* 40.8 (1990), pp. 601–604. issn: 0006-3568. doi: 10.2307/1311300.

[Thi15] Yann Thierry-Mieg. “Symbolic Model-Checking Using ITS-Tools”. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by Christel Baier and Cesare Tinelli. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 231–237. isbn: 978-3-662-46681-0. doi: 10.1007/978-3-662-46681-0_20.

[Thi16] Yann Thierry Mieg. “From Symbolic Verification To Domain Specific Languages”. Habilitation à Diriger Des Recherches. Sorbonne Université , UPMC ; Laboratoire d’informatique de Paris 6 [LIP6], Dec. 2016.

[Tho+22] Colin Thomas, Maximilien Cosme, Cédric Gaucherel, and Franck Pommereau. “Model-Checking Ecological State-Transition Graphs”. In: *PLOS Computational Biology* 18.6 (June 2022), e1009657. issn: 1553-7358. doi: 10.1371/journal.pcbi.1009657.

[Tho22] Thomas. “Notebooks from the PhD Dissertation "Analysis of State-Transition Graphs of Ecosystems Using Model-Checking"”. In: (Dec. 2022). doi: 10.5281/zenodo.7625156.

[Tit20] Pablo Tittonell. “Assessing Resilience and Adaptability in Agroecological Transitions”. In: *Agricultural Systems* 184 (Sept. 2020), p. 102862. issn: 0308-521X. doi: 10.1016/j.agsy.2020.102862.

[Tra+16] Pauline Traynard, Adrien Fauré, François Fages, and Denis Thieffry. “Logical Model Specification Aided by Model-Checking Techniques: Application to the Mammalian Cell Cycle Regulation”. In: *Bioinformatics* 32.17 (Sept. 2016), pp. i772–i780. issn: 1367-4803. doi: 10.1093/bioinformatics/btw457.

[Ver22] François Vernadat. *Cartographie de Structures de Kripke*. private communcation, work in progress. 2022.

[Wha69] Clifton R. Wharton. “Subsistence Agriculture: Concepts and Scope”. In: *Subsistence Agriculture & Economic Development*. Routledge, 1969, pp. 12–20. isbn: 978-1-315-13040-8.

[WLW03] Philip H. Warren, Richard Law, and Anita J. Weatherby. “Mapping the Assembly of Protist Communities in Microcosms”. In: *Ecology* 84.4 (2003), pp. 1001–1011. issn: 1939-9170. doi: 10.1890/0012-9658(2003)084[1001:MTAOPC]2.0.CO;2.

[WSA12] Rui-Sheng Wang, Assieh Saadatpour, and Réka Albert. “Boolean Modeling in Systems Biology: An Overview of Methodology and Applications”. In: *Physical Biology* 9.5 (Sept. 2012), p. 055001. issn: 1478-3975. doi: 10.1088/1478-3975/9/5/055001.

[WW20] Brian Walker and Mark Westoby. “Past, Present and Future of State and Transition Language”. In: *Rangeland Journal* 42.1 (2020), pp. 71–72. issn: 1036-9872. doi: 10.1071/RJ20020.

[WWL98] Anita J. Weatherby, Philip H. Warren, and Richard Law. “Coexistence and Collapse: An Experimental Investigation of the Persistent Communities of a Protist Species Pool”. In: *Journal of Animal Ecology* 67.4 (July 1998), pp. 554–566. issn: 0021-8790. doi: 10.1046/j.1365-2656.1998.00212.x.

[WWN89] Mark Westoby, Brian Walker, and Imanuel Noy-Meir. “Opportunistic Management for Rangelands Not at Equilibrium”. In: *Journal of Range Management* 42.4 (July 1989), pp. 266–274. issn: 0022-409X. doi: 10.2307/3899492.

[YCH01] Truman P. Young, Jonathan M. Chase, and Russell T. Huddleston. “Community Succession and Assembly Comparing, Contrasting and Combining Paradigms in the Context of Ecological Restoration”. In: *Ecological Restoration* 19.1 (Mar. 2001), pp. 5–18. issn: 1522-4740, 1543-4079. doi: 10.3368/er.19.1.5.

A Appendix

A.1 Proof of the symbolic CTL algorithm

In this section, we prove the definitions of CTL operators as single fixed points presented in Table 2.2. We also provides fixed points definitions for additional temporal operators [BK08] that are implemented inside ecco: W "*weak until*", R "*release*" and M "*strong release*" . Starting from the canonical definition of ∃U as a fixed point [BK08; Cla+18a]:

$$\exists(\varphi_1 \mathsf{U} \varphi_2) = \mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X} Z)$$

From the definition of ∀U as a fixed point for finite and infinite maximal path given in [PR07]:

$$\forall(\varphi_1 \mathsf{U} \varphi_2) = \neg(\mu\, Z.(\neg\varphi_2 \wedge (\neg\varphi_1 \vee \neg\exists \mathsf{X}\top)) \vee (\neg\varphi_2 \wedge \exists \mathsf{X} Z)) \wedge \neg(\nu\, Z.\neg\varphi_2 \wedge \exists \mathsf{X} Z)$$

From the following canonical rewriting of CTL operators [BK08; Cla+18a]:

- $\exists \mathsf{F}\varphi = \exists(\top \mathsf{U} \varphi)$ and $\forall \mathsf{F}\varphi = \forall(\top \mathsf{U} \varphi)$
- $\exists \mathsf{G}\varphi = \neg\forall \mathsf{F}\neg\varphi$ and $\forall \mathsf{G}\varphi = \neg\exists \mathsf{F}\neg\varphi$
- $\exists\varphi_1 \mathsf{W} \varphi_2 = \exists\varphi_1 \mathsf{U} \varphi_2 \vee \exists \mathsf{G}\varphi_1$ and $\forall\varphi_1 \mathsf{W} \varphi_2 = \neg\exists\neg\varphi_2 \mathsf{U} \neg(\varphi_1 \vee \varphi_2)$
- $\exists\varphi_1 \mathsf{R} \varphi_2 = \exists\varphi_2 \mathsf{W} \varphi_1 \wedge \varphi_2$ and $\forall\varphi_1 \mathsf{R} \varphi_2 = \forall\varphi_2 \mathsf{W} \varphi_1 \wedge \varphi_2$
- $\exists\varphi_1 \mathsf{M} \varphi_2 = \exists\varphi_2 \mathsf{U} \varphi_1 \wedge \varphi_2$ and $\forall\varphi_1 \mathsf{M} \varphi_2 = \forall\varphi_2 \mathsf{U} \varphi_1 \wedge \varphi_2$

And using the following theorems:

Theorem 2.1. For any $\varphi \in CTL$ and any state $s \in \mathcal{S}$, one and only one of $\exists \mathsf{X}\varphi$, $\forall \mathsf{X}\neg\varphi$ and $\neg\exists \mathsf{X}\top$ holds in s. Thus they can be rewritten into one another:

$$\begin{aligned} \exists \mathsf{X}\varphi &= \neg(\forall \mathsf{X}\neg\varphi \vee \neg\exists \mathsf{X}\top) \\ \forall \mathsf{X}\neg\varphi &= \neg(\exists \mathsf{X}\varphi \vee \neg\exists \mathsf{X}\top) \\ \neg\exists \mathsf{X}\top &= \neg(\exists \mathsf{X}\varphi \vee \forall \mathsf{X}\neg\varphi) \end{aligned}$$

Theorem 2.2 ([McM93])**.** If $\mathcal{S}$ is finite, and if τ is monotonic ($S \subseteq S' \Rightarrow \tau(S) \subseteq \tau(S')$), then τ has a *least fixed point* noted $\mu\, Z.\tau(Z)$ (the smallest for the set inclusion) and a *greatest fixed point* noted $\nu\, Z.\tau(Z)$ (the largest for the set inclusion).

Moreover $\exists n \in \mathbb{N}$ such that $\mu\, Z.\tau(Z) = \tau^n(\emptyset)$ and $\nu\, Z.\tau(Z) = \tau^n(\mathcal{S})$.

Theorem A.1 (μ/ν duality [BS07a])**.** $\mu\, Z.\tau(Z) = \neg\nu\, Z.\neg\tau(\neg Z)$

Theorem A.2. Let $\tau = \lambda x.f(x) \vee \varphi$ with φ a formula without occurrence of x, and f a function conserving the union: $f(A \cup B) = f(A) \cup f(B)$. Then if the state space $\mathcal{S}$ is finite:

$$\nu Z.\tau(Z) = (\nu Z.f(Z)) \vee (\mu Z.\tau(Z))$$

Proof. Let us prove by recurrence on $i \geq 1$ that $\tau^i(\mathcal{S}) = f^i(\mathcal{S}) \cup \tau^i(\emptyset)$:

- $\tau(\mathcal{S}) = f(\mathcal{S}) \cup \varphi = f(\mathcal{S} \cup \emptyset) \cup \varphi = f(\mathcal{S}) \cup f(\emptyset) \cup \varphi = f(\mathcal{S}) \cup \tau(\emptyset)$
- suppose that $\tau^i(\mathcal{S}) = f^i(\mathcal{S}) \cup \tau^i(\emptyset)$ then:

$$\begin{array}{lll}
\tau^{i+1}(\mathcal{S}) & = \tau(\tau^i(\mathcal{S})) & \\
& = \tau(f^i(\mathcal{S}) \cup \tau^i(\emptyset)) & \text{by recurrence} \\
& = f(f^i(\mathcal{S}) \cup \tau^i(\emptyset)) \cup \varphi & \\
& = f(f^i(\mathcal{S})) \cup f(\tau^i(\emptyset)) \cup \varphi & \text{because } f(A \cup B) = f(A) \cup f(B) \\
\tau^{i+1}(\mathcal{S}) & = f^{i+1}(\mathcal{S}) \cup \tau^{i+1}(\emptyset) &
\end{array}$$

$\mathcal{S}$ is finite, thus by Theorem 2.2, $\exists n \in \mathbb{N}$ such that $\nu Z.\tau(Z) = \tau^n(\mathcal{S})$, $\nu Z.f(Z) = f^n(\mathcal{S})$ and $\mu Z.\tau(Z) = \tau^n(\emptyset)$ (in fact there is a n for each fixed point and we take the max of them). We have $\tau^n(\mathcal{S}) = f^n(\mathcal{S}) \cup \tau^n(\emptyset)$ thus $\nu Z.\tau(Z) = (\nu Z.f(Z)) \vee (\mu Z.\tau(Z))$. □

In particular Theorem A.2 can be applied to $f = \exists \mathsf{X}$ as it conserves the union:

$$\begin{array}{ll}
\exists \mathsf{X}(A \cup B) & = \{s \in \mathcal{S} \mid \exists s_A \in A \text{ such that } s \to s_A \text{ or } \exists s_B \in B \text{ such that } s \to s_B\} \\
& = \{s \in \mathcal{S} \mid \exists s_A \in A \text{ such that } s \to s_A\} \cup \{s \in \mathcal{S} \mid \exists s_B \in B \text{ such that } s \to s_B\} \\
\exists \mathsf{X}(A \cup B) & = \exists \mathsf{X}(A) \cup \exists \mathsf{X}(B)
\end{array}$$

We can now define the CTL operators as fixed points:

$$\begin{array}{ll}
\forall(\varphi_1 \mathsf{U} \varphi_2) & = \neg(\mu Z.(\neg\varphi_2 \wedge (\neg\varphi_1 \vee \neg\exists \mathsf{X}\top)) \vee (\neg\varphi_2 \wedge \exists \mathsf{X}Z)) \wedge \neg(\nu Z.\neg\varphi_2 \wedge \exists \mathsf{X}Z) \\
& = \neg\Big((\mu Z.(\neg\varphi_2 \wedge (\neg\varphi_1 \vee \neg\exists \mathsf{X}\top)) \vee (\neg\varphi_2 \wedge \exists \mathsf{X}Z)) \vee (\nu Z.\neg\varphi_2 \wedge \exists \mathsf{X}Z)\Big) \\
& = \neg\nu Z.(\neg\varphi_2 \wedge (\neg\varphi_1 \vee \neg\exists \mathsf{X}\top)) \vee (\neg\varphi_2 \wedge \exists \mathsf{X}Z) \\
& = \neg\nu Z.\neg\varphi_2 \wedge (\neg\varphi_1 \vee \exists \mathsf{X}Z \vee \neg\exists \mathsf{X}\top) \\
& = \neg\nu Z.\neg(\varphi_2 \vee (\varphi_1 \wedge \neg\exists \mathsf{X}Z \vee \exists \mathsf{X}\top)) \\
& = \mu Z.\varphi_2 \vee (\varphi_1 \wedge \neg\exists \mathsf{X}\neg Z \wedge \exists \mathsf{X}\top) \\
\forall(\varphi_1 \mathsf{U} \varphi_2) & = \mu Z.\varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X}Z)
\end{array}$$

$$\exists \mathsf{F}\varphi = \exists(\top \mathsf{U} \varphi) = \mu Z.\varphi \vee \exists \mathsf{X}Z$$

$$\forall \mathsf{F}\varphi = \forall(\top \mathsf{U} \varphi) = \mu Z.\varphi \vee \forall \mathsf{X}Z$$

$$\begin{array}{ll}
\exists \mathsf{G}\varphi & = \neg\forall \mathsf{F}\neg\varphi \\
& = \neg\forall(\top \mathsf{U}\neg\varphi) \\
& = \neg\mu Z.\neg\varphi \vee \forall \mathsf{X}Z \\
& = \nu Z.\neg(\neg\varphi \vee \forall \mathsf{X}\neg Z) \\
& = \nu Z.\varphi \wedge \neg\forall \mathsf{X}\neg Z \\
\exists \mathsf{G}\varphi & = \nu Z.\varphi \wedge (\exists \mathsf{X}Z \vee \neg\exists \mathsf{X}\top)
\end{array}$$

$$
\begin{aligned}
\forall \mathsf{G}\varphi &= \neg\exists \mathsf{F}\neg\varphi \\
&= \neg\exists(\top \mathsf{U}\neg\varphi) \\
&= \neg\mu\, Z.\neg\varphi \vee \exists \mathsf{X} Z \\
&= \nu\, Z.\neg(\neg\varphi \vee \exists \mathsf{X}\neg Z) \\
&= \nu\, Z.\varphi \wedge \neg\exists \mathsf{X}\neg Z \\
\forall \mathsf{G}\varphi &= \nu\, Z.\varphi \wedge (\forall \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)
\end{aligned}
$$

$$
\begin{aligned}
\exists\varphi_1 \mathsf{W}\varphi_2 &= \exists\varphi_1 \mathsf{U}\varphi_2 \vee \exists \mathsf{G}\varphi_1 \\
&= \Big(\mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X} Z)\Big) \vee \Big(\nu\, Z.\varphi_1 \wedge (\exists \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)\Big) \\
&= \Big(\mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X} Z)\Big) \vee \Big(\nu\, Z.\varphi_1 \wedge \exists \mathsf{X} Z\Big) \vee \\
&\qquad \Big(\mu\, Z.(\varphi_1 \wedge \exists \mathsf{X} Z) \vee (\varphi_1 \wedge \neg\exists \mathsf{X}\top)\Big) \\
&= \Big(\mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \neg\exists \mathsf{X}\top) \vee (\varphi_1 \wedge \exists \mathsf{X} Z)\Big) \vee \Big(\nu\, Z.\varphi_1 \wedge \exists \mathsf{X} Z\Big) \\
&= \nu\, Z.\varphi_2 \vee (\varphi_1 \wedge \neg\exists \mathsf{X}\top) \vee (\varphi_1 \wedge \exists \mathsf{X} Z) \\
\exists\varphi_1 \mathsf{W}\varphi_2 &= \nu\, Z.\varphi_2 \vee (\varphi_1 \wedge (\neg\exists \mathsf{X}\top \vee \exists \mathsf{X} Z))
\end{aligned}
$$

$$
\begin{aligned}
\forall\varphi_1 \mathsf{W}\varphi_2 &= \neg\exists\neg\varphi_2 \mathsf{U}\neg(\varphi_1 \vee \varphi_2) \\
&= \neg\mu\, Z.\neg(\varphi_1 \vee \varphi_2) \vee (\neg\varphi_2 \wedge \exists \mathsf{X} Z) \\
&= \neg\mu\, Z.\neg(\varphi_1 \vee \varphi_2) \vee (\neg\varphi_2 \wedge \neg\forall \mathsf{X}\neg Z \wedge \exists \mathsf{X}\top) \\
&= \neg\mu\, Z.\neg\big((\varphi_1 \vee \varphi_2) \wedge (\varphi_2 \vee \neg\exists \mathsf{X}\top \vee \forall \mathsf{X}\neg Z)\big) \\
&= \nu\, Z.(\varphi_1 \vee \varphi_2) \wedge (\varphi_2 \vee \neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z) \\
\forall\varphi_1 \mathsf{W}\varphi_2 &= \nu\, Z.\varphi_2 \vee (\varphi_1 \wedge (\neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z))
\end{aligned}
$$

$$
\begin{aligned}
\exists\varphi_1 \mathsf{R}\varphi_2 &= \exists\varphi_2 \mathsf{W}\varphi_1 \wedge \varphi_2 \\
&= \nu\, Z.(\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge (\neg\exists \mathsf{X}\top \vee \exists \mathsf{X} Z)) \\
\exists\varphi_1 \mathsf{R}\varphi_2 &= \nu\, Z.\varphi_2 \wedge (\varphi_1 \vee \neg\exists \mathsf{X}\top \vee \exists \mathsf{X} Z)
\end{aligned}
$$

$$
\begin{aligned}
\forall\varphi_1 \mathsf{R}\varphi_2 &= \forall\varphi_2 \mathsf{W}\varphi_1 \wedge \varphi_2 \\
&= \nu\, Z.(\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge (\neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z)) \\
\forall\varphi_1 \mathsf{R}\varphi_2 &= \nu\, Z.\varphi_2 \wedge (\varphi_1 \vee \neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z)
\end{aligned}
$$

$$
\begin{aligned}
\exists\varphi_1 \mathsf{M}\varphi_2 &= \exists\varphi_2 \mathsf{U}\varphi_1 \wedge \varphi_2 \\
&= \mu\, Z.(\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge \exists \mathsf{X} Z) \\
\exists\varphi_1 \mathsf{M}\varphi_2 &= \mu\, Z.\varphi_2 \wedge (\varphi_1 \vee \exists \mathsf{X} Z)
\end{aligned}
$$

$$
\begin{aligned}
\forall\varphi_1 \mathsf{M}\varphi_2 &= \forall\varphi_2 \mathsf{U}\varphi_1 \wedge \varphi_2 \\
&= \mu\, Z.(\varphi_1 \wedge \varphi_2) \vee (\varphi_2 \wedge \forall \mathsf{X} Z) \\
\forall\varphi_1 \mathsf{M}\varphi_2 &= \mu\, Z.\varphi_2 \wedge (\varphi_1 \vee \forall \mathsf{X} Z)
\end{aligned}
$$

The whole fixed point semantics of CTL on finite and infinite paths is summarised in Table A.1.

Technical remark

Note that Theorem A.2 does not stand if f does not conserve $\cup$, for example $f = \forall \mathsf{X}$.

If you take:

$$\begin{aligned}\forall(\varphi_1 \mathsf{W} \varphi_2) &= \nu\, Z.\varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X} Z) \\ &\neq (\nu\, Z.\varphi_1 \wedge \forall \mathsf{X} Z) \vee (\mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X} Z)) \\ &\neq \forall \mathsf{G} \varphi_1 \vee \forall(\varphi_1 \mathsf{U} \varphi_2)\end{aligned}$$

	$\exists$	$\forall$
X	$\exists \mathsf{X}\varphi = Pred(\varphi)$	$\forall \mathsf{X}\varphi = \exists \mathsf{X}\top \wedge \neg\exists \mathsf{X}\neg\varphi$
U	$\exists(\varphi_1 \mathsf{U} \varphi_2) = \mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \exists \mathsf{X} Z)$	$\forall(\varphi_1 \mathsf{U} \varphi_2) = \mu\, Z.\varphi_2 \vee (\varphi_1 \wedge \forall \mathsf{X} Z)$
W	$\exists(\varphi_1 \mathsf{W} \varphi_2) =$ $\nu\, Z.\varphi_2 \vee (\varphi_1 \wedge (\neg\exists \mathsf{X}\top \vee \exists \mathsf{X} Z))$	$\forall(\varphi_1 \mathsf{W} \varphi_2) =$ $\nu\, Z.\varphi_2 \vee (\varphi_1 \wedge (\neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z))$
R	$\exists(\varphi_1 \mathsf{R} \varphi_2) = \nu\, Z.\varphi_2 \wedge (\varphi_1 \vee \neg\exists \mathsf{X}\top \vee \exists \mathsf{X} Z)$	$\forall(\varphi_1 \mathsf{R} \varphi_2) = \nu\, Z.\varphi_2 \wedge (\varphi_1 \vee \neg\exists \mathsf{X}\top \vee \forall \mathsf{X} Z)$
M	$\exists(\varphi_1 \mathsf{M} \varphi_2) = \mu\, Z.\varphi_2 \wedge (\varphi_1 \vee \exists \mathsf{X} Z)$	$\forall(\varphi_1 \mathsf{M} \varphi_2) = \mu\, Z.\varphi_2 \wedge (\varphi_1 \vee \forall \mathsf{X} Z)$
F	$\exists \mathsf{F}\varphi = \mu\, Z.\varphi \vee \exists \mathsf{X} Z$	$\forall \mathsf{F}\varphi = \mu\, Z.\varphi \vee \forall \mathsf{X} Z$
G	$\exists \mathsf{G}\varphi = \nu\, Z.\varphi \wedge (\exists \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)$	$\forall \mathsf{G}\varphi = \nu\, Z.\varphi \wedge (\forall \mathsf{X} Z \vee \neg\exists \mathsf{X}\top)$

Table A.1: Fixed points definitions of CTL operators on finite and infinite maximal paths.

A.2 Justification of the *Borana model*

A.2.1 Modelling methodology

The *Borana model* is based on the following studies about the vegetation dynamics of the Borana Zone in Ethiopia: [Lia16; LCD18; LC18; Lia+20].

> *"The complexity of rangeland vegetation dynamics can be interpreted by the state-and-transition model, in which rangeland dynamics are described as a set of discrete "states" of vegetation at a specific site and changes between states that occur as discrete "transitions"." [LCD18, p.2]*

> *"Transitions from one state to another often require a combination of climatic circumstances and management actions (e.g., fire or grazing) to bring them about." [Lia+20, p.7]*

The *Borana model* represents the discrete vegetation states by a set of Boolean variables, other Boolean variables called controls also cover management actions and climatic circumstances. The transitions between states are described by *if-then* rules linking a *condition* (on the values of the variables) with a *consequence* (an update of the variables).

> *"Therefore, spatial knowledge of current vegetation states plus understanding of past and future transition pathways is needed to properly prescribe and apply efforts to mitigate undesirable processes such as bush encroachment.*
> *The goal of this study was to provide pastoralists, rangeland managers, and policy makers with a spatial understanding of the past, current, and potential rangeland vegetation states in Borana" [LCD18, p.2]*

In order to foresee the future transition pathways, the *Borana model* is not limited to a description of the observed transition pathways (the STGs available in [LCD18; LC18; Lia+20]). From a set of initial states, the *Borana model* computes every state reachable by the cascading applications of if-then rules. Thus the *Borana model* outputs unobserved transition pathways, assuming that the vegetation dynamics can be deduced from the description of the discrete transitions (i.e. the set of if-then rules). In consequence, this methodology can be used to foresee the effects of new ecosystem management policies, as long as they result into discrete transitions.

A.2.2 Variables

We chose five vegetation variables (`Gr`, `Sh`, `Tr`, `Sa`, `Cr`, see Figure 4.3) to represent the eight vegetation classes (Table 4.1) forming the states of the Borana STMs [LCD18]. We then added three variables representing the presence of grazers or browsers (`Lv`, `Gz`, `Bw`, see Figure 4.3).

> *"While climatic and edaphic factors primarily determine broad-scale vegetation distribution, complex patches of open and closed canopy rangelands can exist within a single climate zone, suggesting that controls such as fire and herbivory are important at a finer spatial scale." [LCD18, p.3]*

> *"Mean annual rainfall ranges from $300mm$ in the lowlands to $1,000mm$ in the highlands. [...] Generally, annual precipitation is positively correlated with elevation." [LCD18, p.2]*

> *"The government prohibited grazing in such forested areas for conservation purposes" [LCD18, p.8]*

> *"Until the 1950s, crop cultivation throughout the Borana Zone was banned by indigenous rules. [...] In recent years, commercial farming has become more prevalent." [LCD18, p.8]*

> *"Adding more goats and camels while reducing the number of cattle in the herds could be crucial. [...] Rather than simply living with bush encroachment, pastoralists can actively contribute to its mitigation by changing their livestock portfolios." [LCD18, p.8-9]*

We defined seven control variables (Figure 4.3), representing climate/altitude (`Alt`), fire ban (`Fb`), crop ban (`Cb`), or herbivory (`Wl`, `Ps`, `Ig`, `BLv`). Controls influence the system but cannot change along the dynamics (see the ruleset Figure 4.4), thus each valuation of the controls represents a specific scenario.

A.2.3 Initial states

> *"Since the 1970s, fire has been banned, leaving livestock grazing as major local-level disturbance factor. The last high-intensity fire set the rangeland state as grassland." [LC18, p.7]*

The initial states represent the grassland vegetation class (grasses are the only present vegetation) after the last high-intensity fire (no animals). Hence the only variable initially valuated on is `Gr+` (Figure 4.3). There is a single initial state per scenario (i.e. control valuation), thus there are $2^7 = 128$ initial states.

A.2.4 Rules

Fire rules: `R1`, `R2`

> *"Since the 1970s, fire has been banned." [LC18, p.7]*

Hence `Fb-` in the condition of `R1`, `R2`.

> *"Low intensity fires would periodically burn grasses, shrubs, and tree saplings on sparse scrubland, but would leave adult trees undamaged." [LC18, p.7]*

Hence Sh−, Sa− in the consequence of R1.

> *"High intensity but low frequency fire could change the landscape into a grass-dominated system." [LC18, p.7]*

Hence Sh−, Tr−, Sa− in the consequence of R2.

> *"Although bush burning ban has been lifted since the 2000s, herbaceous biomass in the understory was minimal, and fuel loads would not build up and could not set the stage for fires to properly thin the woody layer." [LC18, p.7]*

Hence Gr+ in the condition of R1 , R2.

> *"Grasses and other herbs usually established themselves first after fire" [LC18, p.7]*

Hence we did not set Gr− in the consequence of R1 and R2. Thus, as Gr+ is in the condition of R1 and R2, it is still present after their application.

Finally, animals are not mentioned during fire descriptions [LC18; Lia+20], we assumed that they flee. Hence Lv−, Gz−, Bw− in the consequence of R1 and R2.

Trees recruitment: R3

> *"Given favorable environmental conditions, tree seedlings could grow into mature trees and gradually close the canopy." [Lia16, p.43]*

Grass recruitment: R4

> *"Grasses and other herbs usually established themselves first after fire" [LC18, p.7]*

We assumed that grasses established themselves first after any perturbation clearing the vegetation cover.

Closed Canopy Woodland transition: R5

> *"In the highlands [...] given higher precipitation and absence of fire, tree seedlings in dense scrubland could grow into mature trees and gradually close the canopy." [LCD18, p.8]*

Hence Alt+ (*"in the highlands"*), Fb+ (*"absence of fire"*), and Gr- (*"in dense scrubland"*) in condition.

Bushland transition: R6

> *"(Bushland) at the higher end of the (elevation) range is shifting into dense scrubland. [...] In the relatively dry lowlands of the Borana Zone, the primary vegetation transition is from grassland to bushland." [LCD18, p.8]*

We chose to represent this fact by the competitive exclusion of tree saplings Sa by shrubs Sh at low altitude Alt-.

Grazers: R7

We assumed that the presence of wild grazers Gz+ is conditioned by the presence of grasses Gr+ and by the absence of livestock Lv-.

Browsers: R8, R9

We assumed that the presence of wild browsers Bw+ is conditioned by the presence of shrubs Sh+ (R8) or saplings Sa+ (R9), and by the absence of livestock Lv-.

Livestock: R10, R11, R12

We assumed that the presence of livestock Lv+ is conditioned by the presence of grasses Gr+ (R10), or under browsing livestock policies BLv+ by the presence of shrubs Sh+ (R11) or saplings Sa+ (R12). We also assumed that livestock Lv+ excludes both wild grazers Gz- and wild browsers Bw-.

Grazing: R13, R14

> *"With woody plant recruitment and wildlife grazing, the grassland could gradually shift into a sparse scrubland state. [...] Light to moderate grazing reduced understory cover of grassland, and the system shifted into sparse scrubland given plant recruitment. [...] Similarly, on the open canopy woodland, woody plants would also become denser given moderate grazing pressure, gradually shifting the rangeland into the sparse scrubland state." [LC18, p.7]*

Intensive grazing: R15

> *"Heavy grazing on the sparse scrubland could diminish forage in the understory within a short time period, thus leaving the scattered woody plants free from competition." [LC18, p.7]*

> *"As pastoralists sedentarize and herd livestock near and around their settlements in response to external sedentarization initiatives, rangelands can shift into bare ground or shrublands with minimal grazing value." [Lia+20, p.2]*

Browsing: R16, R17

> *"Wildlife browsing could keep the re-sprouting woody species in check." [LC18, p.7]*

> *"Increasing browsing pressure by goats and camels can thin the woody plant layer and suppress the growth of shrubs and trees, which can indirectly facilitate the growth of herbs on the ground." [LCD18, p.8]*

We chose to represent this fact by enabling browsers (wild Bw+ or domestic BLv+ $\wedge$ Lv+) to remove shrubs Sh− and saplings Sa−. Grasses Gr+ are supposed to establish themselves first in the cleared space.

Crops: R18, R19

> *"Dense scrublands, along with other minor classes such as closed and open canopy woodlands, that are situated at above $1,200m$ are being converted to cultivated areas, which allows the practice of rain-fed agriculture." [LCD18, p.8]*

> *"Cropland expansion [...] accelerate rangeland degradation and wildlife habitat loss, and discourage mobile livestock herding" [Lia+20, p.2]*

We chose to represent dense scrublands, closed and open canopy woodland by the presence of trees (see Table 4.1). Hence the condition of R18: high altitude Alt+, crops being allowed Cb− and trees Tr+. The consequence of R18 includes the replacement of grasses Gr−, shrubs Sh− and saplings Sa− by cultivated species Cr+, and the disappearance of both livestock Lv− and wild life Gz−, Bw−.

> *"During the 2003–2013 decade, 355km^2 of cropland transitioned backed to dense scrublands and $124km^2$ back to open canopy woodlands." [LCD18, p.7]*

We assumed grasses establish themselves first after crops are abandoned (R19).

A.2.5 Improvements

Here we list some worth considering improvements, unfortunately unavailable data would be required in order to implement them:

- Add a variable representing bare soil. Indeed the definition of the vegetation classes [LCD18, Tab.1] is based on vegetation cover, which does not always sum up to 100%. Thus the percentage of bare soil is a part of the vegetation classes description. Nevertheless, bare soil is almost never mentioned in the sources, so we chose to not include it in the variables.
- Represent species instead of plant functional types. Indeed the vegetation classes [LCD18, Tab.1] do not encompass the same species of grasses, shrubs nor trees. Unfortunately, data on the species dynamics is lacking for the Borana zone.
- Desynchronise shrubs and saplings browsing. The browsing rules (`R16`, `R17`) remove saplings and shrubs simultaneously, but it may not be the case in reality (because of foraging preferences for example). Without more precise data, we chose to synchronise both removals.

A.3 Proof of the symbolic FARCTL algorithm

In this section, we provide in-depth proofs of the fixed points definitions of FARCTL operators $\exists_\alpha^{\mathcal{F}_S}\mathsf{G}(\varphi)$ and $\exists_\alpha^{\mathcal{F}_W}\mathsf{G}(\varphi)$ given in Lemma 5.1, Theorem 5.1 and Theorem 5.2.

A.3.1 Detailed proof of Lemma 5.1 and Theorem 5.1

Recall the following definitions used in Lemma 5.1 and Theorem 5.1:

Definition 5.6 ($\overset{\infty}{\exists}, \overset{\infty}{\forall}$)**.** Let π be a maximal path and $e_{\mathcal{S}} \in ARCTL_{\mathcal{S}}$ a state-event:

- $\pi \models \overset{\infty}{\exists}\, e_{\mathcal{S}}$ iff $\forall i \in \mathbb{N} \quad i \leqslant |\pi| \Rightarrow \exists j \in \mathbb{N}$ such that $i \leqslant j \leqslant |\pi|$ and $\pi[j]_{\mathcal{S}} \models e_{\mathcal{S}}$
- $\pi \models \overset{\infty}{\forall}\, e_{\mathcal{S}}$ iff $\exists i \in \mathbb{N}$ such that $i \leqslant |\pi|$ and $\forall j \in \mathbb{N} \quad i \leqslant j \leqslant |\pi| \Rightarrow \pi[j]_{\mathcal{S}} \models e_{\mathcal{S}}$

Let π be a maximal path and $e_{\mathcal{A}} \in ARCTL_{\mathcal{A}}$ an action-event:

- $\pi \models \overset{\infty}{\exists}\, e_{\mathcal{A}}$ iff $|\pi| = \infty$ and $\forall i \in \mathbb{N} \quad \exists j \geqslant i$ such that $\pi[j]_{\mathcal{A}} \models e_{\mathcal{A}}$
- $\pi \models \overset{\infty}{\forall}\, e_{\mathcal{A}}$ iff $|\pi| = \infty$ and $\exists i \in \mathbb{N}$ such that $\forall j \geqslant i \quad \pi[j]_{\mathcal{A}} \models e_{\mathcal{A}}$

Definition 5.15. Let $e_{\mathcal{S}} \in ARCTL_{\mathcal{S}}$ a state-event, $e_{\mathcal{A}} \in ARCTL_{\mathcal{A}}$ an action-event, and $Z \subseteq \mathcal{S}$:

- $\exists_\alpha \vec{\mathsf{x}}_{\models e_{\mathcal{S}}}(Z) = e_{\mathcal{S}} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top)$
- $\exists_\alpha \vec{\mathsf{x}}_{\not\models e_{\mathcal{S}}}(Z) = \neg e_{\mathcal{S}} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top)$
- $\exists_\alpha \vec{\mathsf{x}}_{\models e_{\mathcal{A}}}(Z) = \exists_{\alpha \wedge e_{\mathcal{A}}} \mathsf{X} Z$
- $\exists_\alpha \vec{\mathsf{x}}_{\not\models e_{\mathcal{A}}}(Z) = \exists_{\alpha \wedge \neg e_{\mathcal{A}}} \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top$

Definition 5.16 ($\tau_\alpha^{\mathcal{F}_S}$)**.** Let $\mathcal{F}_S = \{\mathcal{F}_S(e_1, e_2) \mid e_1 \in ARCTL_{\mathcal{S}}\}$ a fairness assumption composed exclusively of strong fairness constraints whose first events are state-events $e_1 \in ARCTL_{\mathcal{S}}$. We define $\tau_\alpha^{\mathcal{F}_S}$ as:

$$\tau_\alpha^{\mathcal{F}_S}(Z) = \bigwedge_{\mathcal{F}_S(e_1,e_2) \in \mathcal{F}_S} \left(\exists_\alpha \vec{\mathsf{x}}_{\not\models e_1}(Z) \vee \exists_\alpha (Z \mathsf{U} Z \wedge \exists_\alpha \vec{\mathsf{x}}_{\models e_2}(Z)) \right)$$

Lemma 5.1. $\left(s \in \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))\right) \Rightarrow \left(s \models \exists_\alpha^{\mathcal{F}_S}\mathsf{G}(\varphi)\right)$

Proof. The proof follows the same structure as the proof of the main text, but both: (1) details the cases depending on whether e_2 are state-events or action-events, and (2) uses the formal semantics of the strong fairness instead of its intuitive one.

Let $S' = \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))$. For every state $s \in S'$, we have $s \models \varphi$ and $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$ at least one of the following is true:

1. $s \models \exists_\alpha \vec{\mathsf{x}}_{\not\models e_1}(S')$, i.e. because e_1 is a state-event we have: $s \models \neg e_{\mathcal{S}1} \wedge (\exists_\alpha \mathsf{X} S' \vee \neg \exists_\alpha \mathsf{X} \top)$. Thus either s is an α-dead-end not satisfying e_1, or s does not satisfy e_1 and has an α-successor in S'. Thus when building a maximal path π within

S', one can pass by s and either be trapped in a dead-end without e_1 happening (thus $\pi \not\models^{\infty}_{\exists} e_1$ and consequently π is fair), or extend π in S' without e_1 happening.

2. $s \models \exists_\alpha(S' \mathsf{U} S' \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(S'))$, meaning that there is a state $s' \in S'$ which is α-reachable from s within S' such that $s' \models \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(S')$. There are two cases, depending on whether e_2 is a state-event or an action-event:
   2.i. If $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event. Then $s' \models e_{\mathcal{S}2} \wedge (\exists_\alpha \mathsf{X} S' \vee \neg \exists_\alpha \mathsf{X} \top)$. Thus either s' is an α-dead-end satisfying e_2, or s' satisfies e_2 and has an α-successor in S'.
   2.ii. If $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event. Then $s' \models \exists_{\alpha \wedge e_2} \mathsf{X} S'$. Thus s' has an $(\alpha \wedge e_2)$-successor in S'.

   Thus if $s \models \exists_\alpha(S' \mathsf{U} S' \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(S'))$, then when building a maximal path π within S', one can pass by s and extend π within S' to reach a state s' that either is an α-dead-end with e_2 happening (thus $\pi \models^{\infty}_{\exists} e_2$ and consequently π is fair), or where π can be further extended with e_2 happening.

Thus from any $s \in S'$, one can build a maximal path π within S' satisfying continuously φ and either ending in a dead-end satisfying every strong fairness constraint, or infinitely carrying on while $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, if e_1 happens then it is eventually followed by e_2. If several distinct e_1 happens infinitely often along the construction of π, then we can extend π to reach infinitely often every associated e_2 by alternating between them. Thus if e_1 happens infinitely often along π then e_2 happens infinitely often as well: $\left(\pi \models^{\infty}_{\exists} e_1\right) \Rightarrow \left(\pi \models^{\infty}_{\exists} e_2\right)$. Hence $s \models \exists^{\mathcal{F}_S}_\alpha \mathsf{G}(\varphi)$. □

Theorem 5.1. $\exists^{\mathcal{F}_S}_\alpha \mathsf{G}(\varphi) = \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau^{\mathcal{F}_S}_\alpha(Z)))$

Proof. The proof follows the same structure as the proof of the main text, but both: (1) details the cases depending on whether e_2 are state-events or action-events, and (2) uses the formal semantics of the strong fairness instead of its intuitive one.

The proof of $\exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau^{\mathcal{F}_S}_\alpha(Z))) \subseteq \exists^{\mathcal{F}_S}_\alpha \mathsf{G}(\varphi)$ is the same as in the main text.

Let us prove that $\exists^{\mathcal{F}_S}_\alpha \mathsf{G}(\varphi) \subseteq \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau^{\mathcal{F}_S}_\alpha(Z)))$. Let $s \in \exists^{\mathcal{F}_S}_\alpha \mathsf{G}(\varphi)$ and take $\pi \in \Pi|^{\mathcal{F}_S}_\alpha(s)$ such that $\pi \models \mathsf{G}\varphi$. We will use the fact that $\nu\, Z.\tau(Z) \stackrel{\text{def}}{=} \cup\{S \subseteq \mathcal{S} \mid S \subseteq \tau(S)\}$ [BS07a] to show that $s \in \exists_\alpha(\varphi \mathsf{U} \nu\, Z.(\varphi \wedge \tau^{\mathcal{F}_S}_\alpha(Z)))$. Indeed $\tau^{\mathcal{F}_S}_\alpha$ is monotonic (i.e. if $S \subseteq S'$ then $\tau^{\mathcal{F}_S}_\alpha(S) \subseteq \tau^{\mathcal{F}_S}_\alpha(S')$) as one can see when instantiating $\tau^{\mathcal{F}_S}_\alpha$:

$$\tau^{\mathcal{F}_S(e_{\mathcal{S}1}, e_{\mathcal{S}2})}_\alpha(Z) = (\neg e_{\mathcal{S}1} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top)) \vee \exists_\alpha(Z \mathsf{U} Z \wedge e_{\mathcal{S}2} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top))$$
$$\tau^{\mathcal{F}_S(e_{\mathcal{S}1}, e_{\mathcal{A}2})}_\alpha(Z) = (\neg e_{\mathcal{S}1} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top)) \vee \exists_\alpha(Z \mathsf{U} Z \wedge \exists_{\alpha \wedge e_{\mathcal{A}2}} \mathsf{X} Z)$$

There are two cases, depending on whether π is finite or infinite:

1. If $|\pi| \in \mathbb{N}$, then take $s' = \pi[|\pi|]_{\mathcal{S}}$ the α-dead-end ending π. We will prove that $\{s'\} \subseteq \varphi \wedge \tau^{\mathcal{F}_S}_\alpha(\{s'\})$. Indeed $s' \in \varphi$ because $\pi \models \mathsf{G}\varphi$. Let us prove that $s' \in \tau^{\mathcal{F}_S}_\alpha(\{s'\})$. For all $\mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$ we will prove that $s' \in \exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(\{s'\}) \vee \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(\{s'\}) \subseteq \exists_\alpha \vec{\mathsf{X}}_{\not\models e_1}(\{s'\}) \vee \exists_\alpha(Z \mathsf{U} Z \wedge \exists_\alpha \vec{\mathsf{X}}_{\models e_2}(\{s'\}))$ because s' is an α-dead-

end (i.e. $s' \models \neg\exists_\alpha \mathsf{X}\top$) and $\pi \models \mathcal{F}_S(e_1, e_2) = \left(\pi \not\models^{\infty}_{\exists} e_1\right) \vee \left(\pi \models^{\infty}_{\exists} e_2\right)$. There are two cases, depending on whether e_2 are state-events or action-events:

1.i. If $e_1, e_2 \in ARCTL_{\mathcal{S}}$. If $\pi \models^{\infty}_{\exists} e_2$ then $s' \models e_2$ and thus $s' \models \exists_\alpha {}^{\vec{\mathsf{x}}}_{\models e_2}(\{s'\}) = e_2 \wedge (\exists_\alpha \mathsf{X}\{s'\} \vee \neg\exists_\alpha \mathsf{X}\top)$. Conversely if $\pi \not\models^{\infty}_{\exists} e_1$ then $s' \not\models e_1$ and thus $s' \models \exists_\alpha {}^{\vec{\mathsf{x}}}_{\not\models e_1}(\{s'\}) = \neg e_1 \wedge (\exists_\alpha \mathsf{X}\{s'\} \vee \neg\exists_\alpha \mathsf{X}\top)$.

1.ii. If $e_1 \in ARCTL_{\mathcal{S}}$ and $e_2 \in ARCTL_{\mathcal{A}}$. Then $\pi \not\models^{\infty}_{\exists} e_2$ because $|\pi| \neq \infty$. Thus $\pi \not\models^{\infty}_{\exists} e_1$ meaning that $s' \not\models e_1$. Thus $s' \models \exists_\alpha {}^{\vec{\mathsf{x}}}_{\not\models e_1}(\{s'\}) = \neg e_1 \wedge (\exists_\alpha \mathsf{X}\{s'\} \vee \neg\exists_\alpha \mathsf{X}\top)$..

Thus $s' \in\subseteq \tau_\alpha^{\mathcal{F}_S}(\{s'\})$. Then $s' \in \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))$, and finally $s \in \exists_\alpha(\varphi \mathsf{U} \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$.

2. If $|\pi| = \infty$. Let $i \in \mathbb{N}$ be the minimal index such that: $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$ if $\pi \not\models^{\infty}_{\exists} e_2$, then $\forall j \geqslant i, \pi[j] \not\models e_1$. Such i exists because $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, $\left(\pi \not\models^{\infty}_{\exists} e_1\right) \vee \left(\pi \models^{\infty}_{\exists} e_2\right)$, meaning that if e_2 happens only finitely often along π, then e_1 happens only finitely often along π as well. Take $\pi[i \ldots]$ the infinite suffix of π starting from its i-th state (i.e. $\pi[i]_{\mathcal{S}} \xrightarrow{\pi[i+1]_{\mathcal{A}}} \pi[i+1]_{\mathcal{S}} \xrightarrow{\pi[i+2]_{\mathcal{A}}} \ldots$), that is the suffix of π such that $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, if e_2 does not happen infinitely often then e_1 does not happen at all. Take $\pi[i \ldots]_{\mathcal{S}}$ the states of this suffix, let us prove that $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}}(\pi[i \ldots]_{\mathcal{S}})$. $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi$ because $\pi \models \mathsf{G}\varphi$. $\forall j \geqslant i$ and $\forall \mathcal{F}_S(e_1, e_2) \in \mathcal{F}_S$, let us prove that $\pi[j]_{\mathcal{S}} \in \exists_\alpha {}^{\vec{\mathsf{x}}}_{\not\models e_1}(\pi[i \ldots]_{\mathcal{S}}) \vee \exists_\alpha(Z \mathsf{U} Z \wedge \exists_\alpha {}^{\vec{\mathsf{x}}}_{\models e_2}(\pi[i \ldots]_{\mathcal{S}}))$. There are two cases, depending on whether e_2 are state-events or action-events:

2.i. If $e_1, e_2 \in ARCTL_{\mathcal{S}}$. If $\pi[j]_{\mathcal{S}} \models e_1$ then $\exists k > j$ such that $\pi[k]_{\mathcal{S}} \models e_2$, and thus $\pi[j]_{\mathcal{S}} \models \exists_\alpha(\pi[i \ldots]_{\mathcal{S}} \mathsf{U} \pi[i \ldots]_{\mathcal{S}} \wedge e_2 \wedge (\exists_\alpha \mathsf{X}(\pi[i \ldots]_{\mathcal{S}}) \vee \neg\exists_\alpha \mathsf{X}\top))$. Otherwise $\pi[j]_{\mathcal{S}} \not\models e_1$, and thus $\pi[j]_{\mathcal{S}} \in \neg e_1 \wedge (\exists_\alpha \mathsf{X}(\pi[i \ldots]_{\mathcal{S}}) \vee \neg\exists_\alpha \mathsf{X}\top)$.

2.ii. If $e_1 \in ARCTL_{\mathcal{S}}$ and $e_2 \in ARCTL_{\mathcal{A}}$. If $\pi[j]_{\mathcal{S}} \models e_1$ then $\exists k > j$ such that $\pi[k]_{\mathcal{A}} \models e_2$, thus $\pi[j]_{\mathcal{S}} \models \exists_\alpha(\pi[i \ldots]_{\mathcal{S}} \mathsf{U} \pi[i \ldots]_{\mathcal{S}} \wedge \exists_{\alpha \wedge e_2} \mathsf{X}(\pi[i \ldots]_{\mathcal{S}}))$. Otherwise $\pi[j]_{\mathcal{S}} \not\models e_1$, and thus $\pi[j]_{\mathcal{S}} \in \neg e_1 \wedge (\exists_\alpha \mathsf{X}(\pi[i \ldots]_{\mathcal{S}}) \vee \neg\exists_\alpha \mathsf{X}\top)$.

Thus $\pi[i \ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_S}(\pi[i \ldots]_{\mathcal{S}})$. Then $\pi[i \ldots]_{\mathcal{S}} \in \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z))$, and finally $s \in \exists_\alpha(\varphi \mathsf{U} \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_S}(Z)))$.

□

A.3.2 Detailed proof of Theorem 5.2

Recall the following definitions used in Theorem 5.2:

Lemma 5.3. Let π a maximal path, $e_{\mathcal{S}} \in ARCTL_{\mathcal{S}}$ a state-event, and $e_{\mathcal{A}} \in ARCTL_{\mathcal{A}}$ an action-event:

$$\pi \not\models^{\infty}_{\forall} e_{\mathcal{S}} \text{ iff } \pi \models^{\infty}_{\exists} \neg e_{\mathcal{S}}$$
$$\pi \not\models^{\infty}_{\forall} e_{\mathcal{A}} \text{ iff } |\pi| \neq \infty \text{ or } \pi \models^{\infty}_{\exists} \neg e_{\mathcal{A}}$$

Corollary 5.3. Let π a maximal path, and e_2 a state-event or an action-event:

- Let e_1 a state-event, $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(\pi \models^{\infty}_{\exists} \neg e_1 \text{ or } \pi \models^{\infty}_{\exists} e_2\right)$
- Let e_1 an action-event, $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(|\pi| \neq \infty \text{ or } \pi \models^{\infty}_{\exists} \neg e_1 \text{ or } \pi \models^{\infty}_{\exists} e_2\right)$

Theorem 5.2. Let $\mathcal{F}_W$ a fairness assumption composed exclusively of weak fairness constraints, we define $\tau_\alpha^{\mathcal{F}_W}$ as:

$$\tau_\alpha^{\mathcal{F}_W}(Z) = \bigwedge_{\mathcal{F}_W(e_1,e_2)\in\mathcal{F}} \left(\exists_\alpha(Z \mathsf{U} Z \wedge (\exists_\alpha {}^{\vec{\mathrm{x}}}_{\not\models e_1}(Z) \vee \exists_\alpha {}^{\vec{\mathrm{x}}}_{\models e_2}(Z)))\right)$$

Then: $\exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi) = \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$

Proof. Let us prove that $\nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z)) \subseteq \exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi)$. Let $S' = \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$. For every state $s \in S'$, we have $s \models \varphi$ and $\forall \mathcal{F}_W(e_1, e_2) \in \mathcal{F}_W$ we have $s \models \exists_\alpha(S' \mathsf{U} S' \wedge (\exists_\alpha {}^{\vec{\mathrm{x}}}_{\not\models e_1}(S') \vee \exists_\alpha {}^{\vec{\mathrm{x}}}_{\models e_2}(S')))$. Thus from s there is a finite path π in S' of length $|\pi| = i \in \mathbb{N}$ ending in a state $\pi[i]_{\mathcal{S}} \in S'$ such that $\pi[i]_{\mathcal{S}} \models (\exists_\alpha {}^{\vec{\mathrm{x}}}_{\not\models e_1}(S') \vee \exists_\alpha {}^{\vec{\mathrm{x}}}_{\models e_2}(S'))$. Thus if $\pi[i]_{\mathcal{S}} \models \exists_\alpha {}^{\vec{\mathrm{x}}}_{\not\models e_1}(S')$, there are two cases:

1. If $e_1 \in ARCTL_{\mathcal{S}}$ is a state-event. Then $\pi[i]_{\mathcal{S}} \models \neg e_1 \wedge (\exists_\alpha \mathsf{X} S' \vee \neg \exists_\alpha \mathsf{X} \top)$, meaning that $\pi[i]_{\mathcal{S}} \not\models e_1$ and either $\pi[i]_{\mathcal{S}}$ is an α-dead-end or π can be α-extended within S'.
2. If $e_1 \in ARCTL_{\mathcal{A}}$ is an action-event. Then $\pi[i]_{\mathcal{S}} \models \exists_{\alpha \wedge \neg e_1} \mathsf{X} S' \vee \neg \exists_\alpha \mathsf{X} \top$, meaning that either $\pi[i]_{\mathcal{S}}$ is an α-dead-end or π can be α-extended within S' such that $\pi[i+1]_{\mathcal{A}} \not\models e_1$.

Likewise if $\pi[i]_{\mathcal{S}} \models \exists_\alpha {}^{\vec{\mathrm{x}}}_{\models e_2}(S')$, there are also two cases:

1. If $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event. Then $\pi[i]_{\mathcal{S}} \models e_1 \wedge (\exists_\alpha \mathsf{X} S' \vee \neg \exists_\alpha \mathsf{X} \top)$, meaning that $\pi[i]_{\mathcal{S}} \models e_1$ and either $\pi[i]_{\mathcal{S}}$ is an α-dead-end or π can be α-extended within S'.
2. If $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event. Then $\pi[i]_{\mathcal{S}} \models \exists_{\alpha \wedge e_2} \mathsf{X} S'$, meaning that π can be α-extended within S' such that $\pi[i+1]_{\mathcal{A}} \models e_2$.

Let us prove that from $s \in S'$, we can build a maximal path $\pi \in \Pi|_\alpha(s)$ such that $\forall \mathcal{F}_W(e_1, e_2) \in \mathcal{F}_W$ we have $\pi \models \mathcal{F}_W(e_1, e_2)$. There are two cases:

1. If $e_1 \in ARCTL_{\mathcal{S}}$ is a state-event. Recall Corollary 5.3: $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(\pi \models^{\infty}_{\exists} \neg e_1 \text{ or } \pi \models^{\infty}_{\exists} e_2\right)$. At least one of the following is true:
   - 1.i. $\pi[i]_{\mathcal{S}} \models e_1$ and $\pi[i]_{\mathcal{S}}$ is an α-dead-end, thus $\pi \models^{\infty}_{\exists} \neg e_1$.
   - 1.ii. $\pi[i]_{\mathcal{S}} \models e_1$ and π can be α-extended within S'.
   - 1.iii. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and $\pi[i]_{\mathcal{S}}$ is an α-dead-end thus $\pi \models^{\infty}_{\exists} e_2$.
   - 1.iv. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and π can be α-extended within S'.
   - 1.v. $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event and π can be α-extended within S' such that $\pi[i+1]_{\mathcal{A}} \models e_2$.

   Thus by induction over s either π ends in an α-dead-end satisfying $\mathcal{F}_W(e_1, e_2)$,

or can be infinitely α-extended within S' while satisfying at least one of (e_1, e_2) infinitely often. Thus $\pi \models \mathcal{F}_W(e_1, e_2)$.

2. If $e_1 \in ARCTL_{\mathcal{A}}$ is an action-event. Recall Corollary 5.3: $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(|\pi| \neq \infty \text{ or } \pi \models^{\infty}_{\exists} \neg e_1 \text{ or } \pi \models^{\infty}_{\exists} e_2\right)$. At least one of the following is true:
   2.i. $\pi[i]_{\mathcal{S}}$ is an α-dead-end.
   2.ii. π can be α-extended within S' such that $\pi[i+1]_{\mathcal{A}} \not\models e_1$.
   2.iii. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and $\pi[i]_{\mathcal{S}}$ is an α-dead-end thus $\pi \models^{\infty}_{\exists} e_2$.
   2.iv. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and π can be α-extended within S'.
   2.v. $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event and π can be α-extended within S' such that $\pi[i+1]_{\mathcal{A}} \models e_2$.
   Thus by induction over s either π ends in an α-dead-end (i.e. $|\pi| \neq \infty$), or can be infinitely α-extended within S' while satisfying at least one of (e_1, e_2) infinitely often. Thus $\pi \models \mathcal{F}_W(e_1, e_2)$.

To conclude $\forall s \in \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))\ \exists \pi \in \Pi|_\alpha(s)$ such that $\pi \models \exists_\alpha \mathsf{G}\varphi$ and $\forall \mathcal{F}_W(e_1, e_2) \in \mathcal{F}_W$ we have $\pi \models \mathcal{F}_W(e_1, e_2)$. Thus $\nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z)) \subseteq \exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi)$.

Now let us prove that $\exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi) \subseteq \nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$. Let $s \in \exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi)$, thus $\exists \pi \in \Pi|_\alpha(s)$ such that $\pi \models \mathsf{G}\varphi$ and $\forall \mathcal{F}_W(e_1, e_2) \in \mathcal{F}_W$ we have $\pi \models \mathcal{F}_W(e_1, e_2)$. Let us prove that the states of π, noted $\pi[\ldots]_{\mathcal{S}}$, are in $\nu Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(\pi[\ldots]_{\mathcal{S}}))$. We will use the fact that $\nu Z.\tau(Z) \stackrel{\text{def}}{=} \cup\{S \subseteq \mathcal{S} | S \subseteq \tau(S)\}$ [BS07a]. Indeed $\tau_\alpha^{\mathcal{F}_W}$ is monotonic (i.e. if $S \subseteq S'$ then $\tau_\alpha^{\mathcal{F}_W}(S) \subseteq \tau_\alpha^{\mathcal{F}_W}(S')$) as one can see when instantiating $\tau_\alpha^{\mathcal{F}_W}$:

$$\tau_\alpha^{\mathcal{F}_W(e_{\mathcal{S}1}, e_{\mathcal{S}2})}(Z) = \exists_\alpha(Z \mathsf{U} Z \wedge (\neg e_{\mathcal{S}1} \vee \neg e_{\mathcal{S}2}) \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top))$$
$$\tau_\alpha^{\mathcal{F}_W(e_{\mathcal{A}1}, e_{\mathcal{S}2})}(Z) = \exists_\alpha(Z \mathsf{U} Z \wedge (\exists_{\alpha \wedge \neg e_{\mathcal{A}1}} \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top \vee (e_{\mathcal{S}2} \wedge \exists_\alpha \mathsf{X} Z)))$$
$$\tau_\alpha^{\mathcal{F}_W(e_{\mathcal{S}1}, e_{\mathcal{A}2})}(Z) = \exists_\alpha(Z \mathsf{U} Z \wedge ((\neg e_{\mathcal{S}1} \wedge (\exists_\alpha \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top)) \vee \exists_{\alpha \wedge e_{\mathcal{A}2}} \mathsf{X} Z))$$
$$\tau_\alpha^{\mathcal{F}_W(e_{\mathcal{A}1}, e_{\mathcal{A}2})}(Z) = \exists_\alpha(Z \mathsf{U} Z \wedge (\exists_{\alpha \wedge \neg e_{\mathcal{A}1}} \mathsf{X} Z \vee \neg \exists_\alpha \mathsf{X} \top \vee \exists_{\alpha \wedge e_{\mathcal{A}2}} \mathsf{X} Z))$$

$\pi[\ldots]_{\mathcal{S}} \subseteq \varphi$ because $\pi \models \mathsf{G}\varphi$. $\forall \mathcal{F}_W(e_1, e_2) \in \mathcal{F}_W$ and $\forall s' \in \pi[\ldots]_{\mathcal{S}}$, there are two cases:

1. If $e_1 \in ARCTL_{\mathcal{S}}$ is a state-event. Recall Corollary 5.3: $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\left(\pi \models^{\infty}_{\exists} \neg e_1 \text{ or } \pi \models^{\infty}_{\exists} e_2\right)$. At least one of the following is true:
   1.i. $\pi \models^{\infty}_{\exists} \neg e_1$, thus either π ends in an α-dead-end satisfying $\neg e_1$ meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \neg e_1 \wedge \neg \exists_\alpha \mathsf{X} \top)$, or $\neg e_1$ happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \neg e_1 \wedge \exists_\alpha \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.
   1.ii. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and $\pi \models^{\infty}_{\exists} e_2$. Thus either π ends in an α-dead-end satisfying e_2 meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge e_2 \wedge \neg \exists_\alpha \mathsf{X} \top)$, or e_2 happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge e_2 \wedge \exists_\alpha \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.
   1.iii. $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event and $\pi \models^{\infty}_{\exists} e_2$. Thus e_2 happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \exists_{\alpha \wedge e_2} \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.
   Thus by induction over s', we have $\pi[\ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_W}(\pi[\ldots]_{\mathcal{S}})$.

2. If $e_1 \in ARCTL_{\mathcal{A}}$ is an action-event. Recall Corollary 5.3: $\pi \models \mathcal{F}_W(e_1, e_2)$ iff $\Big(|\pi| \neq \infty$ or $\pi \models^{\infty}_{\exists} \neg e_1$ or $\pi \models^{\infty}_{\exists} e_2\Big)$. At least one of the following is true:
   - 2.i. $|\pi| \neq \infty$, thus $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \neg\exists_\alpha \mathsf{X}\top)$
   - 2.ii. $\pi \models^{\infty}_{\exists} \neg e_1$, thus $\neg e_1$ happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \exists_{\alpha\wedge\neg e_1} \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.
   - 2.iii. $e_2 \in ARCTL_{\mathcal{S}}$ is a state-event and $\pi \models^{\infty}_{\exists} e_2$. Thus either π ends in an α-dead-end satisfying e_2 meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge e_2 \wedge \neg\exists_\alpha \mathsf{X}\top)$, or e_2 happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge e_2 \wedge \exists_\alpha \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.
   - 2.iv. $e_2 \in ARCTL_{\mathcal{A}}$ is an action-event and $\pi \models^{\infty}_{\exists} e_2$. Thus e_2 happens infinitely often along π meaning that $s' \models \exists_\alpha(\pi[\ldots]_{\mathcal{S}} \mathsf{U} \pi[\ldots]_{\mathcal{S}} \wedge \exists_{\alpha\wedge e_2} \mathsf{X} \pi[\ldots]_{\mathcal{S}})$.

   Thus by induction over s', we have $\pi[\ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_W}(\pi[\ldots]_{\mathcal{S}})$.

To conclude $\forall s \in \exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi)$, we have $\exists \pi \in \Pi|_\alpha(s)$ such that $s \in \pi[\ldots]_{\mathcal{S}} \subseteq \varphi \wedge \tau_\alpha^{\mathcal{F}_W}(\pi[\ldots]_{\mathcal{S}})$. Thus $s \in \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$ and $\exists_\alpha^{\mathcal{F}_W} \mathsf{G}(\varphi) \subseteq \nu\, Z.(\varphi \wedge \tau_\alpha^{\mathcal{F}_W}(Z))$. □

Remerciements

Vous savez, moi je ne crois pas qu'il y ait de bonne ou de mauvaise situation. Si je devais résumer ma vie aujourd'hui avec vous, je dirais que c'est d'abord des rencontres. Des gens qui m'ont tendu la main. Et c'est assez curieux de se dire que les hasards, les rencontres, forgent une destinée… Parce que quand on a le goût de la chose bien faite, le beau geste, parfois on ne trouve pas l'interlocuteur en face, le miroir qui vous aide à avancer. Alors ça n'est pas mon cas, puisque moi au contraire j'ai pu. Et je dis merci à la vie, je lui dis merci, je chante la vie, je danse la vie… Je ne suis qu'amour ! Et finalement, quand beaucoup de gens aujourd'hui me disent "Mais comment fais-tu pour avoir cette humanité ?", et bien je leur réponds très simplement, je leur dis que c'est ce goût de l'amour. Ce goût donc qui m'a poussé aujourd'hui à entreprendre une thèse, mais demain qui sait ? Peut-être simplement à me mettre au service de la communauté, à faire le don, le don de soi…

Pas d'encadrement, pas de thèse. Merci Cédric de m'avoir permis de faire une thèse à cheval entre informatique théorique et écologie, ce genre de sujet ne court pas les rues et cela n'aurait pas été possible sans ton ouverture d'esprit. Merci beaucoup Franck pour ton aide et pour tes mots toujours justes pour prendre le recul nécessaire et dédramatiser les situations. Merci d'avoir combiné rigueur scientifique et curiosité intellectuelle, vision d'ensemble et adaptation à mes envies, formalisation et implémentation, encadrement et bienveillance… Merci d'avoir été un soutien et une source d'inspiration. J'ai énormément apprécié ces trois années passées ensemble, que ce soit humainement ou scientifiquement, et j'espère pouvoir retrouver un jour un environnement de travail aussi compréhensif.

Pas d'évaluation par les pairs, pas de science. Merci aux membres du jury, Hanna Klaudel, Christine Largouët, Yann Thierry-Mieg, Isabelle Boulangeat, Serenella Ceritto et Loïc Paulevé, d'avoir accepté de lire et d'évaluer mon travail. Merci pour les conversations passionnantes que nous avons eues lors de la soutenance.

Pas de passé, pas de présent. Merci à Chuan Liao de m'avoir autorisé à me baser sur son travail pour le cas d'étude du Borana, et à adapter ses figures pour illustrer ma thèse. Sans ses talents graphiques, mon manuscrit aurait été bien plus aride.

Pas de Mathieu, pas de protistes. Merci à Mathieu de Goër de Herve d'avoir cru en ma suggestion que les protistes feraient un bon sujet de post-doctorat. Ton travail m'a fait changer de perspective sur la forme que devait avoir un graphe état-transition et sur comment on pouvait l'analyser. Merci de m'avoir autorisé à utiliser tes recherches pour les cas d'études de ma thèse. Merci aussi à Richard Law, Philip Warren et Anita Weatherby pour les conversations enrichissantes et pour nous avoir fourni leurs données.

Pas de Maximilien, pas de dialogue. Un immense merci à Maximilien pour cette double thèse en tandem. Sans toutes nos discussions je n'aurais sûrement pas fait la moitié de

ce que j'ai fait. Malheureusement ce manuscrit ne reflète pas assez à mon goût l'immense impact que tu as eu. Merci pour ton enthousiasme, pour ton ouverture d'esprit et pour ton investissement. C'était vraiment un plaisir de partager un bureau avec toi pendant ces années.

Pas de stagiaires, pas d'ecco. Merci à tous les stagiaires qui ont fait partie de l'équipe pendant ces trois années: Amandine, Juliette, Maxime, Olivia, et Santiago. Le retour des utilisateur·ice·s c'est le plus important quand on développe un outil, ecco est en très grande partie issu de vos idées (désolé que vos demandes aient souvent été comblées juste avant votre départ). Merci surtout pour les sorties à la rivière et les bières après le travail !

Pas de labo, pas de bureau. Merci à l'AMAP et à l'IBISC pour leur accueil, j'ai eu la chance de tomber dans deux laboratoires très sympathiques, bien loin des atmosphères de compétition dont on entend parler ailleurs. Merci à Thierry Fourcaud et Gilles le Moguedec pour leur humanité et leur présence bienveillante. Merci à Miléna, Murielle et Nathalie pour leur aide inestimable pour toute la partie administrative, et leur complaisance vis-à-vis de la ponctualité toute relative du rendu des différents formulaires qui ont parsemé ma thèse. Merci à l'équipe de la cantine du cirad, pour les fish and chips et les boulettes de viande qui ont égayé tant de mes journées. Merci à Larbi pour la propreté du labo, je regrette que nos échanges ne se soient trop souvent résumés à "Bonjour, vous allez bien ?". Merci aux copains du labo pour les conversations et l'entraide: Anthony, Antoine, Begüm, Camille, Claudia, Clément, Colin (pas moi), Dimitri, Émilie, François, Fred, Giann Karlo, Guangqi, Houssem, Jeanne, Juliette, Laetitia, Lily, Mathias, Paul, Pierre, Raphaël, Tom, et tou·te·s les autres que j'oublie sûrement (pardon).

Pas de palais, pas de palais. Merci à mes colocataires Boris, Chloé, Corentin, Fred, Johanna et Yunus de m'avoir supporté pendant ces trois années. Un merci tout particulier à Boris et Corentin, ça n'a pas toujours été facile de faire trois thèses en parallèle dans un même appartement (et dans un même labo), mais ça m'a vraiment fait plaisir de l'avoir fait ensemble. Ce n'était pas toujours évident de concilier travail et collocation, mais je trouve qu'on s'en est plutôt bien sorti !

Pas d'amour, pas de force. Merci à mes amis pour leur réconfort et pour m'avoir aéré la tête quand j'étais sous l'eau. Merci pour les raclettes, les jeux de société, les randos et les conversations infinies sur la réfraction du son. Merci à ma famille pour leur soutien et leur patience, même si l'on ne s'est malheureusement pas beaucoup vu pendant ces trois ans entre la thèse et le covid. Merci Jeanne pour la répétition en visio, et merci pour le remake de la soutenance à Noël. Enfin, merci Amandine d'être là, de me ravitailler en cocottes, de me faire penser à l'après, d'avoir supporté tout ce temps passé à travailler, de partir en vadrouille avec moi en camion, de penser que...

Synthèse

Cette thèse est à l'interface entre informatique et écologie, elle présente des méthodes formelles pour la modélisation et l'analyse de la dynamique d'un écosystème sous la forme d'un graphe état-transition à l'aide de techniques de model-checking. Tout d'abord, nous introduisons le concept de graphe état-transition d'écosystème qui, bien qu'étant original, capture une longue histoire de représentations disparates de la dynamique d'un écosystème sous la forme d'un graphe. Un graphe état-transition décrit le comportement d'un système comme un graphe dont les noeuds sont les états du système et les arrêtes sont les transitions entre ces états. Un graphe état-transition est donc une carte représentant les différentes trajectoires qu'un écosystème peut emprunter. La gestion d'un écosystème revient alors à diriger la trajectoire que celui-ci suit dans le graphe, par exemple pour restaurer un milieu dégradé.

En informatique, un graphe état-transition représente les différentes exécutions possibles d'un logiciel. Le model-checking est une méthode automatique d'analyse utilisée pour s'assurer de l'absence de bug durant l'exécution d'un logiciel. Un outil de model-checking vérifie que les trajectoires du graphe état-transition satisfont une propriété d'intérêt décrite par une formule de logique temporelle, par exemple qu'un comportement nuisible est toujours évité. Dans la perspective de la gestion d'un écosystème, le model-checking pourrait par exemple être utilisé pour vérifier que les trajectoires de l'écosystème conservent une espèce d'intérêt. En informatique, le model-checking est utilisé itérativement: tant que le model-checking trouve un bug, on modifie le logiciel, ce qui modifie le graphe état-transition de ses exécutions, et on teste ce nouveau graphe état-transition avec l'outil de model-checking. Cette procédure ne peut être appliquée directement à la gestion d'un écosystème en remplaçant le logiciel par l'écosystème, en effet on ne peut/veut pas modifier directement l'écosystème mais on veut influencer sur son comportement.

C'est pourquoi nous proposons dans cette thèse une méthode d'analyse basée sur le partitionnement des états entre ceux dont les trajectoires vérifient une propriété d'intérêt et ceux dont les trajectoires ne la vérifient pas. Ainsi le but n'est pas de modifier l'écosystème, mais de sélectionner les états ou les transitions permettant d'influencer son comportement. Cette méthode, dérivée du model-checking CTL, résulte en une représentation hybride explicite/symbolique, c'est à dire en un graphe dont chaque noeud est un sous-ensemble de noeuds du graphe état-transition initial. Cette méthodologie est implémentée dans `ecco`, une boite à outil développée en Python pour la modélisation et l'analyse d'écosystèmes. Cette approche est illustrée par deux études de cas: les changements de végétation de la région du Borana en Éthiopie et l'assemblage de communautés de protistes. Dans le premier cas, on sélectionne les scénarios de gestion empêchant l'embuissonnement ou le rendant réversible, dans le second cas on partitionne les états par rapport aux communautés stables auxquelles ils mènent. Ces deux études de cas sont limitées par le fait que l'on voudrait que certains événements, par exemple les change-

ments de scénarios de gestion ou les invasions d'espèces, ne se produisent pas arbitrairement mais de manière contrôlée.

Cette limitation peut être surmontée grâce à ARCTL, une extension de CTL qui permet de restreindre les transitions autorisées au cours de la formule. Nous étendons ARCTL avec la notion d'équité, c'est à dire de contrainte de réalisme sur le l'ordre et le taux d'occurrence d'événements le long des trajectoires, ce qui résulte en FARCTL. Nous fournissons un algorithme symbolique pour le model-checking de FARCTL qui est implémenté dans ecco. Enfin, nous appliquons FARCTL aux deux études de cas: pour examiner les conséquences des changements de scénarios de gestion sur la dynamique de végétation du Borana et les conséquences d'invasions d'espèces dans l'assemblage de communautés de protistes.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Analyse de graphes état-transition d'écosystèmes à l'aide de model-checking

Mots clés: graphes état-transition, écosystèmes, model-checking, ARCTL, équité

Résumé: Cette thèse présente des méthodes formelles pour la modélisation d'écosystèmes et l'analyse des graphes état-transition résultants à l'aide de model-checking. Tout d'abord, nous introduisons le concept de graphe état-transition d'écosystème qui, bien qu'étant nouveau, capture une longue histoire de représentations disparates de la dynamique d'écosystèmes sous forme de graphes. Ensuite, nous proposons une méthode d'analyse basée sur le partitionnement des états grâce au model-checking, qui résulte en une représentation hybride explicite/symbolique appelée graphe de composantes. Cette méthodologie est implémentée dans ecco, une boite à outil développée en Python pour la modélisation et l'analyse formelle d'écosystèmes. Cette approche est illustrée par deux études de cas: les changements de végétation de la région du Borana en Éthiopie, et l'assemblage de communautés de protistes. Ces deux études de cas sont limitées par le fait que l'on voudrait que certains événements, par exemple des changements de scénarios de gestion ou des invasions d'espèces, ne se produisent que d'une manière contrôlée. Cette limitation peut être surmontée grâce à ARCTL, une extension de CTL qui permet de restreindre les actions autorisées au cours de la formule. Nous étendons ARCTL avec la notion d'équité, ce qui résulte en FARCTL, et nous fournissons un algorithme symbolique pour le model-checking de FARCTL qui est implémenté dans ecco. Enfin, nous appliquons FARCTL aux deux études de cas, pour examiner les conséquences des changements de scénarios de gestions, et les conséquences d'invasions d'espèces.

Title: Analysis of state-transition graphs of ecosystems using model-checking

Keywords: state-transition graphs, ecosystems, model-checking, ARCTL, fairness

Abstract: This thesis presents formal methods for the modelling of ecosystems and the analysis of their state-transition graphs using model-checking. First, we introduce the concept state-transition of ecosystems graph that, while being a novelty, captures a long history of disparate representations of the dynamics of an ecosystem as a graph. Then, we propose an analysis methodology based on the partitioning of the states using model-checking, which results in hybrid explicit/symbolic representation called component graph. This methodology is implemented inside ecco, a Python toolbox developed for the formal modelling and the analysis of ecosystems. This approach is exemplified in two case studies: vegetation changes of the Borana Zone in Ethiopia under diverse management scenarios, and protists community assembly. Both case studies are limited by the fact that we would want some specific events, for example changes in management scenarios or species invasions, to occur only in a controlled manner. This limitation can be overcome using ARCTL, an extension of CTL that allows to restrict the set of enabled actions along a formula. We extend ARCTL with fairness constraints resulting in FARCTL, and provide a symbolic model-checking algorithm for FARCTL that we implemented inside ecco. Finally, we apply FARCTL to both case studies, to investigate the consequences of shifting between management scenarios, and to look for specific invasion behaviours.