ADAPTIVE SIMULATION MODEL CONFIGURATION

by

Randall Gray, B.A. (Flinders University of South Australia)

Submitted in fulfilment of the requirements
for the Degree of Doctor of Philosophy

Department of Mathematics
University of Tasmania
October, 2016

Signed: ______________________________
Randall Gray

Date: ____________________

ABSTRACT

Models of complex systems may be improved by allowing the way they represent component parts to change as the state of the model and its components move through their state-spaces.

An argument is made that both process fidelity and computational efficiency can be improved. This is demonstrated by a simple adaptive model that changes its basic representation for a population encountering contaminants performs better than either of the conventional forms alone. In this example, there was a substantial decrease in runtime relative to a purely *i*-state configuration model, and the adaptive model performed as well as the *i*-state configuration model, while the purely *p*-state implementation suffered from errors arising from the "blurring" of contact with contaminants through the distributed population. This example develops the notion of model representations maintaining *i*-state configuration data of other representations so that representations can be recovered with minimal error.

Deciding when to change representations of components is addressed initially in a paper exploring a possible set of dynamics associated with a simple, hypothetical model of a seven component ecosystem. The components of the system are described as *i*-state configuration models or as *p*-state models, and a mechanism for determining when to change representations is outlined.

To support the analysis and selection of representations, a metric-space with the properties of a commutative ring is defined. The elements of the metric-space are trees which can be used to encode the structural character of a set of submodels which comprise the model of a system, and to allow a targetted transition from one configuration to another.

Finally, a complete model is presented. The model closely follows the hypothetical example and was designed as a reference model, and is freely available on `https://github.com/snarkypenguin/Mutans.git` This implementation demonstrates the proposed dynamic configuration management, the maintenance of the state of superceded submodel representations, and the support structures needed to implement models of this type.

ACKNOWLEDGEMENTS

My deepest thanks go to my very patient wife, Anne, who coped with far too many years with me firmly attached to my keyboard. Thanks also go my supervisor, Simon, who was never short of optimism and encouragement. I would also like to acknowlege my debt and gratitude to the late Dr William Cornish of Flinders University, who kindled my love of mathematics in 1985.

CHAPTER 1

Introduction

1.1 Overview and introduction

This thesis puts forward the argument that better models may be built if we allow the representation of component parts of a model to change according to their state. The nature of their interaction with other components, and the needs and states both of the other components and of the model as a whole. Practitioners in many fields often appeal to Occam's Razor when selecting models, explanations or solutions to problems and, in some way, the proposed strategy embeds an analogous winnowing in the very structure of a model—a representation of the system that best meets the requirements is sought, and changes are made in the representation as the conditions within the model change.

Such a strategy has a number of potential benefits:

- we can make many simple representations, submodels, for a niche (*sensu* Gray et al. [2006a] and Gray et al. [2014a]) in the model, each of which deals well with a particular part of the submodel's domain;
- the comparative simplicity of these representations effectively reduces the number of potential code paths within a model at any given moment, since representations do not have to cope with edge cases, they merely indicate that they are entering a marginal or inappropriate domain;
- we can use analytic representations which are more efficient at representing large numbers of entities;
- we can use individual-based representations which capture the fine-scale dynamics that dominate when we are dealing with discrete events or low numbers of entities;
- we can choose representations that make the best use of available data within the model, or can ask for better representations in the ensemble;

- it is simple to incorporate code to track information about representation changes, relative execution speed, and cumulative error into the modelling system;
- we can include (or not) agents that identify the emergence of perverse dynamics within the system;
- we can decouple the production of the results from processes which simulate the systems and subsystems being modelled.

This strategy for building models makes it simple to address the questions *"How do we deal with situations where the assumptions that underpin our representation no longer hold?"* and *"How do we manage the execution of submodels which simulate systems or entities with multi-modal behaviour?"* quite straightforward.

Consider this example: rather than a single representation for the population of a coastal city, we may have a number of different *submodels* with different levels of aggregation and different temporal or spatial scales. In a simulation of a cyclone season, we may start with a simple single age-histogram representation. As a tropical storm build we may disaggregate the histogram, appropriately distributing the population to finer age-histograms associated with localities throughout the region. As it approaches the coastline, the essentially static representations which lie in areas likely to suffer damage are converted into *agents* — instances of running submodels — which represent households. Shortly before the cyclone reaches the point where damage occurs, we may resolve the representation further, instantiating emergency response agents and converting households agents to individuals at risk. In the aftermath, aggregation may occur in regions where there are no acute effects, but other parts of the system may remain finely resolved.

In this example, we change the representations to deal with both of the questions above. We initially assumed that for most of the purposes of the simulation, our population could be treated as relatively homogeneous, and may have kept aggregate information about population distribution, wealth and demographic characteristics, but as the storm hits, the simulation needs to change the state of a portion of the population (those with damaged property, for example), and we have to refine our representation. Similarly, the behaviours following the storm are modally different: those who live in protected areas are mostly free to carry on with essentially the same "normal" representation, but those who living in damaged areas must engage in quite different activities, and may have quite a different exposure to risks.

Adaptive approaches commonly occur in numerical technique for numerical approximation (regression, root finding, and parameter estimation for example), numerical solutions for systems of differential equations, feature detection and recognition, control systems and route planning. Often these approaches involve adjusting the size of the domain considered (subdivisions or step size), or the rates associated with a process. In the case of route planning, sets of routes may be marked as "impassible" as data becomes available, triggering a reassessment of the set of possible routes. More broadly, *domain* *adaptation* describes a general approach where a model or system adjusts itself to the data it works with—one of the canonical examples is Bayesian spam filter which includes a user's assessment of whether email is spam or not in its subsequent assessments. A common trait these adaptive techniques share is that *algorithm* which processes the data remains essentially the same.

Zhang et al. [2016] describe a system for the detection and tracking of pedestrians which selects the algorithms and parameters to be used in the analysis of segments of data based on the nature of the data it is given. This approach has qualitative similarities to the approach suggested in this work, since the whole method of evaluation changes based on its input, rather than adjusting the scales or domains.

Discrete changes in the behaviour of a system, or part of a system, are commonplace. The scales of systems that exhibit switching behaviour range from a molecular level, such as the behaviour of freezing liquids, through to climatic changes. The discussion of "tipping points" has increased dramatically in the last decade ([**?**]), indicating a broad recognition that systems' dynamics can (and do) switch rapidly from one mode to another

Huston et al. [1988b] is an early review paper which deals with individual-based modelling as an alternative to purely equation-based models which deal with population level data. It opens with the observation to the effect that mathematical models in ecology often make the assumptions "[individual organisms] can be described by a single variable, such as population size", and that "each individual is assumed to have an equal effect on every other individual" [because locations are ignored]. They argue that individual-based models are able to incorporate dynamics across scales and that the fine scale dynamics experienced by individuals plays a significant role in many of the population scale patterns observed in ecological studies.

These models are not a panacea, however—models of this sort may become quite costly as the number of individuals or the interactions between individuals or super-individuals grows, and small discrepancies between the "behaviour" or parameterisation of the modelled entities and their real counterparts may produce large discrepancies at the population level. Many of the parameters that may influence the life-history of organisms at an individual level are difficult or impossible to estimate in Stu, and so the effects of individuals' modelled behaviours or processes may not scale well.

It is not uprising that exogenous factors, such as changing environmental conditions such as the availability of water or prey can significantly alter the behaviour of organisms. Migrations are common in many populations, reasons include moving to particular locations for breeding, avoiding seasonal scarcity of resources, or the encroachment of competing species such as *H. sapiens*. These examples are relatively predictable, and typically involve a homogeneous response from the migrating population. Ward and Krebs [1985] discusses the behaviour of lynxes in response to declining prey populations. In this study, two distinct behavioural responses prevailed: some animals choose a nomadic lifestyle as a means of optimising their likelihood of hunting success, while others remain in their own territory. Though Ward's sample size was small, the distinct responses suggest that lynx populations respond to prey scarcity in a heterogeneous way and, as a result, may be less amenable to modelling as a population.

Toxic ants and other chemical contamination in the environment are likely to have a contained spatial distribution and an uneven impact on members of a population. The significant issues in modelling this type of interaction is that the population can be fragmented into a number of distinct sub populations bases on their level of contact and uptake, and airborne or water born contaminants' footprint may vary dramatically through time. Zala and Penn [2004] presents a useful review of the the effects of behaviour disrupting contaminants in a broad range of animals. Even in simple situations, where the behaviour and viability of members of a population are not compromised, the consequences of contact may percolate through the food-chain to higher order predators. Swan et al. [2006] has found that very low levels of cloven are fatal to old world vultures which acquire it by scavenging carcasses of dead cattle, and the bio magnification and effects of DDT in the food chain have been well discussed in the scientific literature since 1964.

When an altered behaviour is associated with the spread or reproduction of organisms; in these cases, the scope for positive feedback is increased, and the dynamics can diverge rapidly from representations that are adequate for an unperturbed system. The effects of *T. gondii* on rats ([**?**]) is an ideal example: rats exposed to infected cat feces lose their innate fear of cats — the positive reinforcement on the spread of *T. gondii* afforded by this leads to greater potential for the pathogen to infect more cats, and hence, more rats. Dobson [1988] investigates the population dynamics of these kinds of interactions, and present a useful approach to incorporating these effects into analytic models. This paper explored the population dynamics of parasite-host systems where the parasites influenced host behaviour found that the reproductive capacity of the host populations could be significantly modified by the behaviour altering parasites. [**?**] observes that roaches infected with *Lingual intestinal is* were three to five times more numerous in cormorant catches than in the roach population of the IJsselmeer (as estimated from commercial catches), suggesting that something in the fish's behaviour makes them more susceptible to capture. Poulin [1994] assesses the effect on host behaviour in a number of host-parasite pairings, and found that the parasites had a significant effect on the behaviour of their hosts. In the cases addressed in these papers, the process in question is predation, and the infected individuals are often either disproportionately preyed upon, or involved in the parasite's reproductive cycle.

The situation is more complex when there are endogenous reasons for fundamental changes in their basic dynamics. This kind of situation can induce radical, even pathological, changes in behaviour. Social animal populations may behave in quite strange ways when their population density grows too large or the population's social profile is disrupted. A seminal (and grim) example of this is described in Calhoun [1973]. Calhoun recounts an experiment in which mice are confined in a domain where all their physical needs were met, all possible sources of mortality apart from senescence and death by injury were excluded, and there was no possibility of emigration. Social and behavioural disintegration began to manifest in the third generation (day 315), and the population went into terminal decline after 560 days, and for all practical purposes the social organisation had collapsed utterly.

Calhoun discounted the population density as the cause of the social disintegration, rather attributing the collapse to the inability of young adults to engage in "normal roles" due to high competition for the *social niches* which were filled by older, more established mice. The behavioural changes attending the social collapse did not revert to more normal when population levels dropped (past 560 days).

Systems comprised of a number of the types of these types of systems may exhibit dynamics which are difficult to address in a conventional way. Software development principles encourage loose coupling and narrow interfaces between submodels, but as the complexity and range of functional elements being modelled increases, the number of potential interactions for a component grows. With this growth, the ability to ensure the robustness of a submodel over its range of potential states and interactions may become prohibitively difficult, simply because it is difficult to check all possible interactions and execution paths. Even with very clean, robust code, the probability of poor interactions during a run increases dramatically as the number of lines of code in submodels increases. In my own experience in modelling human-ecosystem interactions, the demand for richer models of ecosystems and more detailed models of human activity has driven an increase in the number of types of submodels and their complexity by roughly a factor of four every seven years, though it may be much faster. In part this may artificially limited to the tractability of the problem with respect to the hardware available, but, even so, the growth in the complexity of the code required

1.2 Historical work

Many individual-based models incorporate environmental characteristics that influence the behaviour of of the individuals simulated. Botkin et al. [1972c] and DeAngelis [1978] are important early examples: Botkin *et al.* modelled the effect of spatially explicit environmental conditions on simulated trees (rather than stands or coupes) in a mixed species population in North America; in the case of Botkin *et al.*, the model was used to explore the distribution of fish (modelled as individuals) in a speculative body of water with a known distributions of temperature and food availability. Both of these models simulated the dynamics resulting from the physical conditions real plants and animals might encounter.

The coastal marine ecosystem models in Gray et al. [2006a] used different representations for the organisms based on their life-stage. As organisms that comprised the benthic habitat matured their representations would change to suit their niche in the system. In this case the sequence of transitions was determined before compilation of the model; juvenile biomasses could be represented either by gridded cellular automata or by polygonal clouds which changed shape as they were advected, while adult stages could be represented by several different submodels which might be optimised either for speed or for spatial fidelity. This model was in most other ways similar to conventional agent-based modelling of the time.

Bobashev et al. [2007] describes a model of epidemic simulation in which the representation of populations or portions of populations are decided based on the number of infected individuals relative to a nominated trigger value. This model demonstrates that there is an advantage to changing representation in terms of computational efficiency and the fidelity of the model. The published model in Chapter 2 is similar: the rule governing the switching from one representation to another depends on the state of the system: switching occurs when some monitored quantity crosses a nominated boundary. The problem of contact with infection and contact with a contaminant is similar, though the treatments are quite different. In Bobashev et al. [2007], an individual moves from `Susceptible`, through `Exposed`, to `Infectious` and then `Recovered`; in contrast, when individuals with contaminant loads in Chapter 2 leave the region where contamination is possible, they are subsumed back into the population and their data is incorporated so that the individual contaminant profiles are maintained and subject to depuration.

The model discussed in Gray et al. [2014a] and Fulton et al. [2009] simulates the effect of a number of strategies for managing human recreational and industrial activity along the northwestern coastline of Australia. The model incorporated incorporated distinct individuals, super-individuals, mean field submodels, submodels which were equivalent to cellular automata and systems of differential equations. In this model, the representation of whales was notable because individual whales would be transferred to another (mostly inaccessible) domain when their migration took them outside the model's domain: the whales would be maintained in a rudimentary way until it was time for them to migrate back into the model domain. Other agents within the model, predators, wildlife management and tourism operators, were influenced in their decision processes by the presence or absence of whales in the model domain.

1.3 Structure

Models of complex systems usually incorporate alternative code paths or expressions in a component to deal with situations where there are fundamentally different dynamics or properties by testing for these conditions at each potential fork in the code. This can engender a complicated network of potential execution paths through the model. In contrast, the approach discussed in this thesis addresses this problem by constructing the models as an ensemble of agents which can cede their role to another representation which is more suitable when the need arises. A resident population might thus be represented by a single *population* agent, a set of *individual-based* agents, *super-individual* agents or by some mixture of these representations. It may be that particular representations are unable to respond to the conditions they encounter: a population based agent may be unable to to interact with a contaminant plume, for example, and a representation that can is required for the interaction to occur. Should contact with the plume no longer become necessary, the system ought to be able to convert the representation back to its original form, with information about the contaminant load maintained (and depurated, if appropriate) in the original representation.

The decision to change the representation of a submodel occupying a niche in the model is based on the state of the system, the capabilities (or incapability) of the agents in the system and the objectives of the modeller — in configuring a model, we might prioritise speed over accuracy in a real-time simulation in a computer game or combat training simulator, or the obverse for a scientific extrapolation of the state of a harbour for each of a number of development scenarios.

Representing the state of the model, either as a whole or of its constituent parts, is not simple: not only may the filling the niches of the model vary through time, but their own dependences on other components and niches may change as the state of the model changes. A model which contains a "whale spotting" tourism venture may follow the activities of whales at particular times of the year, but be utterly indifferent to the whales at times when there is little likelihood of their presence in an accessible location. Similarly, the association of entities represented by a population-based model may need to be maintained if the population disaggregates into agents based on super-individuals (small cohorts) or agents representing individuals.

The interplay of factors like these make a simple vector-based encoding mechanism for the states of a model and its components awkward, thus we turn to a metric space whose elements are trees with a finite number of weighted, labelled nodes. This simplifies the comparison of possible ways to fill the niches in the model for a given global state, and makes available any algorithms (particularly useful are clustering) which depend only on the properties of a metric space.

The decision to change representations can be made by an agent that recognises that it is unable to continue in the conditions in which it finds itself (akin to the code-path decisions in more traditional models), or by a similar assertion from some higher agency (a *monitor* in the discussion which follows) which assesses states more broadly. In the case of an agent determining that it needs to change, such as a penguin moving from the "nestling" submodel to the "juvenile" submodel, this can be effected directly, though the more general (and in this case, burdensome) strategy would be for the desired state-change to be flagged and acted on by a *monitor*.

The models explored in this work bear a resemblance to a multitasking operating system, but, unlike an operating system, the model's "kernel" must maintain temporal ordering in the execution of agents ... the start times within the agent queue must be strictly non-decreasing. Interactions between agents are largely mediated by the kernel and new agents may be created or removed with relative ease. Choosing this as an organisational template means that there are many patterns to serve as templates for further development.

All of the submodels in the example model presented in Chapter 5 are able to act to some degree as a kernel themselves — in a sense, models can be nested.

1.4 Scales

The natural time step or spatial scales of a model may change if one or more of its constituent submodels changes its representation. This seems like an obvious statement, but many models are structured with quite carefully chosen time steps and spatial scales, and they may behave quite poorly when these scales are changed. Models of individual organisms are likely to require much smaller temporal and spatial scales than representations at a population level, so a model which seeks to accommodate both possibilities must necessarily be able to adapt to the scales that are important at the moment. In practice, changing spatial resolution is relatively straightforward if submodels do not rely on knowledge of the underlying implementations of other submodels. The sorts of causal issues that accompany predation, for example, tend not to arise when we move from a finely gridded landscape to a coarser version.

Changes in temporal scale can be more problematic, however. Time influences causality in a way that space does not. Deciding how to manage the flow of time in an ecological simulation model is one of the first decisions in its design. Many ecological models have been constructed as a large set of arrays containing state variables which are inspected and updated in the body of an event loop (or many loops). Some models achieve temporal optimisation by dividing the arrays into various groups of fast-stepping and slower-stepping variables, only dealing with the necessary parts of the system at each timestep (*variable speed splitting* as in Walters et al. [2000], for example). Gray et al. [2006a] and Gray et al. [2014a] allow agents to dynamically determine their own time step based on their state— time steps may be truncated, or changed for their next turn in response to their situation. There is a trade-off in this: with a variable speed splitting approach, we can calculate all values based on a temporally coherent set of data, and update them all in one pass; in contrast, the dynamic time-stepping approach means that each interaction is essentially conducted in isolation, and the consequences of a set of interactions may be dependent on the order in which the interactions occurs. Both variable speed splitting and dynamic time-step selection are flexible enough to support representational changes for entities, but the greatest advantage comes from constructing the submodels to be robust with respect to arbitrary time steps over a reasonable domain. If a model is consistently run with time steps which are too long or too short, the model or system needs to be able to initiate a change to a more appropriate representation.

Inappropriate or incommensurate time steps can pose a real problem: while the interactions between submodels with short time steps and submodels with long time steps may be managed, at least to some degree, by accumulating changes to the slower model and applying them during the slower model's time step, this is not an ideal solution. One of the major risks this approach poses is a of distortion of resource availability which is dependent on the order in which agents are executed. This sort of error can artificially inflate or deplete apparent resources in a seemingly random fashion, and render the results of the simulation useless.

The principles which have guided the coupling of models remain salient, particularly those aspects associated with issues of coherence in time and space. While matching time steps isn't essential, the discrepancy between the time steps of interacting models should be limited by the magnitude of the changes which may occur as a result of interactions — large changes may call for small time steps.

1.5 Outline

The paper which forms the body of Chapter 2 develops a model of organisms that periodically move through a region subject to plumes of contaminant. The model is capable of modelling the organisms either with a population-based representation or with an individual-based representation. This model is run in three configurations: purely population based, purely individual-based and as a hybrid where the individual-based representation is used when it is possible for any of the population represented to come into contact with the contaminant, and with the population-based elsewhere.

The purpose of the model and its runs is to compare both the execution speed of the simulations and the fidelity of the simulation with respect to the contaminant loads of the simulated population.

Chapter 3 was published in a special issue of *Frontiers in Environmental Science*. It considers the properties needed for a more complex evaluation of possible configurations of a running model, and develops a speculative model as a platform for discussion. To support the dynamic assessment and selection of model configurations, the paper introduces a metric space based on a tree structure. The metric space allows us to calculate distances between configurations and to reduce our potential search spaces by identifying clusters of representations that are largely similar in their constitution.

The final section of this thesis describes an implementation of a framework for a models of this sort, and the implementation of the model described in Chapter **??**. The corpus of code in the framework is roughly 15,000 lines of Scheme which is freely available at `github.com:///snarkypenguin/Mutans.git`.

An essential notion that was treated lightly in the paper of Chapter 2 is there developed much more fully, namely that for a model to allow an oscillation between representations, additional data must be passed between them, maintained and possibly adjusted in order to preserve consistency across transitions. The model discussed in Chapter 2 does this by having the population

maintain a vector of contaminant levels which is updated at each time step, but a more general solution is presented.

CHAPTER 2

Increasing model efficiency by dynamically changing model representations

2.1 Prologue to the paper

This body of this chapter is a (verbatim) paper published in the refereed journal *Environmental Modelling and Software* (Gray and Wotherspoon [2012]). The model discussed in this chapter is built using an older, much simpler body of code than the framework that is explored in Chapter 3. While the primary purpose of the paper is indicated by its title, a significant contribution is its development of the basic mechanisms for changing representations.The source-code can be obtained from `https://github.com/snarkypenguin/Model-Efficiency.git`

The paper explores a model of marine organisms that periodically migrate through an intermittent plume of some contaminant. The uptake and depuration of the contaminants in the organisms are modelled, and the paper compares the results and the run-time for the system in three forms in order to establish how useful using model switching may be in terms of run-time and fidelity. The three configurations tested are

- a purely analytic representation of the migrating population — In this configuration a population whose relative locations follow a Gaussian distribution is moved around the migratory circle, and the uptake of contaminant is calculated using a Runge-Kutta4 algorithm
- a purely individual-based model — Individuals move through the contaminant zone, integrating their contact with the plume and generating an uptake level appropriately
- either a population (as described), a set of individuals (also as described), or as a mix with part of the cohort represented as individuals within the

> risk zone, and the balance represented as a population — Here, individuals are generated (and removed from the population) as the population disk encroaches on the contaminant zone. As individuals leave the contact zone, they are subsumed by the population and their individual contaminant level is decayed appropriately.

The analytic submodel is the simplest of the representations, consisting largely of a value (or vector) which records the contaminant load, and the number of individuals which are represented. A single instance of the analytic submodel is present in purely-analytic and in the switching model. The individual-based model incorporates a number of state variables, such as velocity, contaminant load and the instantiated agents are independent of the analytic representation.

In order to avoid confounding the results, the contaminant is inert since including toxicity effects would have the potential to alter both the number of entities modelled and their behaviour (such as movement rates); since slower individuals would take longer to move through contaminated regions, their likelihood of contact would be higher. The scenario in the model would be substantially similar to simulating the uptake of isotopes associated with particular geographic locations.

Given the very predictable dynamics of the populations, the inclusion of multiple contaminants or multiple sources, such as in Gray et al. [2006a, 2014a], seemed unlikely to improve the argument.[1]

[1]incorporated mortality or morbidity associated with contact, the mortality strategy used in Gray et al. [2014b] would have been an appropriate choice.

Increasing model efficiency by dynamically changing model representations

Randall Gray[2]

CSIRO Division of Marine and Atmospheric Research

Simon Wotherspoon

University of Tasmania

Abstract

There are a number of strategies to deal with modelling large complex systems such as large marine ecosystems. These systems are often comprised of many submodels, each contributing to the overall trajectory of the system. The balance between the acceptable modelling error and the run-time often dictates the form of these submodels. There may be scope to improve the position of this balance point in both regards by structuring models so that submodels may change their algorithmic representation and state space in response to their local state and the state of the model as a whole.

This paper uses an example system consisting of a single population of animals which periodically encounters a diffuse contaminant in a localised region as an example of such a system, and discusses the key issues that arise from the approach.

2.2 Introduction

There is a body of literature stretching back several decades which discusses individual-based modelling as a useful alternative to classical models. Early examples modelled forest canopy dynamics, notably JABOWA and its derivatives (Botkin et al. [1972b], Botkin et al. [1972c]). The number of significant papers and books has steadily increased since the 1980s. These works describe the use of individual-based models across a broad range of systems, and the relative strengths and weaknesses of the approach (such as Huston et al. [1988a], DeAngelis and Gross [1992] and Grimm and Railsback [2005]). Classical models exploring populations and ecological systems are usually associated with modelling the dynamics of large groups and arguably appeared at the end of the eighteenth century with Malthus's ***An Essay on the Principle***

[2]Published in *Environmental Modelling and Software*, 2012
Corresponding author: `Randall.Gray@limnal.net` (Randall Gray), `Simon.Wotherspoon@utas.edu.au` (Simon Wotherspoon)

of Population (1798). The properties of these models are well understood and their state variables usually correspond to measurable quantities. Often, they are much faster than individual-based counterparts, and the analysis of model error may be much more straightforward. Classical and individual-based approaches represent the ends of a spectrum of aggregation in time, space and membership. Representations lying between these extrema, such as described by Scheffer et al. [1995], capitalise on the process-fidelity of an individual-based representation and gain some of the computational efficiency of a more aggregated classical approach, but an adaptive exploitation of the strengths of different representations is possible and worth exploring.

Ecosystem models are becoming broader in scope (Rose et al. [2010], DeAngelis et al. [1998], Harvey et al. [2003] Fulton et al. [2004], Gray et al. [2006a], webpage) and include more species with richer environments. The environmental response to climate change has also made anthropogenic pressure an important feature in many of these models. As this trend grows it seems less likely that a single model drawn from any particular region of this spectrum will be able to address all members and processes equally well. Simulation models often embed their subject in an "environment" comprised of primary data and other models and these components may occupy many places in the spectrum of representations. The model's actual implementation may be anything from a set of distinct models which are coupled together but retain their independence, to a corpus of code with the submodels so integrated that there is no real distinction between one "model" and the next.

The dynamics associated with biological and ecological systems can depend on the distributions and states of individuals in ways which are not amenable to equation-based modelling. The individual-based models described in Farolfi et al. [2010], and de Almeida et al. [2010] deal with systems of this sort. Versions of these models could be embedded in a common simulation environment in order to address more complex problems which span traditional domain boundaries, and such a model could address broader questions, such as how mosquito control strategies may best adapt to evolving agricultural practices and watershed conditions. Thiele and Grimm [2010] describes an extension to NetLogo which allows modellers to incorporate calls to R functions to aid in configuring the model to meet desirable mathematical conditions, to provide ongoing analysis, and to display the model's state through its run. This interface between R and NetLogo could be extended to support incorporating mathematical decision models written in R into the model's decision tree. The fusion of these three elements would form a system capable of simulating possible trajectories for the management of watersheds and human health in ways which would not be possible with a traditional monolithic modelling approach.

Models are including more functional groups and the interactions between components are becoming more detailed. It is costly in terms of computational load to address this increased demand for detail: individual-based models of populations may be very good at capturing vulnerability to exceptional events, but such simulations take a long time. Much of this time may be spent with the model in a largely unchallenging or uninteresting part of its state-space.

This paper explores the technique of changing the representation of a component of a model based on its location in its state-space. Modellers already do this to some degree: time-steps or spatial resolutions are changed, particular code paths may be by-passed to avoid pointless work, or additional calculations might be performed to reduce the error when the state is changing rapidly. These optimisations are largely optimisations of the *encoding* of the model or submodels, rather than an actual change in representation.

Vincenot et al. [2011a] make a clear case for considering what the authors term "hybrid-models." They present four reference cases which they use to describe ways in which equation-based models and individual-based models might be coupled to increase their utility. Their categories of hybrid-models are: individual-based models interacting with a single system dynamics model, system dynamics models embedded in individual-based models, individual-based models interacting with a number of system dynamics models, and models in which the representation swaps between individual-based and an equation-based form. They argue that a hybrid approach may provide a means of increasing the speed and accuracy of our models and Lyne et al. [1994b] and Gray et al. [2006a] have demonstrated that large models of ecosystems can be modelled this way. Vincenot et al. note that they found relatively few models which use both individual-based and equation-based submodels, and they present no existing models representing their fourth reference case. This final case, where models swap from equation-based to individual-based, is briefly described in general terms and is clearly intended to encompass models like the model of this paper.

This "mutating" or "switching" approach to the problem of managing complex simulations was developed using the experience from making several large scale human-ecosystem interaction models (Lyne et al.; Gray et al.; and a current, larger study of Ningaloo coastal region (*work in progress*)). In each of these studies a significant component of the model focused on simulating the interaction between organisms and contaminant plumes, though there is nothing that inherently limits the techniques to these sorts of studies. Lyne et al. assessed the potential of contaminants originating in industrial waste percolating through the food chain into commercially exploited fish stocks. Gray et al. developed a regional model to assess management strategies for human activity which interacts with the biological systems along the Northwest Shelf of Australia.

Simulating contaminant interactions in an ecosystem is expensive in terms of run-time and memory use. The models described by Gray et al. and Lyne et al. include contaminant transport, uptake and depuration modelling, with behavioural sensitivity to contaminants. In Gray et al., the time taken to run a simulation with contaminants increased by roughly an order of magnitude, and in both studies a large amount of time was spent in regions where no interaction with contaminant plumes was possible. Monte [2009] presents a lucid discussion of analytic *contaminant migration-population effects* models. These models incorporate the movement of populations and their internal distribution, the transport of contaminants through the system via biotic and abiotic pathways, and the changes in behaviour and population dynamics associated with contamination. Monte discusses a method of coupling the equations which govern contaminant dispersion with the equations for population dynamics and migration. The technique depends on the equations of the location and the dispersion of members of a population satisfying an independence condition with respect to time and location which must hold. He states that the class of systems where the "movement of animals, the death and birthrates of individuals in $\mathbf{x}$ [location] at instant t [time] depend on previously occupied positions" is not generally amenable to the approach and suggests that repeated simulations of many individuals is an appropriate way of dealing with this situation.

It is unnecessary to run a complex model and carry the burden of maintaining its state when a simple model may perform better. If representations are switched appropriately, there is potential for improvements in run-time and accuracy. We need to consider four basic questions to do this:

1. What data need to persist across representations?
2. When should a model change representation?
3. How is the initial state for a new representation constructed?
4. How should the error associated with the loss of state information be managed?

The answer to these questions is specific to the set of submodels in question. Before expending resources and effort on a large scale model there needs to be a demonstration that the notion is worth pursuing, and some indication of how it might be accomplished. The aim of this paper is to provide this demonstration rather than to develop a comprehensive body of techniques supporting the approach. Many systems may benefit from similar techniques; obvious candidates are models of marginal populations, and the population dynamics of animals with behaviour where short periods of time have a significant influence on population levels (Wolff [1994] and Elderd et al. [2008], for example).

2.3 Overview: an *ODD* model description

The *ODD* protocol [Grimm et al., 2006] is used to describe the example model. We discuss the issues associated with making such a system, strategies and the reasons behind them in the Discussion section.

2.3.1 Purpose

This example model plays two roles. Its first is as an explicit demonstration, and the second is as a tool to explore the larger subject of changing a model's representation in response to its state. This example is overly simple, but it shares a number of features with plausible models and the analysis and development of the mutating model should be a reasonable template for other systems.

The model simulates organisms moving along a simple migratory path which intersects a region containing a field of fluctuating contamination (see Figure 2.3.1). This model exhibits fundamental attributes of larger studies of pollutant/ecosystem interactions (Lyne et al. and Gray et al.) and, while it is not intended to accurately represent any particular system, it might loosely correspond to some body of water influenced by contaminant loads associated with terrestrial runoff resulting from intense rainfalls.

Figure 2.1: Snapshots of individuals' locations at 28 day intervals superimposed on the migratory path. The plume's contact domain is marked by a grey ellipse near the position of individuals at day 28, with the track of a single individual approaching it. The domain of a population is is circumscribed around the individuals at day 196 for comparison.

The test models are composed of one or more submodels which run within a simple time-sharing system. Each submodel runs for a nominated period of time and passes control to the next submodel, very much like tasks running in many modern computer operating systems. In a *mutating* configuration, a trial will have different models take turns representing components of the system.

The population-based and individual-based submodels have been kept as similar as practicable in order to minimise the sources of divergence.

2.3.2 State variables and scales

There are essentially three distinct submodels in the simulation: an individual-based representation of the migrating group, population-based representation of the group, and a contaminant uptake-depuration model. We can think of the models which take the role of the group as candidates for filling a *niche*, which we can think of as the "sub-model shaped hole" in the middle of the program. Because the individual-based and population based models have fundamentally different spatial representations, each of these models include mechanisms to evaluate their contact with a plume as they move through their environment. The spatial domain of the whole model system is a circular region with an arbitrary radius of somewhat more than 100km which encompasses both the area influenced by the contaminant source and the annual migratory path of the organisms. The plume can be viewed as a forcing function in the model and it has a maximum footprint area of approximately $43km^2$ which may be circular or elliptical and is centered on a point of the migratory circle. Both the elliptic and circular variants of the plume have the same area, and their intensities are adjusted so that the integral of the contaminant concentration over the region is the same.

The individual-based representation maintains a contaminant load associated with contact with the plume, a location, a direction and the next time at which it is scheduled to run. The population-based representation treats the group as homogeneous with respect to all state variables other than the contaminant load, and maintains only a record of its next time-to-run and an indication of contaminant load in the population. In the straight population-based representation, this is a single value, but in the mutating system the submodel maintains a list of contaminant loads which correspond to the non-zero loads of individuals. The plume model is deterministic with respect to time and location and maintains no state variables.

2.3.3 Process overview and scheduling

Simulations were run with 90 minute time-steps for a period representing twelve years. At each time-step, each instance of a submodel is roster-ed in a priority queue sorted on the "time-to-run" state variable, and when it comes to the top of the queue it executes.

Populations operate in a straightforward way: their path is deterministic, exposure to contaminants is calculated, and the resulting values are fed through the uptake-depuration equation. Individuals calculate their path (a segment of a directed random walk which follows the path of migration) and contact for the time-step and an apply the uptake-depuration equation. At the end of a time-step in non-mutating configurations, data is accumulated for output and each submodel reinserts itself in the priority queue. Otherwise, a heuristic is used to choose an appropriate representation for the niche in next time-step and that is inserted into the queue. Randomisation within a time-step is unnecessary, since the individual's or the population's contaminant updates are resolved for the contaminant contact across their time-step and are not dependent on the state of any other agents.

2.4 Design concepts

The central reason for the model is the mutability of the representation of the simulated organisms. Individuals and populations in the model are profoundly simple: no real scope is present for any of the trait categories mentioned in Grimm et al. [2006], apart from their interaction with the contaminant plume, though this interaction is completely deterministic with respect to their path through the plume. In place of these traits, we have the basic heuristics associated with triggering a change from a population-based representation to individuals and a corresponding heuristic which indicates when an individual should join (or become) a population. The actual mechanism which turns a population into individuals or its converse is not necessarily a property of those models. Since the objective is to examine the impact of changing model representation in a fairly narrow situation, no attempt is made to optimise the submodels in the "non-contact" areas which constitute most of the model domain.

2.5 Details

2.5.1 Initialisation

Individuals and populations initially begin with no contaminant load, and individuals are positioned according to the two-dimensional normal distribution which characterises the population's assumed distribution. When a population mutates into an appropriate set of individuals, the individuals are positioned in the same fashion (centered on the centre of the population) with their corresponding contaminant loads either taken from the list of non-zero contaminant loads maintained by the population or initialised to be zero should the population's list fall short.

2.5.2 Input

Several characteristic features of the model are determined by the time and location represented. The contaminant intensity (and hence extent) at any point, $\mathbf{r}$, relative to the centroid of the plume, $\mathbf{m}_{plume}$, at a time, t, by the equation

$$I(t,\mathbf{r}) = \frac{1}{2}(1+\cos(2\pi t/p))\exp(-\psi\phi(\mathbf{r},\mathbf{m}_{plume}))$$

where p is the period of 34 days, $\psi = 0.05$ is a decay exponent. We take ϕ to be a distance function, either $\phi(\mathbf{a},\mathbf{b}) = |\mathbf{a}-\mathbf{b}|$, for a circular plume, or $\phi(\mathbf{a},\mathbf{b}) = \sqrt{(\mathbf{a}-\mathbf{b})\cdot(\sqrt{2},\sqrt{1/2})}$, for an elliptical plume. The effective radius of the circular plume in the model is about 3.7% of the circular migratory path of the populations and individuals. The intensity of the elliptical plume is adjusted by scalar multiplication so that the integral of I for the two plumes over their domain is the same.

The individual-based and population-based models follow a circular migratory path about the origin. The path is traced annually and its location at any given time follows the equation $\mathbf{l}(t) = 10^5(\cos(2t\pi/365.25), \sin(2t\pi/365.25))$.

2.5.3 Submodels

Individual-based representation

Individuals follow a directed random walk around the migratory circle described in the previous section. At each time-step the stride the individual takes is calculated according to its proximity to the "target" on the migratory path. There are a number of parameters associated with the movement of the individuals presented in Table 2.1.

If we take δ to be the length of the current time step, and v to be a realisation of an event in a Poisson-like process with a mean of $\overline{V}$, we can take

$$Q = \left[1 - \exp\left(\frac{v}{\overline{V}}\log(1-q)\right)\right]$$

Table 2.1: Parameters associated with individual movement

Parameter	Value	Description
$\overline{V}$	4	A "variability" parameter associated with a Poisson-like process
q	0.5	A magnitude control parameter on directional change
μ_δ	1 day	Notional interval over which we calibrate individual's movement
μ	20km	Indicates the radius which is likely in a period of μ_δ
s	4ms^{-1}	nominal speed of the individuals

to be a "variation" scalar which we use to evaluate an effective radial speed,

$$\nu_s = \left| -1 + \sqrt{1 + 4sQ^2 \frac{\delta}{\overline{V}}} \right| / 2Q^2.$$

Large values of Q correspond to long stretches of time without a change in direction, so we include Q in the calculation of α, the partial change in the individual's direction vector, by setting it to $\alpha = \pi rnd(-Q, Q)$. We can take their effective displacement over the 90 minute interval to be determined by a weighted sum of the normalised vector which joins them to their "target" location on the migratory path and a direction vector of length ν_s which is deflected by α.

Population-based representation

The population-based model assumes that a radially symmetric, normal distribution of individuals is an appropriate representation. Trials using the movement model of the individual-based model were run, and the positions of individuals relative to their "target" on the migratory circle at each time-step closely matched a 2D-normal distribution with a $\sigma^2 = 3136.25^2$. Using this value, we define the density of the population at the point $\mathbf{p} = (p_x, p_y)$, relative to the population's centre, to be

$$\rho(\mathbf{p}) = S_L \frac{1}{2\pi\sigma^2} \exp\left(-\frac{p_x^2 + p_y^2}{2\sigma^2} \right)$$

$S_L = 1.015$ is a scaling parameter chosen so that the integral over the population's effective disk, $\mathbf{D} = \left\{ \mathbf{q} \in Domain(\rho) : |\mathbf{q}| \leqslant 3\sigma^2 \right\}$, gives

$$\int_{\mathbf{D}} \rho(\mathbf{p})\, d\mathbf{p} = 1.$$

Contaminant handling

Initially a contact value is calculated for the time step. For an individual, this value is the integral of the contaminant level over its path. Population contact is calculated in an analogous way over the domain of the population and it represents the average contact of the members of the population.

The mass of contaminant which is available for uptake, or contact is, for individuals, taken to be the result of integrating the intensity of the plume over its path, $\mathbf{P}_t$ to $\mathbf{P}_{t+\delta}$. Namely,

$$M = \int_{\mathbf{P}_t}^{\mathbf{P}_{t+\delta}} I(\mathbf{p})||\mathbf{p}||d\mathbf{p}$$

where our variable $\mathbf{p}$ is a vector with time and location and we assume that the motion from $\mathbf{P}_t$ to $\mathbf{P}_{t+\delta}$ is along a straight line segment. We take $\|\mathbf{p}\|$ to be the speed at which the individual is moving.

Population's contact occurs across its domain and we calculate the definite integral

$$M = \int_{\mathbf{P}_t}^{\mathbf{P}_{t+\delta}} 2 \int_{\mathbf{\Omega}} I(\mathbf{p} + !)\rho(!)d!d\mathbf{p}$$

where $\mathbf{\Omega}$ is an area over which we assess the effective area of the population and $\mathbf{p} + \omega$ denotes the area $\mathbf{\Omega}$ translated so that its centroid corresponds to $\mathbf{p}$. The contact equations are solved using a simple adaptive quadrature routine. This value corresponds to the most likely mean contact in the population.

For both models of our organisms, uptake and depuration is modelled by the ordinary differential equation

$$dC/dt = uM - \lambda C$$

where $u = 0.02$ is the uptake rate, a decay rate which is approximately λ =0.0059. The equation is solved numerically with a fourth order Runge-Kutta algorithm for the value of C given a contact mass, M, and an initial contaminant value or vector of values for C

Mutating sub-models

The individual-based representation requires no change to run in a mutating configuration, but the population-based representation must maintain a list of contaminant loads which are processed in exactly the same way a scalar might be processed in one of the simple configurations. In the mutating configuration, each instance of a model is assessed at the end of its time-step to determine whether a change in representation is appropriate.

When a population dis-aggregates into individuals, the set of individuals with contaminant loads corresponding to the entries in the list are created. Their locations are normally distributed within the population disk. Any shortfall in numbers is handled by creating individuals which have no contaminant load and positioning them in the same fashion. Once the individuals are created, the population model is allowed to terminate.

An individual joins a population by having its contaminant load added to the list the population maintains. The first step in the process is to determine if there is a population close enough to the individual. If not, an empty population is created. Once a population's contaminant load has been inserted into the population the individual is allowed to terminate. The population model itself does not really play a part in this transaction: the "import" call is never used directly by the population, rather it is the supervising scheduler which organises the transfer to and from individuals and populations.

2.6 Results

The data presented in section 2.6.1 are based on two sets of simulations representing forty individuals. The first set uses a circular plume and the second an elliptical plume. These data sets allow a comparison of run-times, the equivalence (or lack of equivalence) amongst the submodels, and that provide data to examine the robustness of the representations to changes in the configuration of the plume. To ensure that run-time comparisons are meaningful all of the simulations in the first set of trials were run on the same computer.

The first set is comprised of forty trials of the homogeneous individual-based model, corresponding trials of the mutating model, and a single run of the population-based model. The second set is comprised of eighty trials of the mutating model and a single run of the population-based model. The data in the first set of trials establishes the equivalence of the homogeneous individual-based model and the mutating model. We pool the data from the mutating and homogeneous individual-based runs from the first set to match the eighty runs in the second to compare the effect of the plume's configuration. The results with an elliptical plume were not consistent across the model representations.

The individual-based representation produces a time series of contaminant levels for each individual, while the population submodel produces a "mean load" across a group of entities. The mutating submodel sits between the two, sometimes producing individual time series and sometimes mean time series for varying parts of the population. We denote representations by a subscript $r \in \{i, m, p\}$, so that $C_{rkj}(t)$ is the contaminant load at t in time series, C, associated with individual j in trial k of representation r, $C_{rk}(t)$ is the mean at a time t over all the groups simulated in the indicated representation and trial, and $C_r(t)$ denotes the mean of $C_{rk}(t)$ across the k trials for the indicated representation. To compare the dynamics of the system we generate mean time series for each of the k trials in the individual-based and mutating sets, $C_{ik}(t)$ and $C_{mk}(t)$. We are careful to generate the correct mean in the mutating submodel from time steps which have a mixture of individual trajectories and mean trajectories from population-based representations. Each of the mean time series, $C_{rk}(t)$, corresponds to the mean contaminant load of the population, $C_p(t)$, produced by the population submodel; averaging them, that is constructing

$$C_r(t) = \frac{1}{k}\sum_{j=1}^{k} C_{rj}(t),$$

where r is one of 'i' or 'm', is equivalent to running many stochastic trials and averaging to fit the population submodel. Using $C_{ik}(t)$, $C_{mk}(t)$ and $C_p(t)$ we find the maximum value attained for each representation, $\hat{C}_r$. We are also interested in the mean value across time of each representation,

$$\bar{C}_r = \frac{1}{T}\sum_{t\in T} C_r(t)$$

2.6.1 Contaminant load correspondence between representations

Both sets, $\bar{C}_r$ and $\hat{C}_r$, are presented in Table 2.2.

Table 2.2: Maxima and Means

Series$_r$	$\hat{C}_r$	$\bar{C}_r$
C_i	0.1787	0.0390
C_m	0.1821	0.0392
C_p	0.1387	0.0350

These data suggest that the mutating representation is consistent with the homogeneous individual-based representation. The population-based representation seems to present markedly different mean and maximum values.

2.6.2 Contaminant load variability

We calculated measures of variability in the time series using the aggregated time series $C_{ik}(t)$ and $C_{mk}(t)$ and their respective means across the k trials, $C_i(t)$ and $C_m(t)$. We will take T to be the total number of time steps taken, and we take

$$\hat{\sigma}_{ab} = \max_{t\in[1,T]}\left[\frac{1}{k}\sum_{j=1}^{k}(C_{ak}(t) - C_b(t))^2\right]^{1/2}$$

and

$$\bar{\sigma}_{ab} = \left[\frac{1}{T}\sum_{t=1}^{T}\left[\frac{1}{k}\sum_{j=1}^{k}(C_{ak}(t) - C_b(t))^2\right]\right]^{1/2},$$

to be the maximum root mean square error and the average root mean square error. Clearly we can write $\bar{\sigma}_{rr}$ as $\bar{\sigma}_r$ without introducing ambiguity, and similarly for $\hat{\sigma}_r$. The values for these measure of variability are presented in Table 2.3.

Table 2.3: Deviations amongst the model runs with respect to a given mean

r.m.s.e.	$r = i$	$r = m$	$r = p$
$\widehat{\sigma}_{ir}$	0.0083	0.0084	0.0534
$\bar{\sigma}_{ir}$	0.0024	0.0024	0.0096
$\widehat{\sigma}_{mr}$	0.0090	0.0090	0.0538
$\bar{\sigma}_{mr}$	0.0024	0.0024	0.0096

The data here indicate that the variability about the mean is consistent in the two representations which use simulated individuals to estimate contact and uptake. This is what we would expect since the mechanisms of uptake and contact are the same. In contrast, the population's values suggest that the contact and uptake are quite different, and that this model does not perform in quite the same way.

2.6.3 Sensitivity to the shape of the plume

We will use the same notation as Section 2.6.2 for the data derived from the circular plumes, while we will add a prime symbol to the data derived from the elliptical plumes. Thus, the mean value time series for the mutating submodel with elliptical plumes would be denoted C'_m and the mean value of that time series is $\bar{C}'$.

There is a good correspondence between the means and deviations associated with the mutating model in the circular and elliptical plume scenarios, but there is much poorer correspondence in the population based results in the two scenarios. The data for the circular plume and for the elliptical plume are presented in Tables 2.4 and 2.5 respectively.

Table 2.4: Circular plume results

$Series_r$	$\hat{C}_r$	$\bar{C}_r$	StdDev	$r = m$	$r = p$
C_m	0.1738	0.0392	$\widehat{\sigma}_{mr}$	0.0088	0.0535
C_p	0.1387	0.0350	$\bar{\sigma}_{mr}$	0.0024	0.0098

The population based model is clearly more sensitive to the shape of the plume than the mutating model. It seems likely that the major driver of this difference is that the long axis of the plume (a region where the net contact will be higher) remains in close proximity to the centroid of population where the population density is greatest.

Table 2.5: Elliptical plume results

$Series_r$	$\widehat{C'}_r$	$\overline{C'}_r$	StdDev	$r = m$	$r = p$
C'_m	0.1856	0.0394	$\widehat{\sigma}'_{mr}$	0.0087	0.0616
C'_p	0.1763	0.0445	$\bar{\sigma}'_{mr}$	0.0025	0.0092

2.6.4 Run-time

Each run collected data regarding the amount of time spent in different parts of the submodel; predictably, most of the effort is in calculating contact and updating contaminant loads.

The optimisation of suppressing the contact calculations when a population is outside the area of potential contact seemed to make very little difference to the run-time of population submodel (about 3%). It seems unlikely to make a great deal of difference to the mutating submodel. In the case of the purely individual-based submodel, this sort of optimisation is likely to play a much bigger role; any penalty would be multiplied by the number of animals simulated.

The population submodel ran for 98.7 cpu seconds. This submodel is deterministic and the amount of cpu time used is very stable, so only a single run is considered for comparison. The purely individual-based submodels took just over a mean time of 4205 cpu seconds with a standard deviation of approximately 16 seconds and the mean of the mutating submodel's run time was 1157 cpu seconds with a standard deviation of slightly over 11 cpu seconds.

2.7 Discussion

In the example our objective is to produce time-series data associated with the contaminant load of the group. Our individual-based model is taken as the best model for capturing the contact that real organisms have with an intermittent plume, and the population based representation has a computational efficiency that the individuals lack. The case for swapping in the example model is reasonably clear: there is a distinct improvement in run-time with no apparent deterioration in the fidelity of the dynamics. It seems likely that the naïve population distribution may be introducing a systematic divergence from what we see as an accurate, but computationally intense, individual-based model.

The general case is not limited to the polar extremes of switching between individual-based models and populations. A niche may have many representations, each of which has a particular set of strengths and weaknesses. This adaptive approach would present the same scope for improvement in purely equation-based models, where rules of thumb might be replaced by first-order approximations, or by complex systems of differential equations. In a purely individual-based example, the depth of the representation of the individual might vary from a simple mass and location through to a level of detail which included the individual's metabolic rates and breeding characteristics.

2.7.1 State spaces

To make our population-based model compatible with the individual-based model we have to extend the population's state space and maintain additional information to preserve the essential parts of the individual representation that makes it valuable to us. This is basically posing the first of the enumerated question from section 2.2. The *significant* information which the individual-based representation possesses is embodied in the contaminant loads amongst the individuals which comprise the group. We assume that the role of their relative locations about the population's centre is not important over a large portion of the global state-space and that we can discard it when we move from individuals to populations.

The union of submodels' state-spaces can generally be decomposed into *processing sets* of state-variables. Partitioning the state variables in this way—particularly in advance—makes it easier to analyse and minimise the boundary effects associated with the transition from one representation to another. Within a representation, the state variables which are unique to it form a special subset. The subset can be divided into the variables which need to be maintained by other representations, which we call V_r, and the variables which do not which we will call U_r. In principle, the variables in V_r might be maintained by a routine which is common to them all. The variables in U_r are more complex: when some other representation is mutating to representation r, the values assigned to the variables in U_r should reflect the state implied by the state of the old representation.

In the example model, an individual's relative location is a member of this set. Variables which are maintained and used by more than one representation are the third major group. This group can be divided into the set which is used consistently across the submodels (W_r), and the group of variables which have different dynamics in the various representations (X_r). In our example case, the contaminant load level of an individual would belong to the set V_{ind}. Its location and velocity would be in U_{ind}, the current time for both individuals and populations belongs to W_r, and a list of contaminant loads for populations belongs to V_{pop}.

2.7.2 Heuristics

The example model has very simple dynamics: the plumes are always in the same place, the migration is very predictable, and the spatial domain an individual may explore is well contained. Implementing a heuristic for the model which efficiently decides when to move from one representation to another is very straightforward: *If we are close enough that an individual might encounter the plume if the plume were at its maximum, switch a population to a group of individuals. Conversely, if there is no chance that an individual heading straight toward the plume (backwards) will encounter it, move the individual to a "close enough" population, or create a new population to accommodate the individual.* The model was constructed so that any number of populations could be run, and the heuristic was framed so that there were no assumptions about the number of population agents and the size of the groups they represented.

In this model, the decision process was shared between the representations themselves and the controlling scheduler. The representations reported their "robustness" to the scheduler which would then decide how to act on the advice, either triggering a change in representation or not.

The goal is to have a complex ensemble of niches which will change representations under the aegis of the scheduler to optimise the global outcome. For this more general approach additional information is needed: the scheduler would incorporate information about what properties each of the current representations required from other niches in order to decide what the mix of submodels filling the niches ought to be.

2.7.3 Transitions

The transition from individuals to population and population to individuals involves the loss and reconstruction of fine-scale position data, which may be a source of error. In our example, we assume that we can reconstruct a plausible position for each individual from the population's distribution function because we know the typical distribution of individuals and there is no behavioural change associated with contaminant load. A contaminant that made an organism sluggish would skew the distributions of both the population as a whole and the distribution of intoxicated organisms within the population. In the example model, individuals were randomly located in this way with enough time to randomise their velocities and blur any artifacts resulting from the selection of their locations. This corresponds to the perception that the velocities and relative locations of the individuals are comparatively unimportant except as they related to the distribution of the population. When values of state variables are generated in the process of changing to another representation, they need to conform to the distributions of the representation they are leaving. If for some strange reason (like behavioural change) the distribution of individuals, for example, does *not* conform to the distribution associated with a coherent population, then additional steps need to be taken accommodate this when changing to a population-based representation.

In section 2.7.2, transitions between representations in the example model are mediated by the scheduler. Decisions to change representation must be based, in part, on whether the transition will increase or decrease the efficacy of the suite of representations as a whole. If the example model were more than a pedagogic tool, it would have been useful to assess the mean error introduced in the transition from population to individual and individual to population.

Our simulation was aimed at producing contaminant load results, but in this context our aim would be to track the mean position, variance and extrema of the distribution of individuals relative to the population. To do this we would perform a comparison similar to that of section 2.6 but using positional data rather than contaminant load. The results would indicate if there might be significant transition effects associated with the change in representation. This sort of testing should ideally be performed at a number of scales (temporal or spatial, for example) since the knowledge of how long it takes for the transition boundary effects to settle (if they do) should feed into the high-level managing scheduler. In a sophisticated system, a representation might be spun up in advance so that the boundary effects have settled before it takes over from a less efficient representation.

2.7.4 Errors

Estimating error and confidence is extremely hard in complex models. Not only are the abstract processes deeply connected, but there may be hundreds of thousands of lines of code[3] which may introduce error of their own. Confronted with the code of a large-scale marine ecosystem model, one might turn around and contemplate joining a monastery rather than try and track the error propagation through the system. When we look at making an aggregate model out of niches, we can make the set of each of the representations which fill the niche simpler than some chimera which tries to take the best bits of each of the candidate representations. With well isolated transition mechanisms, we side-step nests of conditional code-paths and can contain the potential sources of error we must analyse. Since transitions can be recorded (like other useful data), we can generate an indication of the likely level of confidence based on our understanding of the representations used in a simulation.

2.8 Conclusion

Interesting and unanticipated results have come from this experiment. The discrepancy between the data concerning elliptical and circular plumes suggests that, at least in contaminant work, we need to pay closer attention to the movement dynamics of individuals and the density functions of populations. More predictably, the run-times show that mutating configurations provide a reasonable means of increasing computational speed without sacrificing fidelity in appropriate situations. The simple example demonstrated that a model which changes the representation of the system according to its location in its state space could provide much better computational efficiency than a model with a constant representation with no loss of accuracy.

Increasing population and resource use often reduce our environment's resilience and there is a growing need to model larger, more complex parts of

[3]Ningaloo-InVitro is currently more than 233000 lines of C++, NWS-InVitro is slightly more than 118000 lines of C++ and Atlantis is more than 145000.

the system we live in. Techniques for this have been iteratively moving toward a more systematic approach: many models will optimise their run-time and accuracy by suppressing unnecessary calculation, models will split their time-steps according to what they are simulating, or perhaps even adaptively set an appropriate time-step or spatial scale.

This paper proposes that actually changing the representations to suit the different regions of the state space of the model could provide a better balance between computational efficiency and error.

In our experience of large scale marine ecosystem modelling, the size of the system considered is growing much faster than computational capacity. Even for small systems the possibility of adjusting the representation of submodels to optimise the accuracy of the model as a whole has great appeal. Mutating models may provide an effective means of concentrating the use of computational capacity where it is most needed.

The authors would like to thank the three reviewers whose advice and suggestions have made this paper much clearer and to the point. Their care and insight are deeply appreciated.

2.9 Epilogue to the paper

This paper shows that two of the basic problems discussed in Chapter 1, namely situations where the mixing assumption fails, and improving the run-time efficiency of a model without sacrificing fidelity, can be resolved by models which change their representations according to the state of the agents. The initial motivation for considering this kind of model was the recurring need to incorporate contaminant modelling in marine environments Lyne et al. [1994a], Gray et al. [2006a, 2014b].

The decision strategy in this model was unusually straightforward and simple; this was primarily to avoid obscuring the the central idea, since a more realistic treatment might both obscure the basic simplicity of the idea, and need a significantly more capable modelling system for the discourse. At the time, the mathematical and programmatic resources I could bring to bear on the matter were primitive.

A serendipitous coincidence came about soon after this paper was submitted. I became involved in what seemed to be unrelated work which sought to incorporate social survey data into systems-management models. This work lead to the development of a tree structure which is used in Chapter **??** and fully defined, later, in Chapter 4. The configuration of trees can closely follow the structural characteristics of the surveys and provide a robust mathematical means of allowing simulated individuals with different attitudes, organisations or events to exert influence (positively or negatively) the attitudes of others in the system. The mathematical work presented later arose from that project, and during its development it became evident that this could provide a means of encoding complex relationships within a modelling system.

CHAPTER 3

Adaptive submodel selection in hybrid models

3.1 Prologue to the paper

This chapter is a (verbatim) inclusion of a paper that I was invited to submit for a special topic issue of the refereed journal *Frontiers in Environmental Science*. The issue's intent was to focus attention on hybrid models which couple component-models (submodels) from across the range of modelling paradigms, and to encourage the development of this sort of model in the context of ecological research. The paper develops a thought model which demonstrates a high level mechanism for governing the mix of representations used in the model based on the state of the system and the states of its components.

Simple rules can be used to govern transitions from one representation to another (Chapter 2), but these local transitions may degrade the quality of the simulation as a whole. The biomass of grass a farm's paddocks is probably quite adequately represented by a single number if there is little or no grazing, but with more livestock we might need to represent each paddock's biomass individually, . . . and possibly each paddock as a finely resolved field showing where the animals graze most intensely.

The appendix to the paper is an early version of the mathematical machinery developed in Chapter **??**. Since publication, the basic mathematical structure has changed significantly — this early version is much harder to manipulate, and generally less powerful. The propositions and assertions in this chapter are presented largely without proof, though the corresponding propositions for the work in Chapter **??** are proved in the material in Chapter **??**.

Adaptive submodel selection in hybrid models

Randall Gray[1]
University of Tasmania
Simon Wotherspoon
University of Tasmania

Abstract

Hybrid modeling seeks to address problems associated with the representation of complex systems using "single-paradigm" models: where traditional models may represent an entire system as a cellular automaton, for example, the set of submodels within a hybrid model may mix representations as diverse as individual-based models of organisms, Markov chain models, fluid dynamics models of regional ocean currents, and coupled population dynamics models. In this context, hybrid modelers try to choose the best representations for each component of a model in order to maximize the utility of the model as a whole.

Even with the flexibility afforded by the hybrid approach, the set of models constituting the whole system and the dynamics associated with interacting models may be most efficient only in parts of the global state space of the system. The immediate consequence of this possibility is that we should consider adaptive hybrid models whose submodels may change their representation based on their own state and the states of the other submodels within the system.

This paper uses a simple example model of an artificial ecosystem to explore a hybrid model which may change the form of its component submodels in response to their local conditions and internal state relative to some putative optimization choices. The example demonstrates the assessment and actions of a "monitor" agent which adjusts the mix of submodels as the model run progresses. A simple mathematical structure is also described and used as the basis for a submodel selection strategy, and alternative approaches are briefly discussed.

3.2 Introduction

The case has been made for developing systems with submodels that change their representation according to their state. Vincenot et al. [2011b] identify

[1]Published in *Frontiers in Environmental Science*, 20/8/2015
Corresponding author:`Randall.Gray@limnal.net` (Randall Gray),
`Simon.Wotherspoon@utas.edu.au` (Simon Wotherspoon)

reference cases describing the major ways system dynamics models (*SD*) and individual-based models (*IB*) can be coupled. Their final case, *SD-IB* model swapping, is exemplified in the models described by both Bobashev et al. [2007] and Gray and Wotherspoon [2012]. These papers argue that we can improve on conventional hybrid models, in terms of efficiency, fidelity, model clarity or execution speed by using an approach that allows the submodels themselves to change during a simulation. The last two papers implement simple models which demonstrate the approach, with correspondingly simple mechanisms to control transitions between different submodels.

Some authors argue that the explicit coupling of *SD* models and *IB* models may provide greater clarity and resolution in modeling [Vincenot et al., 2011b, Fulton, 2010]: parts of a model that are most clearly the result of aggregate processes are likely to be better suited to modeling with a *SD* approach. In contrast, the parts of a system where individuals have a significant influence on their neighbors [Botkin et al., 1972a] are better suited to an *IB* approach. This argument is closely tied to the notion of model fidelity. Following DelSole and Shukla [2010], we take *fidelity* to be the degree to which a model's trajectory is compatible with real trajectories. If our immediate goal is to maximize the utility of the set of submodels within a model as it runs, this must include the fidelity of the system in the decision process.

Measuring or estimating execution speed and numerical error are comparatively straight-forward, but determining model fidelity is not. Models with a high degree of fidelity should produce results which are consistent with observed data from real instances of the system they model across both a wide range of starting conditions and under the influence of ad hoc perturbations, such as fires through a forested domain. Model fidelity is addressed by DelSole and Shukla [2010] in the context of seasonal forecasting models. They explore the relationship between fidelity and skill using an information-theoretic approach. They describe *skill* loosely as the ability to reproduce actual trajectories, and they describe *fidelity* as measuring the difference between the distribution of model results and the distribution of real world results. They highlight the attractiveness of mutual information and relative entropy as measures (or at least indices) of skill and fidelity, but they observe that in their domain, climate modeling, the necessary probability distributions are unknown.

The issues of fidelity and the attendant cost/benefit balance are central to the discussion in Bailey and Kemple [1992]. This paper assesses the costs and benefits of three different upgrades to an existing model designed to help determine the best mix of types of radios used in a military context; their objective is to prioritize implementation of the refinements of their model. The fundamental issues they address are substantially the same as issues that influence dynamic model selection.

The paper by Yip and Marlin [2004] compares three models used for real-time optimization of a boiler network: simple linear extrapolation from the system's current state, quadratic prediction with the coefficients based on historical data and updated at every step, and a detailed process model that corresponds closely with the physical elements of the modeled system. Their conclusion correlates the fidelity of the model with its ability to control the real-time optimization of the system. They explicitly note that there are real costs associated with the increased fidelity. These costs include model development and the need for more expensive sensors. They note that increasing fidelity in the model enabled the system to adapt to changing fuel more efficiently, and that when there were frequent changes in fuel characteristics the simpler models performed poorly.

The projects described in Little et al. [2006] and Fulton et al. [2011] both used hybrid models as a means of decreasing the run-time, and increasing the fidelity of the modeled contaminant uptake in simulated organisms. This was accomplished by mixing individual-based submodels and regional population-based systems models. Gray and Wotherspoon [2012] explicitly used changes in the representation of agents to improve the execution speed of a contamination tracking model, without losing the fidelity of the individual based uptake model. In this paper we will develop a more general strategy which may be appropriate for more complex systems.

3.3 Model organization

For clarity, we will take the term *niche* to refer to something in the model which could be modeled in several ways: a "porpoise" niche could be filled by many instances of an individual-based model, models of pods, or a regional *SD* model of the porpoises. This is essentially the same as the term *component* in Vincenot et al. [2011b]. The motivation for departing from this convention arose from confusion resulting from inadvertently using *component* both in a technical and non-technical sense. The close analogy between the nature of a niche in an ecosystem and the nature of a component as discussed in Vincenot et al. [2011b] suggested the choice of *niche*.

Each of the alternative ways of representing a niche can be viewed as a *submodel*, and the word *representation* will be used to reflect a particular choice of submodel within a niche. An explicit instance of a submodel (such as a specific pod or an *SD* model) will be referred to as an *agent*. The *configuration* of the model at any moment consists of the particular set of submodels which fill the niches that comprise the model as a whole. For an adaptive hybrid model, there may be a large number of possible configurations and the selection of a "best" configuration is a complex matter.

Each agent running in a model must necessarily have data which can serve to characterize it for these assessments. This data would typically be some subset of its state variables, but the data alone may not be enough to base an assessment on: there may also be extrinsic data which play a role in a particular submodel's or agent's activity and impinges on its suitability. Then, the characterization of an agent — its *state vector* — is an amalgam of its own state and the state of other niches it interacts with, and it can be regarded as a point in the state space which the submodel is defined over.

A corresponding set of data characterizes a niche in the model; here, it is typically some appropriate aggregation of agent-level state variables (a biomass-by-size distribution, for example), relative rankings of the suitability of agents and alternative submodels, and indications of what extrinsic support all of the various alternatives require. This niche-level state vector provides the data needed for optimizing the configuration globally, and for managing the configuration when niche-wide effects become significant, for example, for an incipient epidemic.

Thus, there are three distinct levels of organisation which may influence the considerations regarding the current configuration, and inform any decision about what may need to change, namely

1. agent-level data need to be examined to determine how well suited each agent is to its current state and the context provided by the agents it interacts with,
2. a niche-level assessment which compares the utility of each of its current agents within a niche with their alternative submodels, and
3. a model-wide assessment which determines whether there are cross-agent conflicts or unmet needs arising from a particular configuration.

The state vectors which form the domains of submodels and niches are loci in appropriate state spaces and can be encoded as an elements in appropriate vector spaces. The mathematical tools to manipulate these state vectors can then be applied to calculate the distances between two states, the similarity of loci which represent models or niches, or to identify trends or clusters.

3.3.1 Implications of changing configurations

At a basic level, hybrid models are designed to represent entities or processes in the real world in a way which brings more clarity, efficiency, or fidelity that may be possible with more traditional approaches. Adaptive hybrid models, implicitly acknowledge that the appropriate representation may change through time. An important consequence is that when a submodel in a niche changes, it may trigger changes in representation elsewhere in the model.

We might consider an example where an *SD* submodel which represents the prey for an *SD* based predator changes to *IB* submodels. It seems reasonable to expect the representation of the predator might follow suit. This may change the spatial resolution, the fineness of the "quantities" represented, and possibly the time steps associated with the predators and prey. Disparities in either of the first two are simple enough to deal with: modelers routinely use interpolation as a means of removing inappropriate edges, or generating subscale data, for example. Changes in an agent's time step can have a dramatic causal influence on the subsequent simulation.

Chivers [2009] discusses how individual-based models are sensitive to when state variables are updated. In his discussion, the issue arises as a result of when the probability of a predator-prey interaction is calculated relative to

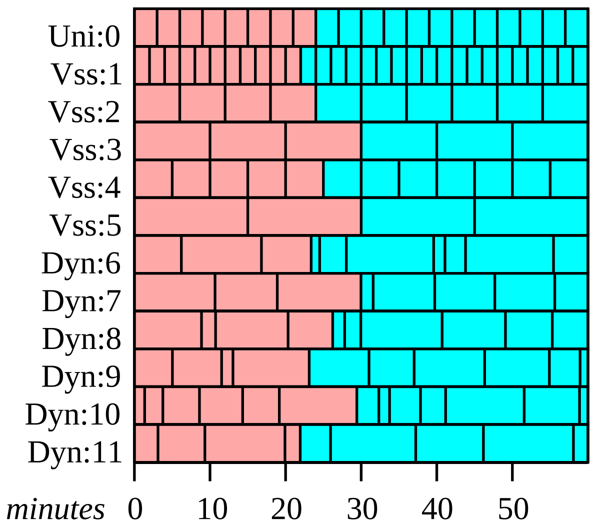

Figure 3.1: Time scheduling strategies. Red boxes represent time steps that have already passed, blue boxes represents scheduled time steps that have not yet been run. “Uni:” and “Vss:” submodels are members of a uniform or variable speed splitting submodels and require uniform time steps, and “Dyn:” submodels have adaptive time steps.

when the prey are removed from the system, though similar effects are also likely to occur in other contexts. The temporal sensitivity of submodels’ interactions needs careful examination in order to construct submodels that proceed through time coherently and interact correctly.

Multi-agent models must have strategies to manage the agents as they step from the start of the simulation to its end. The simplest method is to make everything within the model use the shortest time step required. This is computationally inefficient in a heterogeneous model.

A better approach is the technique of variable speed splitting, such as in Walters and Martell [2004] and many others. (Figure 3.1) This approach allows models to step through time in different intervals by dividing the largest interval required into smaller steps that are more appropriate for the submodels with naturally shorter time scales. While models with uniform time steps are a trivial example of this approach, variable speed splitting is almost as simple and much more efficient. This technique can keep the subjective times of a set of agents moderately consistent, but ad hoc stepping changes would still seem to be awkward or difficult.

Both of these strategies may be subject to artifacts arising from the sequence in which agents are given their time step. The general class of model errors of the sort described in Chivers [2009] arise as a consequence of structure of the processing across the set of agents in a simulation. *IB* models which process agents species-by-species will be particularly vulnerable to these sorts of artifacts, since there will be an implicit advantage or disadvantage to being early in the list. Similarly, advantage or disadvantage can arise when there is a change in representation, perhaps from an *SD* submodel to an *IB* submodel; a shorter time step in this situation may introduce a great many small time steps which agents may exploit. This kind of problem can be overcome by introducing a randomizing process within each time step. Early versions of the variable speed splitting model in Lyne et al. [1994a] suffered from predator-prey artifacts arising from a naïve introduction of predators and prey into the list of agents, and such randomizing was introduced to minimize the effects. In situations where the time steps of the interacting agents differ, implementing a randomization strategy may require a significant increase in the complexity of the system to accommodate irregular stepping through the lists of agents, or a significant change in the basic structure of the model.

Gray et al. [2006b] and Fulton et al. [2009] describe models that have a well developed approach to coordinating agents using adaptive time steps. In these models agents may set their own time steps to intervals that are suitable for their current activity or role. This strategy can readily incorporate submodels with uniform time steps, or collections that employ a variable speed splitting strategy. When agents interact, they either explicitly become synchronous before interaction occurs by setting their time steps appropriately and waiting, or they implicitly acknowledge that there is a temporal mismatch. (Figure 3.1)

While some agents should be given execution priority (such as an agent which models ocean currents), most agents will have their execution order within a time step randomized, effectively preventing a large class of execution order dependent artifacts. The associated overhead in the most recent work, Gray et al. [2006b], Gray and Wotherspoon [2012], is marginally higher than one would expect from single-stepping or variable speed stepping systems, but the advantages arising from the ability to ensure synchrony and change time steps in response to environmental stimulus outweigh the small computational overhead. This last approach seems likely to be the most appropriate for a general hybrid model that supports swapping models.

General adaptive hybrid models must have a mechanism for scheduling each agent's execution which keeps the cohort of agents roughly synchronous, and it should able to handle changes in an agent's time step when the agent changes its representation; where possible, agents should also be designed so that they may run at other time steps as well as their own preferred time step so they can become synchronous and interact at the appropriate temporal scale with other agents.

3.3.2 Systematically adjusting the model configuration

A model's configuration should only change when there is an overall benefit in the efficiency or fidelity of the system. A straightforward way of determining this is to have a monitoring routine that runs periodically, polling the agents, and ranking likely configurations according to their relative benefit or cost. This means that each submodel would need a way to provide, to the monitor, a measure of its current suitability, and to indicate what it needs from other niches.

Algorithm 1 Basic processing pass for the monitor

```
for all niches do
    for all submodels in the niche do
        for all agents in the submodel do
            generate agent state vector
            generate the submodel state vector
                note extrinsic requirements
        end for
    end for

    generate niche state vector
end for

Run niche-level assessment
Flag any whole of model issues
for all candidate configurations do
    Deprecate untenable configuration
    Adjust for unavoidable extrinsic
        requirements
end for

Select best indicated configuration
```

The last step in Algorithm 1 is deliberately vague.

Algorithm 1 illustrates a possible assessment pass for a monitor, though how appropriate it may be is an open question. Configuration ranking for the example model will be cast in terms of evaluating an objective function based on elements of the vector space of tree elements described in the Appendix.

A monitor may have large number of potential candidate configurations, but we would like to keep the actual number quite low. The example model described below has a global domain associated with a particular representation, along with local domains (subregions of the global domain) which are associated with finer scale representations of the modeled entities. The set of potential candidate trees could be quite large; in practice we reduce the number by casting the candidate trees in a more general way—including trees representing particularly good representations and particularly poor representations: the first to steer the configuration toward good choices, and the second to drive it away from poor choices. We can use the hierarchical organisation (whole-model, niche, submodel, agent) to help limit our search space, as well as the geographic context of the agents (whole-domain, local cell, immediate-locus).

The sets of candidate trees which are associated with particular configurations will need to be crafted carefully as a part of the model design. These trees reflect the modelers understanding of the strengths and weaknesses of each of the submodels (or sets of different submodels) which may be employed.

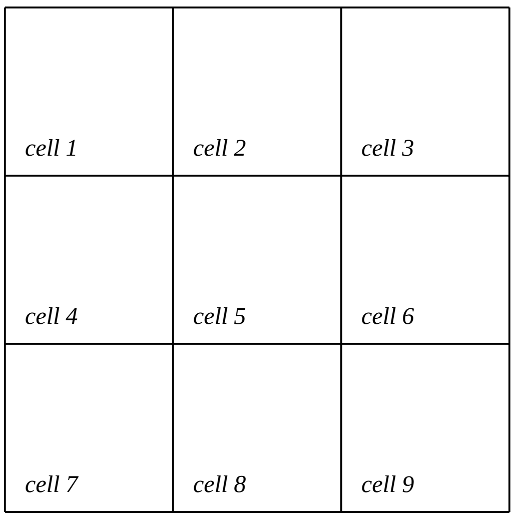

Figure 3.2: The model domain is divided into nine cells. An *SD* agent is associated with each of these cells and with the domain as a whole. Any *IB* agents which are created during the simulation will be associated with one cell at any given time.

Exactly how a monitoring routine is integrated into the model framework is a subjective choice best left to the team implementing the models, but one very attractive option is to implement the monitor as an agent in the system. This would allow the monitor to assess its own performance and the needs of other agents with respect to its own suitability with the option of swapping itself our for a montitor which implements some alternative strategy.

3.4 The example model

The purpose of the example model described below, is to provide a context for a discussion of the dynamics associated with a hypothetical simulation using this model. The ends of the spectrum between *SD* models and *IB* models are represented, and the environment is unrealistically simple in order to keep us from being swamped by detail.

The model consists of a spatially explicit environment that is partitioned into nine cells (Figure 3.2). The biotic elements consist of plants, fruit, seeds, herbivores, and carnivores. The herbivores feed on the plants and their fruit; and carnivores prey upon juvenile herbivores. The plants and herbivores are interdependent: fruit is the sole diet for juvenile herbivores and the plants need juvenile herbivores to make the seeds viable by eating the fruit.

The representations are equation-based *SD* models of the interactions between the plants and animals and *IB* models for plants and animals. The *SD* submodels model the biomass with respect to size, for plants and animals, or simply numeric quantities for fruit and seeds, and they can operate at either the global or cell-sized scale. Modeling biomass in this way makes it possible to minimize the loss of fidelity incurred by swapping from *IB* agents to *SD* agents and visa-versa, since we preserve more of the essential nature of the populations. A more detailed description of the *SD* agents is presented in the *Supplementary Material*.

Fruit and seeds

Fruit and seeds are treated somewhat differently to the rest of the niches. They exist principally as numbers of entities that are updated as a result of the activities of other, more explicit *SD* or *IB* models. There are explicit routines that deal with uniquely "fruit" and "seed" processing to handle spoilage and germination, respectively.

For fruit and seeds we have the following relationships

$$dN_{\mathrm{F}}(t) = \mathit{Production} - \mathit{Spoilage} - \mathit{FruitEaten}$$

and

$$\begin{aligned} dN_{\mathrm{S}}(t) = {} & s * \mathit{FruitEaten} \\ & - \left(1 - \frac{N_{\mathrm{P}}(t)}{K_{\mathrm{P}}}\right) \mathit{Germ}. \end{aligned}$$

where $N_{\mathrm{P}}(t)$ is the biomass of plants at time t, and K_{P} is the carrying capacity of the pertinent domain (either global or cell-based). The processing for fruit

Algorithm 2 Basic processing pass for fruit

```
$N_{\mathrm{F}} \leftarrow N_{\mathrm{F}} - (\mathit{Spoilage}_{\mathrm{F}} \cdot N_{\mathrm{F}})$
```

is quite simple and consists only of applying "spoilage"; no reference to other agents in the system is required, and only the number of fruit is adjusted as a result (Algorithm 2). Seed models will adjust their "seed count" as well as

Algorithm 3 Basic processing pass for seeds

```
$\mathit{NewTreeCount} \leftarrow \mathit{Germination} \cdot \mathit{SeedCount}$
$\mathit{SeedCount} \leftarrow \mathit{SeedCount} - (\mathit{NewTreeCount}$
    $+\mathit{Spoilage}_{\mathrm{S}} \cdot \mathit{SeedCount})$
generate $\mathit{NewTreeCount}$ new plant agents and introduce them into the system
```

the biomass distribution for plants in their time step, according to the level of germination. Germination is probabilistic as is the size of the plant a germinated seed becomes in its pass, though the distribution of possibly sizes is quite restrained (Algorithm 3).

SD representations

Each of the niches has an integral equation expressing the change in biomass for a given size; an animal's equation is of the form[2]

$$\begin{aligned} dN_{\mathtt{A}}(t,x) =& \mathit{Growth\&Starv} + \mathit{Repr} \\ & - \mathit{PredMort} - \mathit{NatMort}. \end{aligned}$$

We do not include migration terms in the *SD* models, since that will be addressed by the *IB* forms. The assumption is that the *SD* representation is most appropriate when population levels are moderately high, and there is adequate food; under these conditions, we will assume that the net migration associated with a domain will be close to zero.

Plants are represented by similar equations, namely

$$\begin{aligned} dN_{\mathrm{P}}(t,x) =& \left(1 - \frac{N_{\mathrm{P}}(t)}{K_{\mathrm{P}}}\right)[\mathit{Growth} + \mathit{Germ}] \\ & - \mathit{PredMort} - \mathit{NatMort} \end{aligned}$$

where $N_{\mathrm{P}}(t,x)$ is the biomass of plants of size x at time t.

The important state variables for the *SD* are, for each domain, the biomass-by-size distributions for plants, herbivores and carnivores, and the raw numbers of fruit and viable seeds.

Algorithm 4 Basic processing pass for the *SD* models

```
for all agents in this domain do
    Incorporate quantities that are
        controlled in other agents
    Run Runge-Kutta4
    Update only quantities that are
        controlled by this agent
end for
```

The system of equations described in the *Supplementary Material* is evaluated using a fourth order Runge-Kutta algorithm; the numbers of fruit and seeds, and both the global and cell-based biomass distributions for plants and animals are updated at the end of the calculation. The model will adjust the values in the global and cell-based models to allow data from models running with better resolution (usually more localized models) (Algorithm 4) to take precedence.

Most of the important parameters and many of the functions associated with the life history of the modeled entities are not specified. This way we may consider possible trajectories without being tied to a particular conception or parameterization of the system.

[2]See the *Supplementary Material* for a more detailed set of equations.

IB representations

Individual-based representations for plants, herbivores and carnivores follow the pattern in Little et al. [2006]; fruit and seeds are only modeled in the *SD* representation, though their numbers are modified by the activities of the herbivores irrespective of how those herbivores are represented.

3.4.1 *IB* Plants

Plants maintain a reference to their cell, their location, a mass and a peak mass. If a plant's mass drops below a certain proportion ($P_{M\Omega}$) of its peak mass, it dies — this provides a means for the herbivores to drive the plant population to local extinction.

We will suppose that plants grow according to a sigmoidal function with some reasonable asymptote and intermediate sharpness; fruiting occurs probabilistically as in the *SD* representation.

Algorithm 5 Basic processing pass for plants

```
if $(Mass \geq P_{Mature}) \wedge (P_{Fruits} \geq \mathrm{rnd}_{0,1})$ then
    ADDFRUIT($P_\rho Mass^{\frac{2}{3}}$)
end if
if $(Mass \leq P_{M\Omega}\, PkMass) \vee (\Omega_{\mathrm{indP}} < \mathrm{rnd}_{0,1})$ then
    DIE
else
    $Mass \leftarrow \Gamma_P(\delta t, Mass)$
    if $Mass > PkMass$ then
        $PkMass \leftarrow Mass$
    end if
end if
```

The plant agent goes through the steps in Algorithm 5 in each of its time steps. In the algorithm, $\Gamma_P(\delta t, mass)$ is an analogue of the probability of a plant growing from one size to another from the *SD* representation, P_{Mature} is the parameter that indicates the mass a plant must be before it fruits, P_{Fruits} is the probability of a mature plant fruiting, and P_ρ is the amount of fruit relative to the fruiting area. The routine ADDFRUIT updates the models representing fruit in the domain.

3.4.2 *IB* Animals

Like the plants, animals maintain a reference to their cell, their location, and a mass. They also maintain several variables that are associated with foraging or predation, namely the amount of time until they need to eat (*Sated*), and the amount of time they have been hungry (*Hungry*).

Animals will grow while they do not need to eat and will only forage when they are hungry. Reproduction happens in a purely probabilistic way once the animal is large enough, and the young are not cared for by the parents.

Animal movement is constrained so that they will tend to stay within their nominated home cell, only migrating (changing their home cell to an adjacent cell) when food becomes scarce or if the population exceeds some nominated value and causes crowding.

The analogues of the mechanisms for growth and starvation in the *SD* representation are quite different to those of the *IB* version. In the *SD* models, starvation and growth occur as a result of the relative population levels of the consumer and the consumed rather than the local availability of food.

There are no real programmatic differences between the *IB* representations of herbivores and carnivores; their differences lie in their choices of food and the way their "time-to-eat" variable is initially managed. InDiViduAl-based, newborn carnivores begin with a long time till they need to eat. This reflects a reliance on some unmodeled foodstuff until they are large enough to prey on the juvenile herbivores. In contrast, the juvenile herbivores must begin eating fruit immediately, and only switch to foraging on plants when they are larger (but before they can reproduce). For both species, if the amount of time they have been hungry exceeds a particular value, H_Ω or C_Ω, the individual dies.

Algorithm 6 Basic processing pass for herbivores and carnivores

```
if $(\Omega_{\mathrm{ind}\mathtt{A}} > \mathrm{rnd}_{0,1}) \vee (Hungry \geq \mathtt{A}_\Omega)$ then
    DIE
end if
$PreyList \leftarrow \textsc{PreyPresent}_\mathtt{A}(Locus, Mass)$
if $Sated \geq 0$ then
    $Mass \leftarrow Mass + \textsc{Growth}_\mathtt{A}(mass, \delta t)$
else if $(Hungry \geq 0) \wedge (\mathrm{len}(PreyList) > 0)$ then
    $Sated \leftarrow \textsc{Eat}(PreyList, \mathtt{A}_{EatLimit}, mass)$
    $Hungry \leftarrow 0$
    $ForageCt \leftarrow 0$
else if $((Hungry \geq 0) \wedge len(PreyList) = 0)$ then
    FORAGE
    $ForageCt \leftarrow ForageCt + 1$
else if $(Hungry \geq \mathtt{A}_{moveT}) \vee \textsc{Crowded}_\mathtt{A}$ then
    $\textsc{Migrate}_\mathtt{A}(Locus)$
else
    if $(mass \geq \mathtt{A}_{RepSize}) \wedge (\mathtt{A}_{RepP} \geq \mathrm{rnd}_{0,1})$ then $\textsc{Reproduce}_\mathtt{A}(Locus)$
    end if
end if
```

So, if we take A to represent either carnivores (C) or herbivores (H) below, then the processing pass for an animal is shown in Algorithm 6, where $\mathtt{A}_{moveT}$ is the amount of time an animal can be hungry before it migrates, $\mathtt{A}_\Omega$ is the amount of time it takes for the animal to starve, $\mathtt{A}_{EatLimit}$ is the most the animal can eat as a proportion of its mass, $\mathtt{A}_{RepSize}$ is the minimum size an animal may breed at and $\mathtt{A}_{RepP}$ is the probability of reproducing. The routines PREYPRESENT$_{\mathtt{H}}$ and EAT$_{\mathtt{H}}$ have different cases for juvenile and adult herbivores, since juveniles prey upon fruit, and the seeds from the fruit they eat need to be accounted for in the appropriate places. There is a similar issue with juvenile carnivores. Their *preylist* will always be set to a value that indicates that they may eat as much as they like, and the corresponding call to EAT$_{\mathtt{C}}$ will handle this value appropriately.

3.4.3 The monitor and model dynamics

The following may be typical of the types of situations that could or should cause changes in the configuration:

- *Low population*—If, in an *SD* representation, the number of individuals filling a niche (either explicitly taken from a distribution, or estimated using a mean and a biomass) drops below a nominated value, then the biomass in that niche should be converted to *IB* agents representing those individuals. This type of change is motivated by the observation that at low population levels the assumption that we can treat the population as having uniform access to resources (or be uniformly available to predators) breaks down;

- *High population*—If a niche in a cell is represented by *IB* agents and the number of individuals exceeds a (higher) nominated value, the biomass those agents represent should be subsumed by the distribution in the local *SD* submodel. The change in representation is attractive here for two reasons: an equation-based representation will be much faster, and *SD* submodels are arguably simpler to calibrate;

- *Starvation risk*—If the mean amount of time an animal in a cell spends *hungry* in a cell exceeds half of $\mathtt{A}_{\omega}$ (or some other nominated time), the prey biomass must convert to *IB* agents if it isn't already so (bearing in mind that this isn't pertinent for fruit). This mean is calculated by averaging the means of each animal in the cell. If this is triggered, it indicates that the biomass of the prey species is sparse enough that homogeneity assumption is unlikely to hold;

- *Relative biomass*—If the biomass available for predation is represented in a local *SD* agent and its density drops below some proportion of the minimum required to support the predators in the domain, the prey species should convert its biomass into *IB* agents and, if the predator is represented by a *SD* agent, it should also convert to an *IB* form. If the biomasses are such that the effective predation rate is unsustainable, the mixing assumption is unlikely to hold.

The pertinent data for conditions will be periodically reported to the monitor through a set of status trees. The trees are able to represent single entities, nested entities and aggregates equally well, and can preserve structural information which may also be used in the comparison of these trees. One of the basic elements we can easily incorporate into a submodel's status tree is the agent's own assessment of its competence relative to its state-vector and its local conditions. This measure of "self-confidence" can probably be maintained at little computational cost for most agents, and may be the most significant component in a monitor's assessment. The *high* and *low* population level conditions can clearly be determined by the agent in question; it can set its level of self-confidence upward or downward as appropriate. *Starvation* can also be encoded in the relevant node of an agent's status tree, but since starvation alone may not indicate a problem with the way the entity is represented, it probably wouldn't reduce the value for its confidence.

A starvation trigger may usually arise as a natural consequence of the population dynamics, but it may also occur when there is a mismatch in representations which has not been adequately addressed in the design stage. The final condition based on the relative biomasses is one which properly lies in the realm of the monitor—it would be quite inefficient for each of the candidate animals to be querying their prey for available biomass, summing the result, and then noting the need for change.

The monitor will primarily use the confidence values associated with agents and their niches, and the distance from trees which describe the state of the model or its set of submodels to trees which describe "known good" configurations. With data obtained directly from the agents in the system and from alternative representations it generates status trees,

- $\check{\boldsymbol{\tau}}^{\Sigma}_{sn}$, is a candidate status tree tied to a specific configuration. The serial number, *sn*, ties it to a configuration with that serial number,
- $\check{\boldsymbol{\tau}}^{\Sigma}_{d}$, is a candidate tree which represents the current state of a domain,

  $\boldsymbol{\tau}^{\Sigma}_{t}$, an aggregate tree for the whole domain at time t,

  $\boldsymbol{\tau}^{\Sigma}_{SD(n),t}$, aggregate trees for each cell, $n \in \{1,\ldots,9\}$,

  $\boldsymbol{\tau}_{\mathsf{R}(i),t}$, specific status trees for each agent,

  $\boldsymbol{\tau}_{R,t}$, specific status trees for a representation R for each representation associated with a niche,

  and

  $\hat{\boldsymbol{\tau}}_{\mathsf{R}(i),t}$, candidate trees for all possible representations of each agent i,

at the beginning of each of its steps. The model may have a mix of *SD* and *IB* representations, and some of the trees will have to incorporate data from many agents ($\boldsymbol{\tau}^{\Sigma}_{t}$, any of the $\hat{\boldsymbol{\tau}}_{\mathsf{R}(i),t}$, and $\boldsymbol{\tau}_{R,t}$, for example). A candidate tree is a status tree which represents an alternative submodel in a niche, and candidate trees are generated for specific agents and for each niche. When the monitor begins to generate status or candidate trees for a given agent, it first looks to see if it has generated an appropriate tree already. If it finds one, it incorporates or adjusts the tree appropriately; perhaps by incorporating the agent's biomass and size into the tree's data. We will also denote the configuration of a domain (global or local) with $\check{\boldsymbol{\tau}}^{\Sigma}_{c}$ where c identifies the domain in question.

The monitor assesses the trees by calculating aggregate values of particular attributes, comparing the trees' divergences from allegedly ideal configurations, and by looking how uniform groups are—groups of individuals that are all very similar are good candidates for simpler representations.

We can calculate the average confidence value from any of these trees by evaluating

$$\frac{(\!|\overline{\mathrm{mask}}(\,\boldsymbol{\tau}, \mathit{confidence}, 0)|\!)}{\mathrm{supp}(\overline{\mathrm{mask}}(\,\boldsymbol{\tau}, \mathit{confidence}, 0))},$$

for example. The trees and functions to manipulate them are described in the Appendix.

Now let us consider what a simulation might look like. Figure 3.3 provides an overview of the configuration of the system as our hypothetical simulation runs. The model begins with eleven agents (not counting the monitor). The monitor runs its first step generating the status trees: $\boldsymbol{\tau}^{\Sigma}_{0}$, which characterizes the model in aggregate, $\boldsymbol{\tau}^{\Sigma}_{SD(0),0}, \ldots, \boldsymbol{\tau}^{\Sigma}_{SD(9),0}$, which record the aggregate state of the ten *SD* submodels, the aggregate status tree for the *IB* agent, $\boldsymbol{\tau}^{\Sigma}_{IB(0),[}9]$, status trees for the *SD* submodels: $\boldsymbol{\tau}_{\mathrm{SD}(0),0}$–$\boldsymbol{\tau}_{\mathrm{SD}(10),0}$, the status tree for the lone carnivore, $\boldsymbol{\tau}_{IB(11),0}$, followed by the trees which represent alternative agents: $\hat{\boldsymbol{\tau}}_{SD(0),0}$–$\hat{\boldsymbol{\tau}}_{SD(10),0}$ and $\hat{\boldsymbol{\tau}}_{IB(11),0}$. As mentioned earlier, there is only the single tree for agent 11 (the carnivore) since its alternative representation is embodied in $\hat{\boldsymbol{\tau}}_{SD(10),0}$. During the simulation a simulated fire will occur.

The first steps which must be taken before ranking of potential configurations is to find the sets of candidate trees which best approximate the current configuration at both the global and cell levels. We do this by calculating a similarity index or a distance which indicates how close each of the candidate trees are to the configuration of each of the domains. There are many ways we could do this: for an index which only considers structural similarity we might use something like the simple function

$$\mathrm{ssim}(\,c, \boldsymbol{\tau}_d) = \frac{\mathrm{overlap}(\,c, \boldsymbol{\tau}_d)}{\max(\|c\|_{\mathrm{T}}, \|\tau_d\|_{\mathrm{T}})},$$

but for a more comprehensive treatment which factors values which are incorporated into the candidate and status trees we might apply the $\Delta(,)$ or d functions described in the Appendix. The d function is a well-defined distance over the vector space of trees, while the $\Delta($,f)unction is an index of similarity that incorporates structural characteristics as well as the numerical distance between compatible subtrees. To refine such an analysis we could apply mask and $\overline{\mathrm{mask}}$ to select only the relevant parts of the candidate and status trees.

So to assess the configuration of a domain, we would use our chosen measure to construct a set of the results of applying an optimisation function, *opt*, to each of the candidate trees and their similarity to the current configuration. So

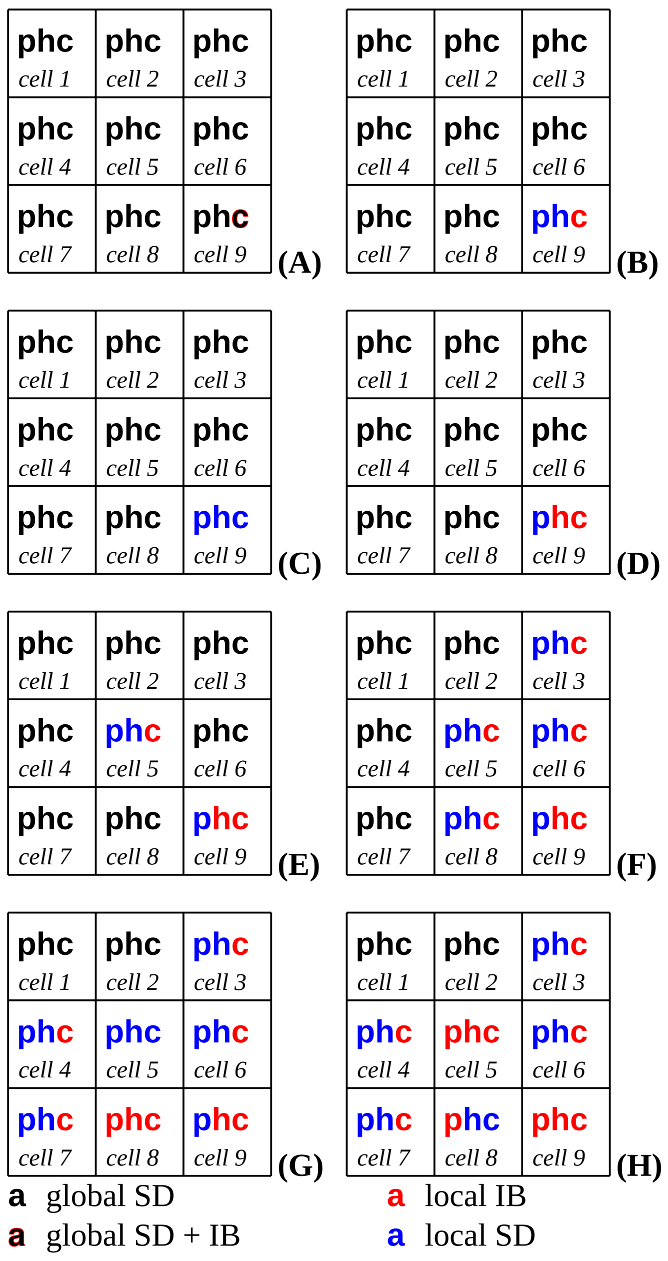

Figure 3.3: The color of the **p**,**h** and **c** indicate an agent's current representation within a cell at various points in the description of a simulation. In each, a black symbol indicates that the biomass of plants (**p**), herbivores (**h**) or carnivores (**c**) is modeled with the global *SD* agent, a blue symbol indicates that the biomass is modeled with a cell's *SD* agent, and red indicates that an *IB* model is being used. Symbols composed of two colors indicate that more than one representation is currently controlling portions of the relevant biomass.

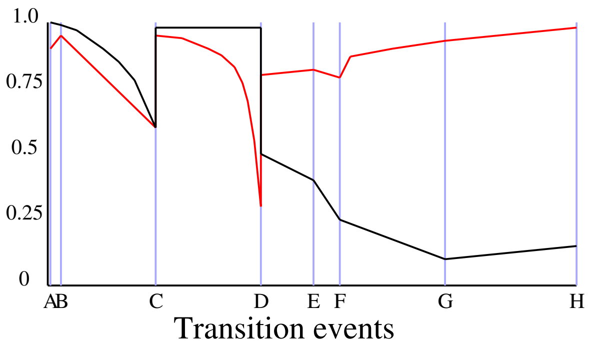

Figure 3.4: Normalized indexes of execution speed (black) and fidelity (red) the against configuration changes through time associated with Figure 3.3

if S is the set of all serial numbers for candidates, $\check{\tau}^{\Sigma}_{d}$ is the status tree fo the current domain, and is the, we calculate

$$(C) = \{(\delta(\check{\tau}^{\Sigma}_{d}, \check{\tau}^{\Sigma}_{i}), i) : \forall i \in \boldsymbol{S}\},$$

and this is used to generate

$$\boldsymbol{C}^{*} = \{(\mathrm{opt}(\check{\tau}^{\Sigma}_{i}), c, \check{\tau}^{\Sigma}_{i}, i) : \forall (c, i) \in \boldsymbol{C}\}$$

where δ stands for our chosen measure of similarity.

The elements in $\boldsymbol{C}^{*}$ are then assessed by the monitor, and the best permissible candidate is selected. If there is only a small improvement on the current configuration, $\check{\tau}^{\Sigma}_{d}$, the monitor will leave the configuration as it is; otherwise, the monitor would then manage the creation of new agents to replace less optimal representations and manage the exchange of state data.

So the early phase of our simulation might begin like so:

1. Both of the aggregate trees τ^{Σ}_{0} and $\tau^{\Sigma}_{SD(9),0}$ indicate that there is an *IB* agent in their domain and that their *SD* representation does not perform well for the indicated biomass. Both the status and candidate trees for agent 11, $\tau_{11(0),,}$ $\tau_{IB(11),0}$ and $\hat{\tau}_{IB(11),0}$, indicate that it is confident that it can represent the biomass, and that there are no immediate unmet requirements from other agents. Figure 3.3 (**A**)

2. The monitor assesses the trees against a prepared set of configurations: each of the alternative configurations (including the current configuration) is compared to a set of prepared, "efficient" configurations. The configuration of cell 9, $\check{\tau}^{\Sigma}_{9}$, notes global *SD* representations for plants and herbivores. This configuration is ranked lower than the alternative which has an individual based model for carnivores and a local *SD* submodel for the other entities in the cell. The monitor makes this change in configuration, and informs the global *SD* agent that it is no longer controlling the biomasses in cell 9. Figure 3.3 (**B**)

3. The model may run for some time without any change in configuration. Both the herbivores and carnivores breed. The increased execution speed between **C** and **D** in Figure 3.4 is a result of a change in representation: the number of carnivores, recorded in $\tau^{\Sigma}_{SD(C),t_4}$, reaches a point that prompts the monitor to convert them to an *SD* form. Figure 3.3 (**C**)

4. The biomass of carnivores has increased significantly by the time the model reaches **D** in Figure 3.4, and they are now eating all the young herbivores; as a result the carnivore population is now prey-limited, and the *Relative biomass* condition is triggered. Both the carnivore and herbivore populations are converted to *IB* representations. Notice that dynamics in the fidelity in Figure 3.4 around **D** arise from the collapse of the carnivore's prey, followed by the increase in fidelity after the representation change at **D**. Figure 3.3 (**D**)

5. A carnivore, agent 43, has been hungry ($Hungry \geq \mathtt{A}_{moveT}$) and has migrated to the cell 5 (noted in $\tau_{IB(43),t_5}$). As occurred in cell 9 at step 2, the monitor converts plants and herbivores in cell 5 to a local *SD* representation, with *IB* carnivores. Figure 3.3 (**E**)

6. A lot of activity has occurred in this monitor interval: a *Starvation risk* is triggered in cell 9 because too many of the carnivores are hungry (many of the $\tau_{IB(n),t_7}$ trees indicate that the elapsed time without eating is greater than *Hungry*). There has been more migration to cells 5,6 and 8 from cell 9 (more of the $\tau_{IB(n),t_7}$ trees indicate residence in new cells), and a chance migration has introduced a carnivore into cell 3 from cell 5. Cells 3,6 and 8 are converted to local *SD* and *IB* representations as happened in step 3. Figure 3.3 (**F**)

7. The population of carnivores in cell 9 crashes as a result of migration and the scarcity of prey, (reported in $\check{\tau}^{\Sigma}_{9}$) The *IB* juvenile herbivores are patchy and harder to find, so only a few carnivores are getting enough to eat. There will be many $\tau_{IB(n),t_{10}}$ which indicate hunger or death due to starvation. The monitor cleans up the dead agents. There are chance migrations from cell 5 into cells 4 and 7 (in $\tau^{\Sigma}_{SD(4),0}$ and $\tau^{\Sigma}_{SD(7),0}$). A fire begins in cell 8, moving through cell 5: biomass loss in all niches causes all niches to shift to *IB* representations. Figure 3.3 (**G**)

8. Juvenile herbivores are reappearing in cell 9, but the available plant biomass (recorded in $\tau_{\mathrm{SD}(9),t_{11}}$) has dropped due to reduced germination rates, triggering the *Relative biomass* condition in cell 9 causing the plants to convert to an *IB* representation. The fire in cell 8 has killed all animal biomass in the cell; they *do not* return to the global *SD* representation because their status trees diverge by too much. Instead, they convert to local *SD* representations (which represent zero biomass quite efficiently). Plants remain as *IB* agents The fire spreads to cell 5. Figure 3.4 shows a modest increase in execution speed between **G** and **H** due to the population losses associate with the fire. Figure 3.3 (**H**)

- . . . the simulation continues

3.5 Discussion

Adaptive hybrid models *can* be constructed so that each submodel is aware of its other representations and is able to change form as appropriate [Gray and Wotherspoon, 2012]. This approach requires each model to have a reasonably close coupling with its alternative representations, and the burden of instrumenting (and maintaining) the necessary code quickly becomes untenable in complex models. Worse, it removes the possibility of more subtle configuration management that can accept poor performance in one part of a system in exchange for much better performance elsewhere. It seems that a guiding principle should be that in an adaptive hybrid model, each representation should know only as much about the rest of the model as it *must* know, and no more. The facility for a submodel to delve into the workings of other submodels, or the workings of the model as a whole, decreases the clarity that hybrid modeling makes possible, and opens avenues for unwanted, unanticipated behavior.

The major argument in favor of closely integrated representations for submodels is that it makes common (or at least similar) state variables easy to maintain across representations, even in the face of many representation changes. It is an attractive arguement, but the long term consequence is an ever growing burden of code maintenance.

Constructing hybrid models isn't significantly more complex than constructing traditional models. Adaptive hybrid models of the sort described in this paper will require a more significant investment in the design of a monitoring routine, and in the crafting of appropriate sets of candidate configurations. The transition dynamics such a model will exhibit depend on the sets of candidate configurations, and it seems likely that a combination of analysis and experimentation may be the most effective way to develop a set of useful configurations. The hybrid models associated with Lyne et al. [1994a], Little et al. [2006], Fulton et al. [2009] were built by extending the repertoire of ways of representing elements of the ecosystem or the anthropic components rather than wholescale redesign and replacement.

We can imagine an ideal adaptive hybrid model, where any state information which must be passed on is accompanied by an appropriate, opaque parcel of code to perform the maintenance. As long as the monitor knows what information each of these maintenance interfaces needs, they can be updated each time the monitor interrogates the agent which has control of the state data. This is a readily attainable ideal: many programming languages support first class functions with closures, and these features are precisely what we need to address this problem. *Scheme, Python, ML, Common Lisp, Lua, Haskell*, and *Scala* all have first order functions with closures and, hence, the capacity to build model systems with this capability.

The state vectors and their supporting maintenance procedures can be treated as data and passed in lists associated with the status trees. If a monitor decides to swap representations, the accumulated lists of maintenance functions may be passed on to the new representation. A new representation inherits a maintenance list with variables that are part of its native state, it can claim them as its own and continue almost as though it had been running the whole time. In this way, a new representation doesn't need to know anything about its near kin, only that it must be able to run these black-box functions that come from other submodels, and to pass them on when required.

It may seem that this concentrates the global domain knowledge in the monitor, but this is not really the case. The monitor knows how to blindly query agents for state data and to the data in maintenance procedures. The monitor also knows how to recognize and rank characterizations of the states of the submodels or niches and to use those data to select a configuration.

The domain knowledge is encapsulated in the sets of targets the monitor matches the current configuration against, and in the heuristic triggers (such as *Starvation risk*) associated with a submodel or niche.

The essential problems any monitor is likely to deal with are problems of set selection (recognition, pattern matching...) and optimisation. These are common tasks: web searches, voice recognition, and route planning have become ingrained parts of modern society. Like route planning, the monitor needs to be able to reassess the "optimal" strategy as an ongoing process.

There are many options to choose from to rank configurations. A few of the likely candidates include

- using an objective function to evaluate each of the possible configurations,
- selecting a configuration based on decision trees,
- using neural nets to match model states and direct us to an appropriate configuration,
- using Bayesian networks to determine the most likely candidate,

and

- using support vector machines to select the target/configuration pairs.

In writing this paper, one of the vexing difficulties has been finding a suitable mathematical representation which would allow comparisons between configurations, submodel states and the states of niches. We need proxies that describe models and configurations of models in a way that we may readily understand, manipulate and reason about, and being able to deal with submodels which are, in themselves, adaptive hybrid models, seems to be a naturally desirable trait. The vector space of trees described in the Appendix has some nice properties, and may be directly useful with many of the options above: it forms a commutative ring (without necessarily having a unit), and would naturally inherit the body of techniques which only require the properties of such a ring.

3.6 Conclusion

There are still some major obstacles to developing a fully fledged adaptive hybrid model which is generic enough to tackle instances as varied as marine ecosystem modeling and urban planning. Foremost is a relative lack of real examples. The simulation of the hypothetical model[3] has tried to expose the character of an adaptive hybrid model which uses a monitor to manage the configuration of the system. There are parts of the description of the example system which are conspicuous by their absence; this is largely because they lie in almost wholly uncharted water. As a modeling community, we need to develop a wide range of approaches to how a model may assess the relative merits of a set of configurations. Many of the mechanisms we need for adaptive hybrid models already exist, but are found in domain specific models, and in wholly different domains, such as search engines and GPS navigation.

Establishing a suitable mathematical representation for model configurations which gives us access to well developed techniques for set selection, pattern recognition and component analysis would seem to be almost as urgent as adaptive hybrid examples of real systems.

Acknowledgments
The authors would like to thank two anonymous reviewers whose comments have improved the paper immeasurably. Thanks also go to a patient and understanding editor at Frontiers, and to Dr Tony Smith, who gave up a weekend to work a scientifically and grammatically fine toothed comb through the paper. The responsibility for any mistakes, awkward sentences, or places where it just does not make sense now rests completely with the lead author.

[3]The model described in this paper is currently under development and will be made freely available when it has been completed.

3.7 Appendix

3.7.1 Mathematical definitions

The trees we use are members of a normed vector space: we can add them, find out how far apart they are and interpolate between them. In principle, we can run clustering algorithms to find configurations that are similar, and identify when a model has left one cluster and entered another.

Definition 3.7.1. *Let $\mathcal{S}$ be a set of labels, and let $\mathbb{K}$be a field like the real or complex numbers. Then we define a node $\boldsymbol{n}$ as a triplet of the form $(s, v, \boldsymbol{E})$, with $v \in \mathbb{K}$, $s \subset \mathcal{S}$, and the set $\boldsymbol{E}$ is a (possibly empty) set of nodes of the same form with the restriction that no two nodes in $\boldsymbol{E}$ may have the same label in their first ordinate. We also define the triple $\mathbf{O} = (\varnothing, 0, \varnothing)$, which we will call the null tree, and define $\boldsymbol{T}$ to be the union of the set of all trees composed of a finite number of these nodes.*

The ordinates of $\boldsymbol{u} = (\boldsymbol{u}_\mathrm{P}, \boldsymbol{u}_v, \boldsymbol{u}_E)$ in $\boldsymbol{T}$ correspond to its label, value, *and* extension set. *An element of $\boldsymbol{u}_E$ will be called an* extension.

In our discussion, it will help to have a few more descriptive terms.

A node with an empty extension set is called a *simple* node, if this node happens to be a root node, then it is a simple tree.

Two trees, $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{T}$ are called *compatible* if either $\boldsymbol{u}_\mathrm{P} = \boldsymbol{v}_\mathrm{P}$ or at least one of $\boldsymbol{u}$ and $\boldsymbol{v}$ is $\mathbf{O}$.

We define the *depth* of a tree with:

$$\operatorname{depth}(\boldsymbol{u}) = \begin{cases} 0 & \text{if } \boldsymbol{u} = (\varnothing, 0, \varnothing) = \mathbf{O} \\ 1 & \text{if } \boldsymbol{u} \text{ is a simple node} \\ 1 + \max(\{\operatorname{depth}(\boldsymbol{v}) : \forall\, \boldsymbol{v} \in \boldsymbol{u}_E\}) & \text{otherwise.} \end{cases}$$

We will also define for $\boldsymbol{u} \in \boldsymbol{T}$,

$$\operatorname{trim}(\boldsymbol{u}) = \begin{cases} \mathbf{O} & \text{if } \boldsymbol{u} = \mathbf{O} \\ \mathbf{O} & \text{if } \boldsymbol{u} \text{ is simple} \\ (\boldsymbol{u}_\mathrm{P}, \boldsymbol{u}_v, \{\operatorname{trim}(\boldsymbol{e}) : & \\ \qquad \forall\, \boldsymbol{e} \in \boldsymbol{u}_E\} \setminus \{\mathbf{O}\}) & \text{otherwise.} \end{cases}$$

The cardinality of a tree is the number of nodes it contains. Specifically,

$$\|\boldsymbol{u}\|_\mathrm{T} = \begin{cases} 0 & \text{if } \boldsymbol{u} = \mathbf{O} \\ 1 + \sum_{\boldsymbol{e} \in \boldsymbol{u}_E} \|\boldsymbol{e}\|_\mathrm{T} & \text{otherwise.} \end{cases}$$

Simple nodes are the only nodes that have a cardinality of one, and $\mathbf{O}$ is the only node or tree with a cardinality of zero.

The support of a tree is the number of nodes which have a non-zero value.

$$\operatorname{supp}(\boldsymbol{u}) = \begin{cases} 0 + \sum_{e \in \boldsymbol{u}_E} \operatorname{supp}(\boldsymbol{e}) & \text{if } \boldsymbol{u}_v = 0 \\ 1 + \sum_{e \in \boldsymbol{u}_E} \operatorname{supp}(\boldsymbol{e}) & \text{otherwise .} \end{cases}$$

Related is the fundamental support of a tree, which only counts nodes with no zero valued nodes in their connection to the root node

$$\operatorname{fund}(\boldsymbol{u}) = \begin{cases} 0 & \text{if } \boldsymbol{u}_v = 0 \\ 1 + \sum_{e \in \boldsymbol{u}_E} \operatorname{fund}(\boldsymbol{e}) & \text{otherwise .} \end{cases}$$

Clearly the support of a tree, supp $\boldsymbol{u}$, must lie in the domain $[0, \|\boldsymbol{u}\|_{\mathrm{T}}]$ and fund $\boldsymbol{u} \le \operatorname{supp}(\boldsymbol{u})$.

The *overlap* between two trees is defined

$$\operatorname{overlap}(\boldsymbol{u}, \boldsymbol{v}) = \begin{cases} 0 & \text{if } \boldsymbol{u} = \mathbf{O} \text{ or } \boldsymbol{v} = \mathbf{O} \text{ or } \boldsymbol{u}_{\mathrm{P}} \neq \boldsymbol{v}_{\mathrm{P}} \\ 1 + \sum\limits_{\substack{e \in \boldsymbol{u}_E \\ f \in \boldsymbol{v}_E}} \operatorname{overlap}(\boldsymbol{e}, \boldsymbol{f}) & \text{otherwise} \end{cases}$$

Clearly two trees, $\boldsymbol{u}$ and $\boldsymbol{v}$, are compatible if and only if $\operatorname{overlap}(\boldsymbol{u}, \boldsymbol{v}) \neq 0$; they will be said to *completely overlap* if $\|\boldsymbol{u}\|_{\mathrm{T}} = \|\boldsymbol{v}\|_{\mathrm{T}} = \operatorname{overlap}(\boldsymbol{u}, \boldsymbol{v})$.

We can now define scalar multiplication, and tree addition.

Definition 3.7.2. *Take $a \in \mathbb{K}$ and $\boldsymbol{u} \in \boldsymbol{T}$, then*

$$a\,\boldsymbol{u} = \begin{cases} \mathbf{O} & \textit{if } \boldsymbol{u} = \mathbf{O} \\ (\boldsymbol{u}_{\mathrm{P}}, a\,\boldsymbol{u}_v, \{a\,\boldsymbol{f} : \boldsymbol{f} \in \boldsymbol{u}_E\} \setminus \{\mathbf{O}\}) & \textit{otherwise.} \end{cases} \tag{3.1}$$

Definition 3.7.3. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be compatible elements of $\boldsymbol{T}$.Then taking the symbol $+$ to be addition in the field $\mathbb{K}$, we extend it to addition in $\boldsymbol{T}$ so that for nodes $\boldsymbol{u}$ and $\boldsymbol{v}$,*

$$\boldsymbol{u} + \boldsymbol{v} = \begin{cases} \mathbf{O} & \textit{if } \boldsymbol{u} = \boldsymbol{v} = \mathbf{O} \\ \boldsymbol{u} & \textit{if } \boldsymbol{u} \neq \mathbf{O} \textit{ and } \boldsymbol{v} = \mathbf{O} \\ \boldsymbol{v} & \textit{if } \boldsymbol{u} = \mathbf{O} \textit{ and } \boldsymbol{v} \neq \mathbf{O} \\ (\boldsymbol{u}_{\mathrm{P}}, \boldsymbol{u}_v + \boldsymbol{v}_v, \varnothing) & \textit{if } \boldsymbol{u}_E, \boldsymbol{v}_E = \varnothing \\ \Big(\boldsymbol{u}_{\mathrm{P}}, \boldsymbol{u}_v + \boldsymbol{v}_v, \big(\{\boldsymbol{f} + \boldsymbol{g} : \boldsymbol{f} \in \boldsymbol{u}_E \textit{ and } \boldsymbol{g} \in \boldsymbol{v}_E \textit{ and } \boldsymbol{f}_{\mathrm{P}} = \boldsymbol{g}_{\mathrm{P}}\} & \\ \quad \cup \{\boldsymbol{f} : \boldsymbol{f} \in \boldsymbol{u}_E \textit{ and } \boldsymbol{f}_{\mathrm{P}} \neq \boldsymbol{g}_{\mathrm{P}} \forall \boldsymbol{g} \in \boldsymbol{v}\} & \\ \quad \cup \{\boldsymbol{g} : \boldsymbol{g} \in \boldsymbol{v}_E \textit{ and } \boldsymbol{g}_{\mathrm{P}} \neq \boldsymbol{f}_{\mathrm{P}} \forall \boldsymbol{f} \in \boldsymbol{u}\}\big) \setminus \{\mathbf{O}\}\Big) & \textit{otherwise.} \end{cases} \tag{3.2}$$

Definition 3.7.4. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be compatible elements of $\boldsymbol{T}$.Then we define inner-multiplication between the two nodes*

$$\boldsymbol{u} \cdot \boldsymbol{v} = \big(\boldsymbol{u}_{\mathrm{P}}, \boldsymbol{u}_v \boldsymbol{v}_v, \{\boldsymbol{f} \cdot \boldsymbol{g} : \boldsymbol{f} \in \boldsymbol{u}_E, \boldsymbol{g} \in \boldsymbol{v}_E \textit{ and } \boldsymbol{f}_{\mathrm{P}} = \boldsymbol{g}_{\mathrm{P}}\} \setminus \{\mathbf{O}\}\big) \tag{3.3}$$

It can be shown that $\boldsymbol{T}$ with scalar multiplication and tree addition forms a vector space. This isn't quite enough to give us distances between trees, however, so we define a semi-norm

Definition 3.7.5. *Let $\boldsymbol{u}$ be an element of $\boldsymbol{T}$.Then we can define a semi-norm over $\boldsymbol{T}$*

$$(\!|\boldsymbol{u}|\!) = \begin{cases} 0 & \text{if } \boldsymbol{u} = \mathbf{O} \\ |\boldsymbol{u}_v| & \text{if } \boldsymbol{u}_E = \varnothing \\ |\boldsymbol{u}_v| + \sum_{e \in \boldsymbol{u}_E} (\!|\boldsymbol{e}|\!) & \text{otherwise.} \end{cases}$$

It is clear that $(\!|\boldsymbol{u}|\!)$ will always be non-negative, and the only shortcoming is that we can have a node $\boldsymbol{u}$ with $(\!|\boldsymbol{u}|\!) = 0$, but $\boldsymbol{u} \neq \mathbf{O}$. In order to turn this into a normed vector space we take the set $\mathfrak{O} = \{\boldsymbol{u} : u \in \boldsymbol{T} \text{ and } (\!|\boldsymbol{u}|\!) = 0\}$ and we construct an equivalence relation on $\boldsymbol{T}$ by the rule $[\boldsymbol{u}] \equiv [\boldsymbol{v}]$ if and only if there exist $z_u, z_v \in \mathfrak{O}$ such that $\boldsymbol{u} + z_u = \boldsymbol{v} + z_v$. It can be shown that scalar multiplication, tree addition, and the semi-norm behave appropriately with respect to the equivalence classes. This means that if we identify elements of $\boldsymbol{T}$ with their equivalence class, then we can take $(\!|\boldsymbol{u}|\!)$ to be a norm and that it induces a distance function

$$\mathrm{d}(\boldsymbol{u}, \boldsymbol{v}) = (\!|\boldsymbol{u} - \boldsymbol{v}|\!).$$

Definition 3.7.6. *We define the functions* mask *and* $\overline{\text{mask}}$ *that set the values associated with particular nodes in a tree to $v \in \mathbb{K}$. Specifically, if $\mathcal{L} \subset \boldsymbol{T}$, then*

$$\text{mask}(\boldsymbol{u}, \mathcal{L}, v) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, v, \{\text{mask}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_E\}) & \text{if } \boldsymbol{u}_{\mathsf{P}} \in \mathcal{L} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\text{mask}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_E\}) & \text{otherwise} \end{cases} \tag{3.4}$$

and

$$\overline{\text{mask}}(\boldsymbol{u}, \mathcal{L}, v) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, v, \{\overline{\text{mask}}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_E\}) & \text{if } \boldsymbol{u}_{\mathsf{P}} \notin \mathcal{L} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\overline{\text{mask}}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_E\}) & \text{otherwise.} \end{cases} \tag{3.5}$$

mask$(\boldsymbol{u}, \mathcal{L}, 0)$ *would return a tree similar to $\boldsymbol{u}$, but all its nodes that have labels in $\mathcal{L}$ would have values of zero.*

We also define several functions which prune or select a child from a tree's extension set. This function returns only the part of $\boldsymbol{u}$ which overlaps $\boldsymbol{p}$,

$$\text{excise}(\boldsymbol{u}, \boldsymbol{p}) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\text{excise}(\boldsymbol{f}, \boldsymbol{g}) : \boldsymbol{f} \in \boldsymbol{u}_E \text{ and } \boldsymbol{g} \in \boldsymbol{p} \text{ and } \boldsymbol{f}_{\mathsf{P}} = \boldsymbol{g}_{\mathsf{P}}\} \setminus \{\mathbf{O}\}) & \text{if } \boldsymbol{u}_{\mathsf{P}} = \mathcal{L} \\ \mathbf{O} & \text{if } \boldsymbol{u}_{\mathsf{P}} \neq \boldsymbol{p}_{\mathsf{P}} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \varnothing) & \text{otherwise,} \end{cases} \tag{3.6}$$

and this one either returns an appropriately labelled child from the extension set (a branch), or $\mathbf{O}$.

$$({}_C\, \boldsymbol{u}, \ell) = \begin{cases} \boldsymbol{f} & \text{if } \boldsymbol{f}_{\mathsf{P}} = \ell \text{ and } \boldsymbol{f} \in \boldsymbol{u}_E \\ \mathbf{O} & \text{otherwise.} \end{cases} \tag{3.7}$$

We now finish with a definition of a function, $\Delta(,)$, that gives us a measure of the degree of divergence between two trees.

Definition 3.7.7. *The degree of deviation between two trees, $\boldsymbol{u}$ and $\boldsymbol{v}$ is given by the expression*

$$\Delta(\boldsymbol{u}, \boldsymbol{v}) = \begin{cases} \|u\|_{\mathrm{T}} + \|v\|_{\mathrm{T}} - 2\,\mathrm{overlap}(\boldsymbol{u}, \boldsymbol{v}) + (\!|\boldsymbol{u} - \boldsymbol{v}|\!) & \textit{if } \boldsymbol{u} \textit{ and } \boldsymbol{v} \textit{ are compatible} \\ \|u\|_{\mathrm{T}} + \|v\|_{\mathrm{T}} & \textit{otherwise.} \end{cases} \tag{3.8}$$

The rationale behind this definition is that if trees $\boldsymbol{u}$ and $\boldsymbol{v}$ are identical, then $\Delta(\boldsymbol{u}, \boldsymbol{v})$ will be zero. We also want nodes that aren't common to both trees to count as differences.

3.8 Epilogue to the paper

The model described in this paper has been used as a template for the model discussed in Chapter 5. The paper attempted to describe a credible "toy" model to provide a setting for the discussion. Unfortunately, when it came to implementation of the model the paper described, a few shortcomings came to light and some of the corresponding algorithms have been altered in the explicit model to correct them. The most notable example, and one which characterises them well, is that the description didn't allow for a cell which had lost all its plant biomass to be recolonised.

This paper extended and generalised the simple state maintenance of Chapter 2 by passing closures rather than a vector of values and the confident knowledge that the closures embodied the correct process for update. By using closures, the actual machinery supporting the update of the state of a previous representation can be passed and both the mechanism and data made, in some sense, opaque to all but the representation it is associated with. Enforcing the notion that the code required for maintaining a state vector of a representation must be provided *by* that representation means that all of the code which changes the state of an agent is embodied within the agent's representation. This has practical appeal in that inappropriate modifications to that state data may cause errors that would be very difficult to isolate.

The trees described in the next chapter are used to characterise the distribution of agents within the model's spatial domain, and encode the different representations and numbers of those agents.

CHAPTER 4

A non-unital ring of trees

4.1 Introduction

The project which triggered the development of the structure in this chapter considered using the modeling of the social dynamics associated with policies about, and responses to climate change and to predict the social and economic consequences of development arising from various scenarios. This was a significant change from previous representations of "public" participants, such as recreational fishers, tourists, accomodation and other small businesses, which were modelled in fairly simple ways (Fulton et al. [2011], Gray et al. [2014a]). The specific goal was to be able to incorporate a simulation of the way public opinion changes in response to policy actions and changes in the economy and environment. The starting point for this was a corpus of responses to a survey on attitudes associated with the topic of climate change (Boschetti et al. [2012]). The data consisted of (largely) numeric answers to individual questions which could represented as either distinct items, or as an aggregation of symbolic elements. Questions like "How much do trust the following individuals or organisations to tell you the truth about changing climate?" might be encoded in a symbolic way by aggregating the symbols climate_change, trust, information_source, AustPeng for the trustworthiness of the *Australian Penguins* as a source of information. The actual value marked could be encoded as a scalar, giving us an expression like "$4/5 + \mathsf{climate\_change} + \mathsf{information\_source} + \mathsf{trust} + \mathsf{AustPeng}$"—expressing it in this way suggested that working with more intricate relationships might be possible.

While my involvement in the project was shortlived, the problem the data posed was engaging, and slowly it became evident how the the mathematical structures I was trying to construct could represent configurations of models, and that it might be able to incorporate the interdependencies between models and other information in a very simple way. My hope was to be able to construct example configurations which were reasonable for particular conditions and to assess how close to "known-good" configurations the system was at any given point. It also seemed possible that it might allow strategies which were apble to interpolate between "good" configurations making intermediate transitions between configurations feasible. The natural network of dependencies exhibited by some components of past models suggested using tree structures to encode the configuration of a model as a starting point.

The structure described in this chapter is the structure behind the work in Chapter 3. The trees are comprised of nodes which have a label and a weight (and, of course, children). Initially, the labels were simply symbols associated with certain attitudes or semantic data which were taken from a nominated set. This had the advantages of simplicity and clarity, but their very simplicity made the multiplicative operator much more complicated. I realised that much of the machinery associated with labels was really just a messy version of arithmetic in commutative ring of multinomials. I believe that using the field of rational multinomials may be productive, but this notion has not been seriously explored. Once labels were identified with multinomials, everything became clear.

4.2 Conventions and preliminary definitions

Generally, we will use lower case, boldfaced symbols to denote a node (or tree), and upper case, boldfaced symbols to denote sets — particularly sets of nodes. Other symbols (such as x) will typically refer to numbers or multinomials. Elements of a node, $\boldsymbol{u}$ will be identified using an appropriate subscript, namely $\boldsymbol{u}_v$, for the node's value and $\boldsymbol{u}_{\mathrm{P}}$ for its label. Initially, the children of a node were thought of as refinements or children of an attitude, so the children of $\boldsymbol{u}$ were its children, and $\boldsymbol{u}_{\mathrm{C}}$ denotes the set of children. We will take $\mathbb{K}[\boldsymbol{A}]$ to be the ring of multinomials over the elements of a finite set of symbols $\boldsymbol{A}$. Here $\mathbb{K}$ would usually be some numeric field such as $\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$, for example.

Definition 4.2.1. *Given $\boldsymbol{A}$ and $\mathbb{K}[\boldsymbol{A}]$ and an arbitrary field, $\mathbb{K}$, we define the set, $\boldsymbol{T}$ of finite (acyclic) trees where each node is of the form $(\boldsymbol{u}_v, \boldsymbol{u}_{\mathrm{P}}, \boldsymbol{u}_{\mathrm{C}})$ where the value, $\boldsymbol{u}_v$, is a member of $\mathbb{K}$, it's label, $\boldsymbol{u}_{\mathrm{P}}$ is a member of $\mathbb{K}[\boldsymbol{A}]$ and the set of children, $\boldsymbol{u}_{\mathrm{C}}$, contains leaf nodes with distinct labels.*

Nodes or trees with no children, $\boldsymbol{E} = \varnothing$, will be called *simple nodes, simple trees*, or *leaf nodes*, and simple nodes which also have scalar multinomials as their labels may be referred to as *scalar nodes* or *scalar trees*. The domain of trees, $\boldsymbol{T}$, is the collection of only those trees with a finite number of nodes. form. We will make use of a special element in $\boldsymbol{T}$, $\mathbf{O} = (0, 0, \varnothing)$, which is an an analogue of zero that we will call the *zerotree*.

Note that the choice to use elements of the ring of multinomials for labels is, in a sense, arbitrary: elements of any commutative ring will serve, though we will see that if we use a commutative ring, $\boldsymbol{T}$ and the derived domains are also commutative rings,; here, the ring of multinomials provide a simple example which is easily manipulated and printed. $\mathbb{K}$ is taken to be $\mathbb{R}$ or $\mathbb{Q}$.

When there is no risk of ambiguity, we will use the same symbol to refer to a set, $\boldsymbol{T}$ for example, and a vector space based on that set. Compatibility (or lack thereof) is really only pertinent to the addition of trees, in the same way that having the same row- and column-rank is only necessary in matrix addition. There is no such constraint in the pairwise multiplication of trees.

First, the definitions for some basic tools for manipulating these trees. Some of the functions defined below are not used in this chapter, but play a role in the explicit model described in Chapter 5.

Definition 4.2.2. *We define the relation which partitions $\boldsymbol{T}$ into* compatible *sets of trees. Two nodes or trees, $\boldsymbol{u}$, $\boldsymbol{v}$ are said to be compatible if they have the same label or at least one of them is* $\mathbf{O}$

$$\sigma(\boldsymbol{R}) = \{\boldsymbol{R}_p : \forall\, \boldsymbol{u} \in \boldsymbol{R}, \boldsymbol{u}_{\mathrm{P}} = p \iff \boldsymbol{u} \in \boldsymbol{R}_p\},$$

which we may denote

$$\boldsymbol{u} \sim \boldsymbol{v} \iff \boldsymbol{u} \in \boldsymbol{R}_{\boldsymbol{v}_{\mathrm{P}}} \iff \boldsymbol{v} \in \boldsymbol{R}_{\boldsymbol{u}_{\mathrm{P}}}.$$

Compatibility is analogous to compatibility in matrices, and nodes may only be added if they are compatible.

Definition 4.2.3. *The cardinality of a tree is the number of nodes it contains. We define it formally as*

$$\|\boldsymbol{u}\|_{\mathrm{T}} = \begin{cases} 0 & \textit{if } \boldsymbol{u} = \mathbf{O} \\ 1 + \sum_{e \in \boldsymbol{u}_{\mathrm{C}}} \|\boldsymbol{e}\|_{\mathrm{T}}. \end{cases}$$

Simple nodes are the only nodes which have a cardinality of one, and $\mathbf{O}$ *is the only node or tree with a cardinality of zero.*

Definition 4.2.4. *For $\boldsymbol{u} \in \boldsymbol{T}$ we define the function*

$$\operatorname{depth}(\boldsymbol{u}) = \begin{cases} 0 & \textit{if } \boldsymbol{u} = (0, 0, \varnothing) = \mathbf{O} \\ 1 & \textit{if } \boldsymbol{u} \textit{ is a simple node} \\ 1 + \max(\{\operatorname{depth}(\boldsymbol{v}) : \forall\, \boldsymbol{v} \in \boldsymbol{u}_{\mathrm{C}}\}) & \textit{otherwise} \end{cases}$$

which gives us the depth of the tree.

Definition 4.2.5. *We will also define for $\boldsymbol{u} \in \boldsymbol{T}$,*

$$\operatorname{trim}(\boldsymbol{u}) = \begin{cases} \mathbf{O} & \textit{if } \boldsymbol{u} = \mathbf{O} \\ \mathbf{O} & \textit{if } \boldsymbol{u} \textit{ is simple} \\ (\boldsymbol{u}_v, \boldsymbol{u}_{\mathrm{P}}, \{\operatorname{trim}(\boldsymbol{e}) : \forall\, \boldsymbol{e} \in \boldsymbol{u}_{\mathrm{C}}\} \setminus \{\mathbf{O}\}) & \textit{otherwise.} \end{cases}$$

Obviously the depth is an indication of how many levels of nodes the tree possesses.

Trimming essentially removes all simple nodes from the tree. A recursive application of trimming will be denoted trim_k, indicating that the tree $\boldsymbol{u}$ will be trimmed k times. Note that $\operatorname{trim}_{\operatorname{depth}(\boldsymbol{u})} \boldsymbol{u} = 0$ and $\operatorname{depth}(\operatorname{trim}_{\operatorname{depth} \boldsymbol{u}-1} \boldsymbol{u}) = 1$.

Definition 4.2.6. *The* overlap *between two trees is defined*

$$\operatorname{overlap}(\boldsymbol{u},\boldsymbol{v}) = \begin{cases} 0 & \text{if } \boldsymbol{u} = \mathbf{O} \text{ or } \boldsymbol{v} = \mathbf{O} \text{ or } \boldsymbol{u}_\mathrm{P} \neq \boldsymbol{v}_\mathrm{P} \\ 1 + \sum\limits_{\substack{e \in \boldsymbol{u}_C \\ f \in \boldsymbol{v}_C}} \operatorname{overlap}(\boldsymbol{e},\boldsymbol{f}) & \text{otherwise} \\ \end{cases}$$

Clearly two trees, $\boldsymbol{u}$ and $\boldsymbol{v}$, are compatible if and only if $\operatorname{overlap}(\boldsymbol{u},\boldsymbol{v}) \neq 0$ *; they will be said to* completely overlap *if* $\|\boldsymbol{u}\|_\mathrm{T} = \|\boldsymbol{v}\|_\mathrm{T} = \operatorname{overlap}(\boldsymbol{u},\boldsymbol{v})$.

4.3 Scalar Multiplication and addition

The aim of this work is to be able to compare trees in a robust way and to manipulate them as though they were vectors: trees which form a vector space or, better, a metric space can be compared and clustered. We will start by defining scalar multiplication of the trees in $\boldsymbol{T}$, and then we will define a few useful mappings which will help keep the expressions simple. Our aim, in this section, is to define addition, and to show that the defined scalar multiplication and addition make this a vector space. When there is no risk of ambiguity, we will use the same symbol to refer to both the set and a vector space based on that set.

4.3.1 Scalar multiplication and some convenience functions

Definition 4.3.1. *For all $a \in \mathbb{K}[A]$ and $\boldsymbol{u} \in \boldsymbol{T}$, we define*

$$a\boldsymbol{u} = \begin{cases} \mathbf{O} & \text{if } a = 0 \text{ or } \boldsymbol{u} = \mathbf{O} \\ (a\boldsymbol{u}_v, \boldsymbol{u}_\mathrm{P}, a\boldsymbol{u}_C) & \text{otherwise} \end{cases}$$

where $a\boldsymbol{A}$ indicates the element-wise product. Since the elements of $\mathbb{K}$ are a subset of $\mathbb{K}[A]$ this also defines scalar multiplication of trees by elements of $\mathbb{K}$.

Definition 4.3.2. *We will define a few useful notation or relations on sets of nodes in $\boldsymbol{T}$. Take $\boldsymbol{U}$ and $\boldsymbol{V}$ be such sets and $\boldsymbol{a}$ be a node in $\boldsymbol{T}$; then*

$$L(\boldsymbol{U}) = \{\boldsymbol{e}_\mathrm{P} : \forall \boldsymbol{e} \in \boldsymbol{U}\}$$

$$\boldsymbol{U}|_{L(\boldsymbol{V})} = \{\boldsymbol{f} \in \boldsymbol{U} : \boldsymbol{f}_\mathrm{P} \in L(\boldsymbol{V})\}$$
$$\boldsymbol{U}|_{\overline{L}(\boldsymbol{V})} = \{\boldsymbol{f} \in \boldsymbol{U} : \boldsymbol{f}_\mathrm{P} \notin L(\boldsymbol{V})\}$$

and

$$a\boldsymbol{U} = \{a\boldsymbol{u} : \forall \boldsymbol{u} \in \boldsymbol{U}\}$$

Definition 4.3.3. *For convenience, we define analogues of several of the above relations for nodes to implicitly refer to the children of those nodes.*

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be arbitrary nodes in $\boldsymbol{T}$. Then we define the following

$$\begin{aligned} L(\boldsymbol{u}) &\equiv L(\boldsymbol{u}_C) \\ \boldsymbol{u}|_{L(\boldsymbol{v})} &\equiv \boldsymbol{u}_C|_{L(\boldsymbol{v}_C)} \\ \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} &\equiv \boldsymbol{u}_C|_{\overline{L}(\boldsymbol{v}_C)} \end{aligned}$$

4.3.2 Addition

Now we define the addition of two trees,

Definition 4.3.4. *For compatible nodes $\boldsymbol{u}$ and $\boldsymbol{v} \in \boldsymbol{T}$,*

$$\boldsymbol{u} + \boldsymbol{v} = \begin{cases} \boldsymbol{u} & \text{if } \boldsymbol{v} = \mathbf{0} \\ \boldsymbol{v} & \text{if } \boldsymbol{u} = \mathbf{0} \\ \Big(\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \big(\boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} \cup \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})} & \\ \quad \cup\{\boldsymbol{r} + \boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})} \text{ and } \boldsymbol{r}_{\mathsf{P}} = \boldsymbol{s}_{\mathsf{P}}\}\big) \setminus \{\mathbf{0}\}\Big) & \text{otherwise} \end{cases}$$

Here we extend the notion to the addition of the elements of two sets. This allows us to be a little more concise in dealing with the children.

Definition 4.3.5 (Notation)**.** *We define a mapping which takes two sets, each comprised of nodes in $\boldsymbol{T}$, to another set of nodes.The resulting set contains the nodes of the each of the operands which are incompatible with all the nodes in the other and the set obtained by applying addition to compatible pairs from the first and second set. This notation is used to express the restriction of addition to compatible nodes in the children of the root nodes. Let*

$$\boldsymbol{B} \boxplus \boldsymbol{C} = \big(\boldsymbol{B}|_{\overline{L}(\boldsymbol{C})} \cup \boldsymbol{C}|_{\overline{L}(\boldsymbol{B})} \cup \{\boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{B}|_{L(\boldsymbol{C})} \text{ and } \boldsymbol{s} \in \boldsymbol{C}|_{L(\boldsymbol{B})} \text{ and } \boldsymbol{r}_{\mathsf{P}} = \boldsymbol{s}_{\mathsf{P}}\}\big) \setminus \{\mathbf{0}\}.$$

Notice that $\boxplus$ must commute by the symmetry of its construction and the commutativity of tree addition. Moreover, it must also be associative, since both tree addition and set intersection are associative.

4.4 Properties of a vector space

In this section we prove that the defined elements and operations give us a vector space.

Proposition 4.4.1. *$\boldsymbol{T}$, with scalar multiplication and addition is a vector space.*

Proof. We will assume that $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{T}$ and $a, b \in \mathbb{K}$, and that addition $\boldsymbol{u} \sim \boldsymbol{v} \sim \boldsymbol{w}$ (they are all compatible).

Additive identity element

This is explicit in the definition of addition between elements of $\boldsymbol{T}$.

Inverse elements with respect to addition

The element $\mathbf{O}$ is its own inverse, since $\mathbf{O} + \mathbf{O} = \mathbf{O}$ by definition.

So, we consider the case of $\boldsymbol{u}$, where depth($\boldsymbol{u}$) = 1, then

$$\begin{aligned} \boldsymbol{u} + -\boldsymbol{u} &= (\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) + (-1\,\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= (\boldsymbol{u}_v + -\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= \mathbf{O} \end{aligned}$$

so for any simple node, $\boldsymbol{u}$, $-\boldsymbol{u}$ is its inverse.

Let $\boldsymbol{v}$ be a non-null node which is not simple, but has simple c, that is to say depth($\boldsymbol{v}$) = 2). Then

$$\begin{aligned} \boldsymbol{v} + -\boldsymbol{v} &= (\boldsymbol{v}_v - \boldsymbol{v}_v, \boldsymbol{v}_{\mathsf{P}}, \{\boldsymbol{e} + -\boldsymbol{e} : \forall\, \boldsymbol{e} \in \boldsymbol{v}_C\} \setminus \{\mathbf{O}\}) \\ &= (0, \boldsymbol{v}_{\mathsf{P}}, \varnothing) \\ &= \mathbf{O} \end{aligned}$$

since the simple leaf nodes are all added to their own additive inverse.

Having established this, we can generalise to trees with a depth greater than two. Assuming that the proposition holds for elements of $\boldsymbol{T}$ with depth n, we consider an element, $\boldsymbol{u}$, where depth($\boldsymbol{u}$) = $n+1$ added to the element $-\boldsymbol{u}$.

$$\boldsymbol{u} + -\boldsymbol{u} = (0, \boldsymbol{u}_{\mathsf{P}}, \{\boldsymbol{e} + -\boldsymbol{e} : \forall\, \boldsymbol{e} \in \boldsymbol{u}_C\})$$

Since the child nodes of the root node of $\boldsymbol{u}$ are, by assumption, added to their additive inverses, they then become

$$\begin{aligned} &= (\boldsymbol{v}_v + -\boldsymbol{v}_v, \boldsymbol{v}_{\mathsf{P}}, \varnothing) \\ &= \mathbf{O} \end{aligned}$$

for each $\boldsymbol{v} \in \boldsymbol{u}_C$ and, by induction, our inverse holds for all members of $\boldsymbol{T}$.

Multiplicative identity element

First observe that $1\,\mathbf{O}$ is by definition $\mathbf{O}$.

Now consider an arbitrary non-null tree in $\boldsymbol{T}$, $\boldsymbol{u}$; $\boldsymbol{u}$ is either simple, or it has child nodes. In the case of a simple $\boldsymbol{u}$, it is obvious that $1 \in \mathbb{K}$ acts as an identity.

$$\begin{aligned} \boldsymbol{u} &= (\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= (1\,\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= 1(\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= 1\,\boldsymbol{u} \end{aligned}$$

Thus, for all leaf nodes on $\boldsymbol{u}$, 1 is the identity for scalar multiplication. Now we take $\boldsymbol{u}$ to be some non-simple node,

$$\begin{aligned}
\boldsymbol{u} &= (\boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, \boldsymbol{u}_C) \\
&= (1\,\boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, 1\,\boldsymbol{u}_C) \\
&= (1\,\boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, \{(1\,\boldsymbol{e}_v, \boldsymbol{e}_\mathsf{P}, \boldsymbol{e}_C) : \forall\, \boldsymbol{e} \in \boldsymbol{u}_C\}),
\end{aligned}$$

and, if the children are all simple nodes,

$$\begin{aligned}
&= 1(\boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, \boldsymbol{u}_C) \\
&= 1\,\boldsymbol{u}.
\end{aligned}$$

so it is the case that 1 is the scalar multiplicative identity for nodes which have a depths of less than three.

Now suppose that 1 is the identity for scalar multiplication of all members of $\boldsymbol{T}$ with a depth of n or less for some natural number n, and we consider $\boldsymbol{v}$ which has a depth of $n+1$. Then

$$\begin{aligned}
\boldsymbol{v} &= (\boldsymbol{v}_v, \boldsymbol{v}_\mathsf{P}, \boldsymbol{v}_C) \\
&= (1\,\boldsymbol{v}_v, \boldsymbol{v}_\mathsf{P}, 1\,\boldsymbol{v}_C)
\end{aligned}$$

.

But each of the elements in the set $1\,\boldsymbol{v}_C$ either has a depth of n or less and so the the children of $1\,\boldsymbol{v}$ is merely $\boldsymbol{v}_C$, and so $1\,\boldsymbol{v} = \boldsymbol{v}$.

By induction, we can demonstrate that 1 is the identity for scalar multiplication of nodes of arbitrary (finite) depth.

Commutativity

Let us consider compatible nodes $\boldsymbol{u}$ and $\boldsymbol{v}$.

The commutativity of addition involving $\mathbf{O}$ is guaranteed by the definition of addition. so we first address the case where both addends are simple.

Take $\boldsymbol{u}$ and $\boldsymbol{v}$ to be simple nodes; then

$$\begin{aligned}
\boldsymbol{u} + \boldsymbol{v} &= (\boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, \varnothing) + (\boldsymbol{v}_v, \boldsymbol{v}_\mathsf{P}, \varnothing) \\
&= (\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_\mathsf{P}, \varnothing) \\
&= (\boldsymbol{v}_v + \boldsymbol{u}_v, \boldsymbol{u}_\mathsf{P}, \varnothing) \\
&= \boldsymbol{v} + \boldsymbol{u}.
\end{aligned}$$

Now suppose that there is some number n for which $\operatorname{depth}(\boldsymbol{u}) \le n$ and $\operatorname{depth}(\boldsymbol{v}) \le n \implies \boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$.

Then if we take $\boldsymbol{u}$ and $\boldsymbol{v}$ to be nodes with depths of $n+1$ or less,

$$\begin{aligned}
\boldsymbol{u}+\boldsymbol{v} &= (\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_C) + (\boldsymbol{v}_v, \boldsymbol{v}_{\mathsf{P}}, \boldsymbol{v}_C) \\
&= \Big(\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \big(\{ \boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})} \text{ and } \boldsymbol{r}_{\mathsf{P}} = \boldsymbol{s}_{\mathsf{P}} \} \\
&\qquad \cup \{ \boldsymbol{r} : \boldsymbol{r} \in \boldsymbol{u}_C \text{ and } \boldsymbol{r}_{\mathsf{P}} \notin L(\boldsymbol{v}) \} \cup \{ \boldsymbol{s} : \boldsymbol{s}_C \in \boldsymbol{v} \text{ and } \boldsymbol{s}_{\mathsf{P}} \notin L(\boldsymbol{u}) \} \big) \setminus \{ \mathbf{O} \} \Big) \\
&= \Big(\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \big(\{ \boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}_C \text{ and } \boldsymbol{r}_{\mathsf{P}} \in L(\boldsymbol{v}) \text{ and } \boldsymbol{s} \in \boldsymbol{v}_C \text{ and } \boldsymbol{s}_{\mathsf{P}} \in L(\boldsymbol{u}) \} \\
&\qquad \cup \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} \cup \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})} \big) \setminus \{ \mathbf{O} \}
\end{aligned}$$

So,

$$\begin{aligned}
\boldsymbol{u}+\boldsymbol{v} = \Big(\boldsymbol{v}_v + \boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \big(\{ \boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}_C \text{ and } \boldsymbol{r}_{\mathsf{P}} \in L(\boldsymbol{v}) \text{ and } \boldsymbol{s} \in \boldsymbol{v}_C \text{ and } \boldsymbol{s}_{\mathsf{P}} \in L(\boldsymbol{u}) \} \\
\cup \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} \cup \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})} \big) \setminus \{ \mathbf{O} \}
\end{aligned}$$

since addition in $\boldsymbol{T}$ is commutative. If we can demonstrate that the expression for the children is independent of order, then it must be the case that sum of the addends, $\boldsymbol{u}$ and $\boldsymbol{v}$, must also be order independent.

The set $\{ \boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}_C \text{ and } \boldsymbol{r}_{\mathsf{P}} \in L(\boldsymbol{v}) \text{ and } \boldsymbol{s} \in \boldsymbol{v}_C \text{ and } \boldsymbol{s}_{\mathsf{P}} \in L(\boldsymbol{u}) \}$ must be order independent since each of the candidate $\boldsymbol{r}$ and $\boldsymbol{s}$ addends must have a depth of n or less. Since set union is commutative, the order of $\boldsymbol{u}|_{\overline{L}(\boldsymbol{v})}$ and $\boldsymbol{v}|_{\overline{L}(\boldsymbol{u})}$ doesn't affect the result, thus, addition must be commutative for all $\boldsymbol{u}$ where $\text{depth}(\boldsymbol{u}) \leq n+1$. By induction, this must be true for all $n \geq 0$.

Associativity

Let us consider compatible nodes $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ in $\boldsymbol{T}$.

First consider the situation where the depths of $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are all less than or equal to one. If they all have a depth of zero, the sum is almost trivially the null tree. Similarly, if only one is the null tree, it rapidly degenerates to simple addition. So we take $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ to be simple. Then

$$\begin{aligned}
(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w} &= ((\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) + (\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing)) + (\boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\
&= (\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) + (\boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\
&= ((\boldsymbol{u}_v + \boldsymbol{v}_v) + \boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\
&= (\boldsymbol{u}_v + (\boldsymbol{v}_v) + \boldsymbol{w}_v), \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\
&= \boldsymbol{u} + (\boldsymbol{v}+\boldsymbol{w}).
\end{aligned}$$

Let us consider the case where these may be non-simple trees. Suppose there is an integer n such that associativity holds for any three trees $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$, whose depth is less than or equal to n, that is if $\text{depth}(\boldsymbol{u}) \leq n$, $\text{depth}(\boldsymbol{v}) \leq n$ and $\text{depth}(\boldsymbol{w}) \leq n$, then it must be the case that

$$(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w} = \boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w})$$

.

Now suppose one or more of these trees has a depth of $n+1$.

$$\begin{aligned}(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w} &= ((\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_C)+(\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{v}_C))+(\boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{w}_C)\\ &= (\boldsymbol{u}_v+\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_C \boxplus \boldsymbol{v}_C)+(\boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{w}_C)\end{aligned}$$

Recall that

$$\boldsymbol{u}_C \boxplus \boldsymbol{v}_C = \left(\boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} \cup \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})} \cup \{\boldsymbol{p}+\boldsymbol{q} : \boldsymbol{p} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{q} \in \boldsymbol{v}|_{L(\boldsymbol{u})} \text{ and } \boldsymbol{p}_{\mathsf{P}} = \boldsymbol{q}_{\mathsf{P}}\}\right) \setminus \{\mathbf{O}\}$$

so, letting

$$\boldsymbol{B} = \boldsymbol{u}_C \boxplus \boldsymbol{v}_C$$

we get

$$\begin{aligned}(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w} &= \Big((\boldsymbol{u}_v+\boldsymbol{v}_v)+\boldsymbol{w}_v, \boldsymbol{u}_{\mathsf{P}}, \big(\boldsymbol{B}|_{\overline{L}(\boldsymbol{w})} \cup \boldsymbol{w}|_{\overline{L}(\boldsymbol{B})} \cup \boldsymbol{B} \boxplus \boldsymbol{w}_C\big) \setminus \{\mathbf{O}\}\Big)\\ &= \Big(\boldsymbol{u}_v+(\boldsymbol{v}_v+\boldsymbol{w}_v), \boldsymbol{u}_{\mathsf{P}}, \big(\boldsymbol{B}|_{\overline{L}(\boldsymbol{w})} \cup \boldsymbol{w}|_{\overline{L}(\boldsymbol{B})} \cup \boldsymbol{B} \boxplus \boldsymbol{w}_C\big) \setminus \{\mathbf{O}\}\Big)\end{aligned}$$

since addition in $\boldsymbol{T}$ is associative

Notice that the elements of all the sets which comprise the children, those in $\boldsymbol{B}$ and in $\boldsymbol{w}_C$, must have a depth of n or less; any addition which occurs amongst the elements of these sets must be associative by our inductive assumption. Hence

$$(\boldsymbol{u}+\boldsymbol{v})+\boldsymbol{w} = \boldsymbol{u}+(\boldsymbol{v}+\boldsymbol{w}).$$

Compatibility of scalar multiplication and multiplication in $\mathbb{K}$

Observe first that $a\mathbf{O} = \mathbf{O}, \forall a \in \mathbb{K}$. We also dispose with the case of simple nodes:

$$\begin{aligned}a(b\boldsymbol{u}) &= (a(b\boldsymbol{u}_v), \boldsymbol{u}_{\mathsf{P}}, \varnothing)\\ &= (ab\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing)\\ &= ((ab)\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing)\\ &= (ab)\boldsymbol{u};\end{aligned}$$

So assuming that multiplication is compatible with nodes with depths of n or less, we consider $\boldsymbol{u}$, where $\text{depth}(\boldsymbol{u}) = n+1$,

$$a(b\boldsymbol{u}) = a(b\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, b\boldsymbol{u}_C)$$

since depth$(\boldsymbol{e}) \leq n \forall\, \boldsymbol{e} \in \boldsymbol{u}_C$, multiplication of these elements is compatible, and

$$= (ab\,\boldsymbol{u}_v,\, \boldsymbol{u}_{\mathsf{P}}, a(b\,\boldsymbol{u}_C))$$

becomes

$$\begin{aligned} &= ((ab)\,\boldsymbol{u}_v,\, \boldsymbol{u}_{\mathsf{P}}, (ab)\,\boldsymbol{u}_C) \\ &= (ab)\,\boldsymbol{u} \end{aligned}$$

Thus the scalar and field multiplication operators are compatible.

Distribution of scalar multiplication with respect to vector addition

Let us consider compatible trees, $\boldsymbol{u}$ and $\boldsymbol{v}$.

First, note that

$$\forall\, \boldsymbol{u} \in \boldsymbol{T}, a(\mathbf{O} + \boldsymbol{u}) = a\,\boldsymbol{u} = a\,\mathbf{O} + a\,\boldsymbol{u},$$

and that

$$\forall\, \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{T}, 0(\boldsymbol{u} + \boldsymbol{v}) = \mathbf{O} = 0\,\boldsymbol{u} + 0\,\boldsymbol{v}.$$

The property holds for simple nodes,

$$\begin{aligned} a(\boldsymbol{u} + \boldsymbol{v}) &= a((\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) + (\boldsymbol{v}_v, \boldsymbol{v}_{\mathsf{P}}, \varnothing)) \\ &= a\,(\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= (a(\boldsymbol{u}_v + \boldsymbol{v}_v), \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= (a\,\boldsymbol{u}_v + a\,\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= (a\,\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) + (a\,\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \varnothing) \\ &= a\,\boldsymbol{u} + a\,\boldsymbol{v} \end{aligned}$$

.

So, suppose that the equation $a(\boldsymbol{p} + \boldsymbol{q}) = a\,\boldsymbol{p} + a\,\boldsymbol{q}$ holds for all compatible nodes $\boldsymbol{p}$ and $\boldsymbol{q}$ such that $depth(\boldsymbol{p}) \leq k$, and depth$(\boldsymbol{q}) \leq j$.

Take $n = \min(j,k)$, $a \in \mathbb{K}$, and nodes $\boldsymbol{u}$ and $\boldsymbol{v}$ such that depth$(\boldsymbol{u}) = n + 1$, and depth$(\boldsymbol{v}) = n + 1$. Note that n must be greater than zero since the property holds for simple nodes. Then

$$\begin{aligned} a(\boldsymbol{u} + \boldsymbol{v}) &= a((\boldsymbol{u}_v, \boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_C) + (\boldsymbol{v}_v, \boldsymbol{v}_{\mathsf{P}}, \boldsymbol{v}_C) \\ &= a\Big(\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \big\{\boldsymbol{u}|_{\overline{L}(\boldsymbol{v})} \cup \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})} \\ &\qquad \cup \{\boldsymbol{r} + \boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})}\}\big\} \setminus \{\mathbf{O}\}\Big) \\ &= \Big(a(\boldsymbol{u}_v + \boldsymbol{v}_v), \boldsymbol{u}_{\mathsf{P}}, \big\{\{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})}\} \cup \{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})}\} \\ &\qquad \cup \{a(\boldsymbol{r} + \boldsymbol{s}) : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})}\}\big\} \setminus \{\mathbf{O}\}\Big) \\ &= \Big(a\,\boldsymbol{u}_v + a\,\boldsymbol{v}_v, \boldsymbol{u}_{\mathsf{P}}, \big\{\{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})}\} \cup \{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})}\} \\ &\qquad \cup \{a(\boldsymbol{r} + \boldsymbol{s}) : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})}\}\big\} \setminus \{\mathbf{O}\}\Big) \end{aligned}$$

.

Notice that the component sets of the set of children to $\boldsymbol{u}+\boldsymbol{v}$, namely $\{a\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}|_{\overline{L}(\boldsymbol{v})}\}$, $\{a\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{v}|_{\overline{L}(\boldsymbol{u})}\}$ and $\{a(\boldsymbol{r}+\boldsymbol{s}) : \boldsymbol{r} \in \boldsymbol{u}|_{L(\boldsymbol{v})}$ and $\boldsymbol{s} \in \boldsymbol{v}|_{L(\boldsymbol{u})}\}$ can only contain nodes with a depth of n or less; Thus, we can proceed inductively, increasing the least upper bound, (min(j,k), for the set of trees that cooperate with distribution of scalar multiplication over vector addition, to any value we wish.

Distribution of scalar multiplication with respect to addition in $\mathbb{K}$

The property is clearly true when $\boldsymbol{u} = \mathbf{O}$, since $(a+b)\,\mathbf{O} = \mathbf{O} = a\,\mathbf{O} + b\,\mathbf{O}$.

We first consider simple nodes:

$$\begin{aligned}(a+b)\,\boldsymbol{u} &= (a+b)((a+b)\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\varnothing)\\ &= ((a+b)\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\varnothing)\\ &= (a\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\varnothing) + (b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\varnothing)\\ &= a\,\boldsymbol{u} + b\,\boldsymbol{u}.\end{aligned}$$

Nodes with a depth of two are slightly more complicated,

$$\begin{aligned}(a+b)\,\boldsymbol{u} &= (a+b)((a+b)\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},(a+b)\,\boldsymbol{u}_{\mathsf{C}})\\ &= (a\,\boldsymbol{u}_v + b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\{(a+b)\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}_{\mathsf{C}}\})\end{aligned}$$

but $\boldsymbol{u}_{\mathsf{C}}$ is composed of simple nodes, so,

$$\begin{aligned}&= (a\,\boldsymbol{u}_v + b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\{a\,\boldsymbol{e} + b\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}_{\mathsf{C}}\})\\ &= (a\,\boldsymbol{u}_v + b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},a\,\boldsymbol{u}_{\mathsf{C}}) + (b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},b\,\boldsymbol{u}_{\mathsf{C}})\\ &= a\,\boldsymbol{u} + b\,\boldsymbol{u}.\end{aligned}$$

Now suppose the property holds for nodes with a depth of n. Then we consider node $\boldsymbol{u}$ with a depth of $n+1$:

$$\begin{aligned}(a+b)\,\boldsymbol{u} &= (a+b)(\ \boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\ \boldsymbol{u}_{\mathsf{C}})\\ &= ((a+b)\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},(a+b)\,\boldsymbol{u}_{\mathsf{C}}),\\ &= (a\,\boldsymbol{u}_v + b\,\boldsymbol{u}_v,\ \boldsymbol{u}_{\mathsf{P}},\{a\,\boldsymbol{e} + b\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}_{\mathsf{C}}\})\end{aligned}$$

since depth$(\ \boldsymbol{e}) = n$

$$= a\,\boldsymbol{u} + b\,\boldsymbol{u}.$$

By induction, the property must hold for all $n >= 0$ □

4.5 Seminorms, norms and metrics

Now that we have a vector space, we can construct model configurations as linear combinations of basis configurations. In the context of models which change their configuration, we need a way for the model itself to combine basis configuration trees by choosing from a set of configurations that are known to exhibit suitable properties. To this end, we need a mechanism for judging how close or far a given configuration is from where it needs to be – we need a way to decompose an extant, running configuration into its basis elements, and then map these to some provably more appropriate configuration. To do this, our structure needs to be a metric space.

We will now construct a seminorm on the vector space $\boldsymbol{T}$. This will induce a norm on a quotient space of $\boldsymbol{T}$ which we can use as a tool for assessing the similarity of trees and, ultimately, provide both a means of clustering trees and selecting trees with particular properties.

4.5.1 $\boldsymbol{T}$ and its seminorm

Definition 4.5.1. *We define the absolute value of a node to be*

$$(\!|\boldsymbol{u}|\!) = \begin{cases} 0 & \text{if } \boldsymbol{u} = \mathbf{O} \\ |\boldsymbol{u}_v| & \text{if } \boldsymbol{u}_C = \varnothing \\ |\boldsymbol{u}_v| + \sum_{e \in \boldsymbol{u}_C} (\!|\boldsymbol{e}|\!) & \text{otherwise.} \end{cases}$$

The absolute magnitude is only based only on the values of the nodes of trees. Note that each node in a tree can only contribute a non-negative quantity to the absolute value of the tree, it is obvious that $(\!|\boldsymbol{u}|\!) \geq 0$ for all $\boldsymbol{u} \in \boldsymbol{T}$ and that equality only occurs if the value of each node in the tree $\boldsymbol{u}$ is zero.

Proposition 4.5.1. *For $a \in \mathbb{K}$ and $\boldsymbol{u} \in \boldsymbol{T}$, $|a|(\!|\boldsymbol{u}|\!) = (\!|a\,v|\!)$.*

Proof. The magnitude of the empty tree is trivially zero, so $|a|(\!|\mathbf{O}|\!) = (\!|a\,\mathbf{O}|\!) = 0$.

Consider simple nodes in $\boldsymbol{T}$:

$$\begin{aligned} |a|(\!|\boldsymbol{u}|\!) &= |a|(|\boldsymbol{u}_v| + 0) \\ &= |a||\boldsymbol{u}_v| \\ &= (\!|a\,\boldsymbol{u}|\!). \end{aligned}$$

Now suppose that there is $n \geq 1$ such that the proposition is true for all trees with a depth of n or less. Then, taking $\boldsymbol{u} \in \boldsymbol{T}$ where $\text{depth}(\boldsymbol{u}) = n + 1$, we have

$$\begin{aligned} |a|(\!|\boldsymbol{u}|\!) &= |a|\big((|\boldsymbol{u}_v| + \sum_{e \in \boldsymbol{u}_C} (\!|\boldsymbol{e}|\!))\big) \\ &= |a||\boldsymbol{u}_v| + \sum_{e \in \boldsymbol{u}_C} |a|(\!|\boldsymbol{e}|\!) \end{aligned}$$

but all the elements in $\boldsymbol{u}_C$ have a depth of k or less

$$\begin{aligned}
&= ||a|\,\boldsymbol{u}_v| + \sum_{e\in \boldsymbol{u}_C} (\!| |a|\,\boldsymbol{e} |\!) \\
&= |a\,\boldsymbol{u}_v| + \sum_{e\in \boldsymbol{u}_C} (\!| a\,\boldsymbol{e} |\!) \\
&= (\!| a\,\boldsymbol{u} |\!).
\end{aligned}$$

By induction, the proposition must be true for all $n \geq 0$. □

Proposition 4.5.2. *For* $\boldsymbol{u}$ *and* $\boldsymbol{v} \in \boldsymbol{T}$, $(\!| \boldsymbol{u} + \boldsymbol{v} |\!) \leq (\!| \boldsymbol{u} |\!) + (\!| \boldsymbol{v} |\!)$.

Proof. We start by considering trees of depths zero and one. The case for null trees is trivial: $(\!| \mathbf{O} + \mathbf{O} |\!) = |0+0| = 0$, and if only one of the trees has a depth of one, we get either $(\!| \boldsymbol{u} + \mathbf{O} |\!) = (\!| \boldsymbol{u} |\!)$ or $(\!| \mathbf{O} + \boldsymbol{u} |\!) = (\!| \boldsymbol{u} |\!)$.

For $\boldsymbol{u}$ and $\boldsymbol{v}$ with depths of one,

$$(\!| \boldsymbol{u} + \boldsymbol{v} |\!) = (\!| (\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathrm{P}}, \varnothing) |\!) = |\boldsymbol{u}_v + \boldsymbol{v}_v|$$

. Since $\boldsymbol{u}_v$ and $\boldsymbol{v}_v$ are scalars in $\mathbb{K}$, we must have $|\boldsymbol{u}_v + \boldsymbol{v}_v| \leq |\boldsymbol{u}_v| + |\boldsymbol{v}_v|$, so

$$|\boldsymbol{u}_v + \boldsymbol{v}_v| \leq |\boldsymbol{u}_v| + |\boldsymbol{v}_v| = (\!| \boldsymbol{u} |\!) + (\!| \boldsymbol{v} |\!).$$

We will now proceed by induction; let n be a positive integer for which the triangle inequality holds for all trees with a depth of k or less. Let's consider compatible trees, $\boldsymbol{u}$ and $\boldsymbol{v}$ whose depths are less than or equal to $n+1$. Then

$$\begin{aligned}
(\!| \boldsymbol{u} + \boldsymbol{v} |\!) &= (\!| (\boldsymbol{u}_v + \boldsymbol{v}_v, \boldsymbol{u}_{\mathrm{P}}, \boldsymbol{u}_C \boxplus \boldsymbol{v}_C) |\!) \\
&= \Big[|\boldsymbol{u}_v + \boldsymbol{v}_v| + \sum_{e\in \boldsymbol{u}_C \boxplus \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!) \Big].
\end{aligned}$$

Observe that $|\boldsymbol{u}_v + \boldsymbol{v}_v| \leq |\boldsymbol{u}_v| + |\boldsymbol{v}_v|$, and that each of the addends in

$$\sum_{e\in \boldsymbol{u}_C \boxplus \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!)$$

has a depth of n or less, so

$$\sum_{e\in \boldsymbol{u}_C \boxplus \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!) \leq \sum_{e\in \boldsymbol{u}_C} (\!| \boldsymbol{e} |\!) + \sum_{e\in \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!).$$

This implies that

$$(\!| \boldsymbol{u} + \boldsymbol{v} |\!) \leq \Big[|\boldsymbol{u}_v| + |\boldsymbol{v}_v| + \sum_{e\in \boldsymbol{u}_C} (\!| \boldsymbol{e} |\!) + \sum_{e\in \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!) \Big];$$

rearranging we get

$$(\!| \boldsymbol{u} + \boldsymbol{v} |\!) \leq \Big[|\boldsymbol{u}_v| + \sum_{e\in \boldsymbol{u}_C} (\!| \boldsymbol{e} |\!) \Big] + \Big[|\boldsymbol{v}_v| + \sum_{e\in \boldsymbol{v}_C} (\!| \boldsymbol{e} |\!) \Big]$$

and hence

$$\llparenthesis \boldsymbol{u} + \boldsymbol{v} \rrparenthesis \le \llparenthesis \boldsymbol{u} \rrparenthesis + \llparenthesis \boldsymbol{v} \rrparenthesis.$$

□

Corollary 4.5.1. *The absolute value forms a seminorm on $\boldsymbol{T}$.*

Proof. Propositions, 4.5.1 and 4.5.2, are sufficient for the absolute value to be a seminorm on $\boldsymbol{T}$. □

At this point we should consider the elements $\boldsymbol{o} \in \boldsymbol{T}$ which are analogues of zero. We define the set $\mathfrak{O} = \{\boldsymbol{o} \in \boldsymbol{T} : \llparenthesis \boldsymbol{o} \rrparenthesis = 0\}$, and observe that for any $\boldsymbol{e} \in \boldsymbol{T}$, and $\boldsymbol{o} \in \mathfrak{O}$ the equation $\llparenthesis \boldsymbol{e} + \boldsymbol{o} \rrparenthesis = \llparenthesis \boldsymbol{e} \rrparenthesis$ must hold.

Since $\boldsymbol{T}$ is a seminormed vector space, it is also a pseudometric space and we can induce a fully fledged metric space over the quotient space $\check{\boldsymbol{T}} = \boldsymbol{T}/\mathfrak{O}$.

For simplicity, we identify the coset of $\mathfrak{O}$ with respect to O with $\check{\mathsf{O}}$, and we take the induced metric on the normed vector space $\check{\boldsymbol{T}}$, to be

$$\mathrm{d}(\check{\boldsymbol{u}}, \check{\boldsymbol{v}}) = \llparenthesis \check{\boldsymbol{u}} - \check{\boldsymbol{v}} \rrparenthesis \text{ for all } \check{\boldsymbol{u}}, \check{\boldsymbol{v}} \in \check{\boldsymbol{T}}$$

. We will continue to use $\llparenthesis \check{\boldsymbol{u}} \rrparenthesis$ to denote the the induced absolute value of $\check{\boldsymbol{u}} \in \check{\boldsymbol{T}}$.

4.6 Element multiplication in $\check{T}$ and establishing the properites of a ring

In this section, we will define a multiplicative operator for trees, and demonstrate that there is a multiplicative identity. My motivation for establising these properties is twofold; most importantly, it broadens the set of mathematical tools we have at our disposal to analyse sets of trees (clustering, classification, interpolation, extrapolation...), but also because we never know just what else we might discover along the way!

Definition 4.6.1. *We define the multiplication of two trees (or nodes) to be*

$$\check{\boldsymbol{u}} \cdot \check{\boldsymbol{v}} = \begin{cases} \check{\mathsf{O}} & \textit{if } \check{\boldsymbol{u}} = \check{\mathsf{O}} \textit{ or } \check{\boldsymbol{v}} = \check{\mathsf{O}} \\ (\check{\boldsymbol{u}}_\nu \check{\boldsymbol{v}}_\nu, \check{\boldsymbol{u}}_{\mathrm{P}} \check{\boldsymbol{v}}_{\mathrm{P}}, \check{\boldsymbol{u}}_{\mathrm{P}} \check{\boldsymbol{v}}_{C} \boxplus \check{\boldsymbol{v}}_{\mathrm{P}} \check{\boldsymbol{u}}_{C}) & \textit{otherwise} \end{cases}$$

.

where the notation $\check{\boldsymbol{u}}_{\mathrm{P}} \check{\boldsymbol{v}}_{C}$or $\check{\boldsymbol{v}}_{C} \check{\boldsymbol{u}}_{\mathrm{P}}$ corresponds to the set obtained by multiplying each label in the root node of the set $\check{\boldsymbol{v}}_{C}$ by the label $\check{\boldsymbol{u}}_{\mathrm{P}}$. Clearly, if the set in the operation (on either side) is null, then the result is null.

Proposition 4.6.1. *$\iota = (1, 1, \varnothing)$ commutes with all other nodes, is the multiplicative identity, and it is unique,* *Proof.* Let $\check{\boldsymbol{v}}$ be some arbitrary tree, then

$$\begin{aligned}
(\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathrm{P}}, \check{\boldsymbol{v}}_C) \cdot \iota &= (\check{\boldsymbol{v}}_\nu 1, \check{\boldsymbol{v}}_{\mathrm{P}} 1, \check{\boldsymbol{v}}_C 1 \boxplus \varnothing \check{\boldsymbol{v}}_{\mathrm{P}}) \\
&= (1\check{\boldsymbol{v}}_\nu, 1\check{\boldsymbol{v}}_{\mathrm{P}}, 1\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{v}}_\nu \varnothing) \\
&= (1\check{\boldsymbol{v}}_\nu, 1\check{\boldsymbol{v}}_{\mathrm{P}}, \check{\boldsymbol{v}}_\nu \varnothing \boxplus 1\check{\boldsymbol{v}}_C) \\
&= (1\check{\boldsymbol{v}}_\nu, 1\check{\boldsymbol{v}}_{\mathrm{P}}, 1\check{\boldsymbol{v}}_C) \\
&= \iota \cdot (\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathrm{P}}, \check{\boldsymbol{v}}_C) \\
&= (\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathrm{P}}, \check{\boldsymbol{v}}_C)
\end{aligned}$$

□

We can see that, since both the weight and label are members of a field, the only possible value for both the weight and the label of the identity is one. This leaves us to consider our options for the set of children. Suppose we have an alternative identity, I, with a non-empty set of children; then the set of children in the product must be $\check{\boldsymbol{v}}_C 1 \boxplus \check{\boldsymbol{I}}_C \check{\boldsymbol{v}}_{\mathrm{P}}$. For this to be the identity, $\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{I}}_C \check{\boldsymbol{v}}_{\mathrm{P}}$ must equal $\check{\boldsymbol{v}}_C$. This means, however that $\check{\boldsymbol{I}}_C \check{\boldsymbol{v}}_{\mathrm{P}}$ contributes only trees which are members of $\mathfrak{O}$, but this implies that $\check{\boldsymbol{I}}_\nu = 0$ which contradicts our observation that it must be 1.

Now we must prove that the necessary multiplicative properties so that we can be confident that arithmetic involving trees works in the "normal" way.[1]

Proposition 4.6.2. *Multiplication of trees in $\check{\boldsymbol{T}}$ is commutative.*

Proof. Suppose there is a number n such that multiplication is commutative for all trees $\check{\boldsymbol{u}}$, $\check{\boldsymbol{v}}$ such that $\mathrm{depth}(\check{\boldsymbol{u}}), \mathrm{depth}(\check{\boldsymbol{v}}) \leq n$.

Multiplication involving nodes with a depth of zero clearly commutes, so n may reasonably take the value 0.

In the case where both nodes are simple, it is evident that they must commute, since scalar multiplication commutes, and the multiplication of ring of multinomials is commutative.

Suppose that one or both of the nodes has a depth of $n+1$. The children of the nodes all have depths of n or less, so the elements of the children of the product must be independent of the order of the operators in the multiplication, and both scalar multiplication and multiplication in the ring of multinomials commute. Hence the multiplication of nodes with a depth of $n+1$ must commute. By induction, we can say that trees of arbitrary depth commute with this definition of multiplication in $\check{\boldsymbol{T}}$. □

Proposition 4.6.3. *Multiplication of trees in $\check{\boldsymbol{T}}$ is associative.*

[1]Ideally, we would have inverses for trees (and hence a multiplicative identity), but, like matrices, this may only be possible (if it is at all) for a comparatively small part of $\check{\boldsymbol{T}}$.

Proof. Let $\check{\boldsymbol{u}}$, $\check{\boldsymbol{v}}$, and $\check{\boldsymbol{w}}$ be nodes in $\check{\boldsymbol{T}}$. Then,

$$\begin{aligned}\check{\boldsymbol{u}}\cdot(\check{\boldsymbol{v}}\cdot\check{\boldsymbol{w}}) &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \check{\boldsymbol{u}}_C)\cdot(\check{\boldsymbol{v}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}(\check{\boldsymbol{w}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C) \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}\,\check{\boldsymbol{u}}_C)\end{aligned}$$

but tree addition is both commutative and associative, and multiplication in the ring of multinomials also commutes, so

$$\begin{aligned}&= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}\,\check{\boldsymbol{u}}_C)\\ &= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_{\mathsf{P}}(\check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{u}}_C) \boxplus \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{v}}_{\mathsf{P}}\,\check{\boldsymbol{u}}_C)\cdot(\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}\cdot\check{\boldsymbol{v}})\cdot\check{\boldsymbol{w}}\end{aligned}$$

□

Proposition 4.6.4. *Tree-multiplication distributes over tree-addition in* $\check{\boldsymbol{T}}$.

Proof. We want to show that for $\check{\boldsymbol{u}}$, $\check{\boldsymbol{v}}$, and $\check{\boldsymbol{w}} \in \check{\boldsymbol{T}}$, where nodes $\check{\boldsymbol{v}}$ and $\check{\boldsymbol{w}}$ are compatible, $\check{\boldsymbol{u}}\cdot(\check{\boldsymbol{v}}+\check{\boldsymbol{w}}) = \check{\boldsymbol{u}}\cdot\check{\boldsymbol{v}}+\check{\boldsymbol{u}}\cdot\check{\boldsymbol{w}}$ is true.

Let us first consider multiplication of a sum by a node with a depth of one, $\check{\boldsymbol{u}}$, over the the sum $\check{\boldsymbol{v}}+\check{\boldsymbol{w}}$,

$$\begin{aligned}\check{\boldsymbol{u}}\cdot(\check{\boldsymbol{v}}+\check{\boldsymbol{w}}) &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \varnothing)\cdot(\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_C) + (\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \varnothing)\cdot(\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{w}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}_\nu(\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{w}}_\nu), \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}(\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{w}}_C) \boxplus (\check{\boldsymbol{v}}_{\mathsf{P}}\varnothing))\\ &= (\check{\boldsymbol{u}}_\nu(\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{w}}_\nu), \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}(\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{w}}_C))\\ &= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C)\\ &= \check{\boldsymbol{u}}\cdot\check{\boldsymbol{v}}+\check{\boldsymbol{u}}\cdot\check{\boldsymbol{w}}\end{aligned}$$

Note that this is independent of the depths of nodes $\check{\boldsymbol{v}}$ and $\check{\boldsymbol{w}}$.

Suppose then that there is an integer n such that multiplication of nodes with a depth of n or less distributes over addition, and we consider the case where our factor, $\check{\boldsymbol{u}}$, has a depth of $n+1$ or less. Then

$$\begin{aligned}\check{\boldsymbol{u}}\cdot(\check{\boldsymbol{v}}+\check{\boldsymbol{w}}) &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \check{\boldsymbol{u}}_C)\cdot\Big((\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_C) + (\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_C)\Big)\\ &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \check{\boldsymbol{u}}_C)\cdot(\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{w}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{w}}_C)\\ &= (\check{\boldsymbol{u}}_\nu(\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{w}}_\nu), \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_{\mathsf{P}}\,\boldsymbol{u}_C \boxplus \check{\boldsymbol{u}}_{\mathsf{P}}(\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{w}}_C))\end{aligned}$$

but since $\check{\boldsymbol{u}}_{\mathsf{P}} \in \mathbb{K}[A]$ and polynomial multiplication distributes over addtion

$$\begin{aligned}&= (\check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{v}}_\nu + \check{\boldsymbol{u}}_\nu\,\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_{\mathsf{P}}\,\boldsymbol{u}_C \boxplus (\check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{v}}_C \boxplus \check{\boldsymbol{u}}_{\mathsf{P}}\,\check{\boldsymbol{w}}_C))\\ &= (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \check{\boldsymbol{u}}_C)\cdot(\check{\boldsymbol{v}}_\nu, \check{\boldsymbol{v}}_{\mathsf{P}}, \check{\boldsymbol{v}}_C) + (\check{\boldsymbol{u}}_\nu, \check{\boldsymbol{u}}_{\mathsf{P}}, \check{\boldsymbol{u}}_C)\cdot(\check{\boldsymbol{w}}_\nu, \check{\boldsymbol{w}}_{\mathsf{P}}, \check{\boldsymbol{w}}_C)\\ &= \check{\boldsymbol{u}}\cdot\check{\boldsymbol{v}}+\check{\boldsymbol{u}}\cdot\check{\boldsymbol{w}}.\end{aligned}$$

□

4.7 Discussion

This metric space arose from attempts to capture the nuanced associations in survey questions like *"Thinking about the weather forecast, how would you rate the chances of your favourite sporting team in the coming match?"* and to be able to incorporate the sorts of conflicting data that respondents may provide into simulation models. Initially, the trees were no more than data structures with a rough and ready distance function, but as the work became more coherent, the underlying mathematical structure began to emerge, and the realisation that the trees might be useful for representing more than survey responses came about. The basic heuristic comparisons used in exploring the survey data were replaced with a better behaved metric based on the tree norm

Applying trees in this manner, in a program which might conceivably take weeks to run or be the basis of management decisions, would require more robust foundations than a heuristic function.

The loosely defined structure was defined and converted into a vector space so that I could then construct model-spaces from a set of basis elements corresponding to submodels. Extending this structure to the assessment of configurations required a metric space.

In the example model developed later, the states of the model as a whole, subdomains of the model and the components within the model are represented by representative trees. There is also a set of trees which are identified by known-good configurations, and the mechanism which handles switching within the model uses the metric in its assessment. Like the model in Chapter 2, the approach is quite shallow, but the hope is that, having established the ring-theoretic properties, more advanced clustering and discrimination techniques can be brought to bear.

In Chapter 3 the model developed is concerned with demonstrating the adaptive selection of models using abstracted representations of the model's components and of the model's configuration. These representations are in the form of trees in $\check{\mathcal{T}}$ with particular forms.

CHAPTER 5

An explicit implementation

Chapter 2 argued that switching models are worth considering because they may provide benefits in fidelity and efficiency. The results in the chapter demonstrate the potential advantages in these regards, and while the example used is quite simple compared to other models of marine or aquatic contaminant interaction, such as in Gray et al. [2006b, 2014a], the results are credible. The role of the example model discussed in this chapter is not to provide a complete, efficient implementation of an adaptive hybrid model. Rather, it is the starting point for further exploration and development. It follows the pattern described by Chapter 3, but includes minor improvements and enough extensions to provide a useful platforn for further work.

The focus in the development of this chapter is to illuminate the most important parts of the framework. This must obviously include brief discussions of the basic design decisions, the overall uses and relationships of the classes in the framework, a description of how control passes from kernel to agents and back, how representational change and maintenance occurs, and how one might go about adding new classes.

5.1 Design and implementation

5.1.1 Design

The design of the example framework is quite conventional for an agent-based model with arbitrary time steps, and the basic structure was influenced by previous work on ecosystem models. The mechanisms for changes in representation and the maintenance of state for superceded representations evolved from the basic strategy developed in Chapter 2. Since the only interactions in Chapter 2 were between the simulated organisms and the environmental plume, there was no need to include mechanisms for agents to interact in a more general sense. Much of the code in the example model is devoted to providing the basic mechanisms with which models of various entities, environmental domains may interact, and to ensure that the operation of the systems is temporally coherent, consistent and readily extended. Additionally, experience in computer operating systems also played a role in the design of the framework, and the queue of agents which is central to the process of shepherding them in an orderly way through time is directly analogous to the simpler UNIX kernels of decades past.

The choice of implementation language was more difficult. An obvious candidate would have been `C++`, since I had extensive experience in modelling in the language, and a significant corpus of code from which I might draw, but the considerations which prompted me to use Scheme in the paper of Chapter 2 still pertained—Implementing the clean closures for general state maintenance would be awkward. While Scheme is not a particularly commonly used language, it is a language under active development and refinement, there are a number of very good interpreters and compilers for it, and many of the implementations are amenable to integration with other languages, either through the linking binary objects or by interacting through a virtual machine (such as a JVM).

The class structure in the example is based on the Scheme implementation of tiny-CLOS (SCLOS), written by Gregor Kiczales **?** while he was working at Xeros PARC in 1992 and 1993. Kiczales has been an instrumental researcher and proponent for the use of metaobject protocols (MOPs) as a tool for making computer programs clearer, more efficient, and more robust. The basic tenet is that generic methods or functions are used to manipulate objects, and that these generic objects inspect the nature of the data being passed to them and pass the processing to a specialised function which deals with the task most appropriately. While this is not an exact analogue to the problems we seek to address in this work, there is enough commonality that using tiny-CLOS and its implicit MOP seemed a natural fit. Moreover, tiny-CLOS is the base that many of the object systems used in modern implementations of Scheme are derived from.

I have used Gambit-C (Gambit), a well regarded implementation of Scheme which runs on all of the major platforms (Linux, Unix, OSX, and Android, for example). Gambit incorporates both a fully developed Scheme interpreter and a compiler, and both can readily link programs with external code and libraries which exist either as compiled or as interpreted scheme code. The choice to use Gambit, rather than another version of Scheme, was heavily influenced by the fact that it was the implementation used for the model in Chapter 2, its compiled code was significantly faster than the interpreted code, and that adding hand-crafted C or C++ code for particularly intensive routines is not difficult. Porting the model to other versions of Scheme should be relatively simple; the only potentially awkward issue in using a different scheme implementation would be the use of `define-macro` which is used to provide syntactic structures which are able to detect and respond to inconsistencies and makes using objects simpler and less error prone[1]. In principle, `define-syntax` and other tools for syntactic extension or modification might be substituted for the macros defined in the `framework` file.

[1] At least for *some* types of error!

5.1.2 Encoding strategy

The first step in being able to quantitatively compare configurations of models is to construct a means for encoding the state in some metric space. The trees described briefly in Chapter 3 and more comprehensively in Chapter 4 are elements of a metric space, and since their set of indeterminate variables is quite arbitrary, we can identify cells and groups of cells explicitly with indeterminates, and we can also extend this to the classes which may be used within a run. So, building our set of appropriate configurations, we might start looking at a tree (or set of trees) which are associated with the aggregate global condition:

5.1.3 Partial ordering of trees and nodes

We can construct a partial order on nodes relatively simply. In the first instance we normalise the polynomial labels if necessary (by collecting like terms) and sort nodes using their labels, where the polynomial labels are ordered canonically with indeterminate factors in each term sortedby lexicographic order, and the terms then sorted by degree.[2] If two nodes, possess the same label, they are then ordered by their weights. Should the weights *also* match, they are sorted on the relative order of their set of children, with empty sets taking precedence over sets with children.

The sorting order makes the construction of regular, readable output straightforward, since it largely conforms to one of the common patterns used in mathematics and computer science.

5.1.4 Model implementation

The model is largely similar to the example used in Chapter 3. In the hypothetical example, the domain consists of nine cells containing tree-like plants. These trees are a critical food source for a population of herbivores, and the trees rely on the young herbivores to eat the ripe fruit in order to make their seeds viable (perhaps they have a particularly tough seedcoat). A carnivore which preys solely on juvenile herbivores is introduced into a stable system. In Chapter 3, the introduction of the carnivore (and a subsequent rangeland fire) initiated a sequence of events which, in turn, engendered changes in the configuration of the model as a whole. The implementation described here is not identical to the system described in the paper—some inadequacies of the model presented in the paper have been addressed (such as the inability of the plants to recolonise a cell after they have become extinct in it).

[2]Of course other sorting strategies are possible.

Initialisation

Models in the framework are initialised by constructing a set of agents (instances of submodels) and introducing them into the runqueue. The framework provides for the ability to make an agent "subsidiary" to another agent in one of two ways: the subsidiary agent may remain a member of the global runqueue—only notionally "contained" by its supervisor, or it may be wholly subsidiary and control only passes to it during the time step executed by its supervisor. This second mechanism is targetted at situations where a modelled entity may be comprised of several submodels, such as an animal with different possible representations for its metabolism.

In the case of the example of Chapter 3, we explicitly associate the agents representing trees with particular cells—trees populate the cells, notionally contained, but introduced in their own right into the runqueue. Background quantities of fruit and seeds are introduced into the model as numeric values associated with the cells, and a number of herbivores, adult and juvenile, are also generated and placed at locations within the domain. An initial population of carnivores is also generated, but they are given an initial time substantially after the start of the run, so they will not have control passed to them till the rest of the agents have caught up.

Additional agents within the system, the monitors and loggers, are also instanciated and added to the queue. The monitors will periodically assess the configurations of agents, ensembles of agents and the model as a whole and make changes in the configuration, if required. The loggers will periodically poll the other agents and produce some sort of output or record of data derived from the state of the model.

After the initial cohort of agents have been instanciated and introduced into the runqueue (or in the subsidiary-runqueues of other agents), control is passed to the kernel which then begins to run each agent in turn for an appropriate time step.

Control flow

The most important mechanism controlling the flow of control in the system is the execution loop which plays the role of the kernel. A sorted list of agents is maintained by the system; the head of this list is removed from the list and execution is passed to `model-body` of the agent with an indicated maximum amount of time over which it might run. Agents need not run their whole time step—events may occur which cause them to truncate their turn, returning control to the kernel with an indication of the amount of time they actually used. When the kernel receives control from an agent, it examines the data passed to it by the agent and act accordingly: dead agents may be silently dropped from the queue, for example, and other agents may be reinserted in an appropriate place in the queue for the their next time step.

Control also passes from agents to the kernel when the agent makes queries about or of other agents: a sabre-toothed tiger might want to know whether there are any prey animals within a certain radius. A request goes to the kernel for a list of nearby prey (transferring control back to the kernel). The kernel will examine the list of agents meeting the requirements (spatial location, temporal contiguity) and pass the list—and control—back to the tiger. These control issues are conventional, both in terms of operating system dynamics, and in agent-based modelling.

The maintenance of state data from a superceded representation is rather different. When an agent has entries of maintenance data, it processes each of them in turn by asking it for the data it needs, obtaining the data and passing control and the data to the closure When the closure has finished its update, control returns to the agent, and it either continues in its time step or passes control back to the kernel.

Communication

Where closures are *supposed* to be bound to communicate with only the agent maintaining them, other entities may have other,more direct, channels. Chief amongst these are the connections between subsidiary agents and their containing supervisor. These agents can use class methods to query each other directly. The constraint imposed on maintenance closures exists primarily because they really only exist *in potentia*, and the data they request may actually be synthesised by the agent responsible for their maintenance, rather than the agents which existed at the time the closure was created.

This is an uncomplicated system, more significant issues would arise in a parallelised system.

Classes and structure

The example model follows a traditional agent-based approach using a class hierarchy that reflects the entities we wish to model. Most of the agent classes are organised more by their niche than by their structural form—we may have several "animal-like classes" which may range from individual-based representations to purely equation-based representations. This is motivated by the observation that the kinds of interactions an animal may engage in are substantially different from the interactions associated with a plant, for example. In some cases an equation-based representation of an animal may need to participate in what seems to be an individual-to-individual interaction or visa-versa. These situations need to be examined in more detail and will be addressed in Section **??**

The framework is structured like an older multitasking, multiprocessing operating system which supports an arbitrary number of interleaved processes which may each have a different (varying) priority and which may run for steps of an aribitrary length. There is only one processing queue in the framework, so we can be certain that only one agent will be executing at any given moment. In principle, the relatively small kernel could be replaced by a multithreaded kernel without requiring major changes to the rest of the system. True concurrent processing would require additional facilities for inter-agent communication and ways to ensure that sets of agents could become synchronous when necessary. Similarly, a number of processes (such as numeric integration) might be readily parallelised. In both these instances, it was felt that these changes would come at a cost in terms of the clarity of the code, and an increase in the difficulty of debugging and exploration.

All of the agents and objects in the model are derived from the `<primitive-object>` class in a slightly modified tiny-CLOS[3], this provides a clean distinction between the classes which are associated only with the tiny-CLOS architecture and the classes associated with the model. The basic classes used to build the entities of the model are the `<object>` class and the slightly more complex `<agent>` class. `<agent>` is a subclass of `<object>` and has the ability to run as a process in the run-queue, and to interact directly with the kernel and with other agents in the system. `<object>`s lack that connection to the kernel, and are only able to operate on their own data, unless assistance is provided by some external agency. An example of the distinction between these classes is that a `<patch>` is a spatially explicit `<agent>` element which is primarily used to give context to ecoservices, and its geographic footprint, a polygon or circle, is represented by an `<object>` which can answer queries such as "How far is *p* from your perimeter?" or "How much area do you contain?"

A brief summary describing the role of the classes and their taxonomic relationships

`<object>` —This class sits at the top of the taxonomic tree for all the significant classes in the modelling framework.

- `<model-maintenance>` —These objects (not agents) contain functions with closures which may be used to maintain the data an agent needs to preserve across transitions. An example of this is the contaminant vector preserved across representations in the model of Chapter 2. The function will respond to a number of "flags", namely `'needs?`, `'state`, `'dump` and `'update`.
- `<polygon>`, `<circle>` —These classes are used to delineate regions within the simulation.
- `<simple-metabolism>`, `<metabolism>` —These classes provide the metabolic component of the animal models.
- `<agent>` —Most submodels or components of submodels will be derived from the agent class. `<agent>` provides the basic structure for interacting with the system and with other agents.
  - `<introspection>` —This class provides the basic machinery used for logging agents. Logging agents will periodically poll lists

[3]This class is actually the `<object>` class in the canonical tiny-CLOS (version 1.7), but I really wanted `<object>` to provide an entity which was like an agent, but without the ability to run as an independent dynamic element in the system—hence the name change

of agents and generate output files consisting of things like numeric data or maps.

`<logfile>` —The logfile class is the basis for the time-series output. Logfiles would typically be used to extract data from a modelled data logger, for example.

`<log-data>` —Instances of the log-data class generate datafiles such as might be loaded into matlab or octave.

`<snapshot>` —This class produces sequential files which contain snapshots of the state of components of the system. A snapshot approach is somewhat analogous to an aerial survey of rangeland species.

`<log-map>` —This class generates (sequential) postscript or PNG maps as the model runs

`<monitor>` —Monitors are the components of the model that assess the configuration of the system and triggers changes

blackboard —Blackboards are used to indicate conditions which may influence other agents in the system. If an agent enters a part of its state space where it performs poorly, it may post a notice on a blackboard indicating the condition.

`<tracked-agent>` —Agents which have their *path* tracked through time. The path in question may be geographic, but it could also be some other state value, such as salinity, voter-sentiment, or location

`<thing>` —This class is the basis for agents with specific locations. Individual members of a group, such as whales in a pod, health centres, or seismographs.

`<simple-plant>` —This class is geared to represent simple tree-like plants.

`<animal>`, `<simple-animal>` —These classes provide the basic animal models. The animal classes inherit from `<metabolism>` or `<simple-metabolism>`, respectively.

`<population>` —Implementations of populations could range from "super-individuals" in the sense of **?** through age structured matrix models and systems of differential equations. The major identifying characteristic is that populations cover an area, rather than a distinct location, and the inherent time steps associated with populations are typically longer than those used for subclasses of `<thing>`.

`<ecoservice>` —Ecoservices correspond to resources which may be used or generated by other agents. They might represent anything from local groundwater to numbers of visitors in a museum. Though they are included with the "landscape" classes, ecoservices are not inherently tied to locations.

`<population-system>` —Population-systems are a subclass of ecoservices, and is tailored to representing simple equation based population dynamics within a spatial domain (a dynamic-patch).

`<patch>`, `<dynamic-patch>` —These classes associate a geographic region (a circle or polygon) with a set of ecoservices or a dynamic system of equations.

`<environment>` —environments provide the spatial context for a model. Classes which actually reflect environmental features (biotic or abiotics, according to the inclination of the modeller) would be derived from the environment class.

- `<landscape>` —A landscape associates a terrain (either a function or some sort of digital elevation model) with an environment.
  - `<habitat>` —Habitats are collections of patches which can be viewed as comprising the landscape.
    - `<habitat*>` —This class adds a "global patch" which encompasses the patches inherited from `<habitat>`, and the global patch acts as an aggregated proxy for the more finely resolved elements.

5.1.5 Methods, model bodies and closures

Methods are functions which are specifically associated with instances of a class. Tiny-CLOS supports having a number of distinct methods with the same name, but different arguments, as determined by their type. This means that `Seal` and `Shark` may both have an `Eat` method—moreover, sharks may have two `Eat` methods: one which recognises when the item being eaten is an `Animal`, and one where it is merely an `Object`, which would usually be inedible. This multiple-dispatch means that no explicit test is required in the model code (beyond writing an "object" version which spits out the offending item). If there is only an `Fish` verion of the `Eat` method for seals, they can only eat fish. This makes the extension of classes much less vulnerable to mishandling interactions with instances of other classes. These class methods are the primary mechanism for implementing behaviour in submodels in the system.

Model bodies are methods which are only called by either the kernel or by the subsidiary execution lists embedded in agents. There is no provision for an agent to directly call model bodies.

Closures have been discussed before as mechanisms for implementing "update function" which preserve the essential state variables for particular representations. A typical closure definition might look like

```
(define-update-closure fish-population
                       <FishPop>
                       '(age mass contaminant) ;; state variables
                       '(temp contlevel) ;; required external values
                       (begin
                         (set! mass (fishgrowth dt age mass temp))
                         (set! age (+ age dt))
                         (set! contaminant (fishcont dt mass contaminant contlevel))
                         ))
```

In this example the state-variables age, mass and the contaminant level are all updated in the body; the update closure requires a temperature, `temp`, and a contaminant level, `contaminant`. The macro definition for `define-update-closure` isn't necessarily simple to follow, a simple example of a closure might look like so:

```
(define counter
   (let ((n 0))
      (lambda x
         (cond
           ((null? x) (set! x (+ 1 x))
           ((eqv? x '(reset)) (set! x 0))
           (#t counter))))
           )
)
```

The symbol `counter` is set to point to the function the `lambda` defines, and the state variable `counter` is "global" to this function, rather like a static variable defined outside a function in a C source file.

Functions with closures are used, most importantly, to implement the code run with the `<model-maintenance>` class. Here, an agent will create a closure containing a function which is able to access the variables within the closure, and maintain their values; it may ultimately pass these values back to another agent which will take over the role played by the model maintenance function. Indeed, scheme was chosen specifically because these functions are a natural representation for a fragment of a model which needs to be maintained.[4]

Initialisation methods are run when instances of classes are created using the `make` function. Two types of these functions occur in the code—`initialize` with a *z* and `initialise` with an *s*—and the distiction is that the *z* versions are associated with the tiny-CLOS initialisation chain, and the *s* version is associated with the modelling classes. When an instance is created the `<object>` `initialize` is called, and this chains to the framework's intitialisers. When `<object>` derived class is instanciated, all of its parent initialisers (with an *s*) are called. The initialisers are

`default-object-initialisation` **and** `default-agent-initialisation` which just initialise the state-variables for the object or agent (see `<polygon>` in the file "landscape-methods.scm"). There is no scope for additional processing using the default initialisation.

`object-initialisation-method` **and** `agent-initialisation-method` which defines a method which will be executed when the object is created.

In fact, the only significant differences between the `object-` and `agent-` versions are the error messages and the textual difference in the model code, but the different labels reinforce the different roles of the entities they refer to.

[4]In principle, each class could be comprised of such fragments from the outset, but it seemed that the purpose of the work would be better served by a more traditional architecture.

Model methods are rather like simple functions, except that they are explicitly associated with classes—there are several versions of the model method `dump`, and each is tied to particular a class and its descendants. This ability to have several functions with the same name is made possible by the use of "generic-methods" which maintain a list of actual functions and information about the arguments they expect. The generic function is able to recognise the types of the arguments it is passed and to forward the call on to the appropriate actual implementation. There are two consequences to this approach which ought to be mentioned

Only define one generic method for each set of similar functions Redeclaring a generic-method "(define hunt (generic-method))" cuts any previously declared "hunt" methods (generic or otherwise!) out of the model's code path. To this end, all declarations of generic methods are either in the "declarations.scm" file, which contains the declarations for most of the model-methods or in the "framework-declarations.scm" file which declares the generic-methods for methods required by the lower levels of the framework. Related to this is the problem of defining a method before you have a generic-method declared— the macros in "framework" should catch this.

Don't try and construct methods that match the same argument lists Generally, tiny-CLOS is very good at getting the right method for the job, but there are situations where it can be confusing for it. If you have a situation where there are several possible desired code paths for essentially similar arguments, it is clearer to have an explicit selection made within the body of a single method.

Each agent's `model-body` is run by the kernel at each of the agent's time steps. While there is an `object-initialisation-method` corresponding to an agent's `agent-initialisation-method`, there is no `object-body` since objects are not able to be run by the kernel.

5.2 Interactions

??

The time taken to simulate interactions between individuals in an individual-based model is often a significant proportion of the total run-time, particularly when there are explicit searches for partners to interact with. Population models suffer less from this particular source of overhead since, at least in ecological modelling, populations are often represented as arrays or function whose elements or values are an aggregation of individuals with respect to particular characteristics.

Interactions between individuals are typically implemented or resolved by simulating (at least to some degree) the events which occur in the system being modelled. In contrast, interactions between the components of populations is necessarily less well resolved—the focus becomes the aggregate effects of the interaction, and these are often modelled either using integral transforms (in the case of functional representations) or repeated evaluations over the histograms representing the populations.

We can map population histograms onto piecewise linear functions with the property that the partial integrals of the the linear functions correspond exactly to the partial sums of histograms at each boundary in the histogram. If populations are functions, then we can evaluate the consequences on the population.

5.3 Maintaining state across representations

Unfinished business

5.4 Using trees to assess model configurations

The model periodically generates a "map" of the current configuration as an element in the space of the tree-ring elements of Chapter 4. We use this mapping to calculate the closest candidate configuration from a set of configurations we believe (or know) to represent the system well in a given part of its state space. Because the trees are elements of a ring, we can, in some cases, interpolate between configurations, arriving at possibly advantageous configurations which have not been explicitly specified.

5.4.1 Candidate configurations

5.4.2 Generating current-state trees

5.4.3 Mapping from interpolated configurations to actual configurations

5.5 Further work: cross-representation predation

This discussion has not been implemented in the example model discussed in this section, but is included since the disparity between individual-based predation and population-level predation may be a frequent cause for representation changes. It is possible for us to model predation between individual-based and population-based representations, but to do so requires an approach quite unlike the approaches of conventional individual-based models, where the pursuit, capture and consumption may be explicitly simulated, or the approaches of conventional age-class structured population models which can often be thought of as ht

$m_a(l)$	the member distribution with respect to size of each agent of interest
$G_{i,j}(l,w)$	the gape filter for predator i with respect to prey j
$T(a)$	returns the taxon or type number of agent a
M_a	the total number of members for agent a
$M_i^*(w)$	the sum of all the distributions of agents with a type i
$\bar{M}_i = \int_0^\infty M_i^*(w)dw$	the total number of beasties of type i

5.5.1 An individual-based and analytic approach

Let us consider the folowing "known" attributes in a system

5.5.2 Calculating mortality

Let

$$I_{i,j}(l,w) = M_i^*(l)G_{i,j}(l,w)$$

and

$$J_{i,j}(l,w) = M_j^*(w)G_{i,j}(l,w).$$

$I_{i,j}$ is the raw distribution of pressure of predator i onto prey j, and $J_{i,j}$ is the raw distribution of the vulnerability of prey j to the predator i.

If the constants

$$k_{i,j} = \int_0^\infty \int_0^\infty I_{i,j}(l,w)dldw$$

and

$$h_{i,j} = \int_0^\infty \int_0^\infty J_{i,j}(l,w)dldw$$

are non-zero, they can be used to scale $I_{i,j}$ and $J_{i,j}$ so that they form kernel functions, and we get

$$K_{i,j}(w) = \frac{1}{k_{i,j}} \int_0^\infty I_{i,j}(l,w)dl$$

which is the normalised predatory pressure with respect to size, and the normalised vulnerability

$$H_{i,j}(w) = \frac{1}{h_{i,j}} \int_0^\infty J_{i,j}(l,w)dl.$$

Values of zero in $k_{i,j}$ and $h_{i,j}$ indicate that no predation is possible—usually because M^* has collapsed. In this case we take either (or both) $K_{i,j}(w)$ and $H_{i,j}(w)$ to be zero.

We calculate

$$v_{i,j} = \int_0^\infty K_{i,j}(w)H_{i,j}(w)dw$$

which has a value in the range $[0,1]$. If $v_{i,j}$ is non-zero we can construct the normalised interaction

$$V_{i,j}(w) = \frac{1}{v_{i,j}} K_{i,j}(w) H_{i,j}(w)$$

which indicates the proportion by size of type j subjected to predation from type i at the given length w. Again a zero value for $v_{i,j}$ indicates that no interaction ("diner" or "dinner") are possible.

The function

$$e_{i,j}(w) = M_j^*(w) V_{i,j}(w)$$

can be used to give us the the number

$$E_{i,j} = \int_0^\infty M_j^*(w) V_{i,j}(w) dw$$

which is the exposure of prey population j to the predators in population i. The converse,

$$C_{i,j} = \int_0^\infty M_i^*(w) V_{i,j}(w) dw,$$

is the potential for predation of the predator type i on an "average" prey of type j.

We can then use a predation relationship of some sort to get the raw number of "kills" based on the exposure averaged over the potential volume (or area) of contact per unit of time, which we call $\Omega_{i,j}$, where $\Omega_{i,j} = \bar{M}_i \mathfrak{F}(E_{i,j}/A_j, p_{i,j}) \Delta t$ where $\mathfrak{F}$ is the predation relationship, and $p_{i,j}$ is the parameterisation for the species, and Δt is the timestep, and A_j is the area/volume we divide by to get a density.

We can sum over a predator type

$$\Omega_j^* = \sum_i \Omega_{i,j}$$

to give us the total possible consumption of prey type j.

Alternatively, we can calculate a consumption-by-size distribution and define the function $\omega_{i,j}(w)$, the raw number of kills for a length w on a consumption-by-size basis, by

$$\omega_{i,j}(w) = \bar{M}_i \mathfrak{F}(e_{i,j}(w)/A_j, p_{i,j}) \Delta t.$$

Thus the impact on the prey population (at least those of length w) is

$$\omega_j^*(w) = \sum_i \omega_{i,j}(w)$$

and for the whole population it is

$$\int_0^\infty \omega_j^*(w) dw$$

(or something like that).

Alternatively, we can express things more in the way that it is calculated in the Atlantis model (Fulton [2011]) with

$$Z_i(w) = \frac{g_i M_i^*(w)}{g_i/c_i + \sum_j a_{i,j} e_{i,j}(w)}$$

where $Z_i(w)$ reflects the aggregate clearance rate of a predator of type i if we take c_i to be the "clearance rate" which incorporates the volume it sweeps and a proportion of prey captured and we take g_i to be the predator's growth rate. So $\int_0^\infty Z_i(w) e_{i,j}(w) \Delta t dw$ is the amount of prey of type j consumed by the predators of type i over the interval Δt.

It should be pointed out that the number of types is fairly small compared to the number of agents, and this shouldn't be too onerous a calculation (at least compared to playing it all out individually).

5.6 Apportioning mortality

Mortality can be calculated either by apportioning it to each agent according to the proportion of the global population it represents (and within it, apportioning the mortality to ages in an analogous fashion), or we can apportion mortality to each age in each agent according to how much of the population it represents.

For the agent-by-agent update we have

$$\delta m_a(w) = \Omega^*_{T(a)} \frac{m_a(w)}{M^*_{T(a)}(w)}$$

and for the age-by-age update we have

$$\delta m_a(w) = \omega^*_{T(a)}(w) \frac{m_a(w)}{M^*_{T(a)}(w)}.$$

The new distribution

$$n_a(w) = m_a(w) - \delta m_a(w).$$

In these steps, places where there are no members of size w should be dealt with carefully in the division, and at all points $M^*_{T(a)}(w) \geqslant m_a(w)$.

CHAPTER 6

Conclusion

6.0.1 Modelling over the past few decades

The domain models are expected to cover is increasing in breadth and complexity — ecosystems models, for example, may now be called on to represent large areas, and significant numbers of species. This is not inherently intractable; equation-based modelling is well suited to the modelling of such systems, but we are also often required to include subsystems where equation-based models are a poor fit. Hybrid models, which incorporate both equation-based representations and agent- or individual-based models for those components they are best suited, offer a way to deal with many of these situations. This solution isn't adequate where the nature of the interactions changes the dynamics of the system—classic examples being the effects of parasites or contaminants which alter behaviour or reproductive success. Chapters 2 and 3 showed that changing the representation of a system using a selection strategy based on its state may have benefits both in terms of representational fidelity and in computational cost.

6.0.2 Possible, but not implemented

Record of transitions—indicative of the sensitivities of the system,

6.0.3 Deterministic strategies

benefits—fast

copes with imperatives

rigidity

6.0.4 Adaptive strategies

not without overhead

can handle imperatives, but still overheads (associated with assessment and mapping to and from assessment space)

flexible

6.0.5 Combinations of the two

6.0.6 Parallelism and Distributed models

6.0.7 Into the future

Use of region partitioning in the state space

nearest neighbours

interpolation

The likelihood is that models will continue to increase in the scope of the domains they model as well as the level of detail they are required to simulate. There are issues that arise from both of these eventualities: as the scope of a model's domain increases, the computational requirements of the model typically increase much more rapidly—adding new species or doubling the area may increase the number of interactions by orders of magnitude. While quantum processors may help with some aspects of simulation (such as path resolution or prey selection) to ameliorate this, it seems unlikely that they will provide a magic bullet; in modelling, past experience suggests that the problem any new model addresses will grow to point that it is just barely tractable. If this is true, no real technological advance or methodological insight (such as changing representations!) is really going to do much more that shift the boundary: modellers will still sit impatiently waiting for the model run to finish, wishing they had more memory or a faster CPU.

Perhaps more significant is the effect of increasing demand for process resolution ("realism"). Clearly faster hardware or more efficient algorithms will help address this in a more sustainable way, but it isn't always the case that a more resolved model will improve the correspondence between the model results with the trajectory of a real system. Equation based systems, for example, are usually constructed with parameterisations that incorporate the naturally occuring aggregation of data that time and space provide. Constructing an individual-based model from parameters obtained by observing the activity of individuals may not produce, the sort of large scale dynamics observed in the analytic model or in vivo, and an individual-based model using population-scale parameterisations may fail to exhibit the dynamics observed in individuals.

Models with embedded submodels which have naturally different scales (temporally or spatially) need to be able to ensure that the interactions between elements of the models are well founded: it may be perfectly sound to simulate juvenile cicadas in a forest stand using timesteps of seven or thirteen years for most of their life cycle, but such a timestep will fail to capture the dynamics of the population once they emerge. There is clearly a need for approaches which can adapt to dramatic changes in behaviour, time scales, and spatial range.

The (HERE WHAT?) in this work addresses the problems which arise when dealing with complex systems with components with dynamics which are sensitive to the state of the system. Models can be constructed so that their components can adapt to the most appropriate form for the state of the model through the whole simulation. In practice, it is possible to choose a strategy which optimises for any appropriate measure — speed, fidelity, or the cost of real-world management, for example.

BIBLIOGRAPHY

Michael P Bailey and William G Kemple. The scientific method of choosing model fidelity. In *Proceedings of the 24th conference on Winter simulation*, pages 791–797. ACM, 1992.

Georgiy V Bobashev, D Michael Goedecke, Feng Yu, and Joshua M Epstein. A hybrid epidemic model: combining the advantages of agent-based and equation-based approaches. In *Simulation Conference, 2007 Winter*, pages 1532–1537. IEEE, 2007.

Fabio Boschetti, Claire Richert, Iain Walker, Jennifer Price, and Leo Dutra. Assessing attitudes and cognitive styles of stakeholders in environmental projects involving computer modelling. *Ecological Modelling*, 247:98–111, 2012.

Daniel B Botkin, James F Janak, and James R Wallis. Some ecological consequences of a computer model of forest growth. *The Journal of Ecology*, 60:849–872, 1972a.

D.B. Botkin, J.F. Janak, and J.R. Wallis. Rationale, limitations, and assumptions of a northeastern forest growth simulator. *IBM J. Res. Dev.*, 16:101–116, 1972b.

D.B. Botkin, J.F. Janak, and J.R. Wallis. Some ecological consequences of a computer model of forest growth. *J. Ecol.*, 60:849–872, 1972c.

John B Calhoun. Death squared: the explosive growth and demise of a mouse population. *Proceedings of the Royal Society of Medicine*, 66(1 Pt 2):80, 1973.

William John Chivers. *Generalised, parsimonious, individual-based computer models of ecological systems*. University of Newcastle, 2009.

Sandro Jerônimo de Almeida, Ricardo Poley Martins Ferreira, Álvaro E Eiras, Robin P Obermayr, and Martin Geier. Multi-agent modeling and simulation of an aedes aegypti mosquito population. *Environmental Modelling & Software*, 25(12):1490–1507, 2010.

D.L. DeAngelis and L.J. Gross, editors. *Individual-Based Models and Approaches in Ecology: Populations, Communities and Ecosystems*. Chapman and Hall, isbn-10: 0412031612 edition, 1992.

D.L. DeAngelis, L.J. Gross, M.J. Huston, W.F. Wolff, D.M. Fleming, E.J. Comiskey, and S.M. Sylvester. Landscape modelling for everglades ecosystem restoration. *Ecosystems*, 1:64–75, 1998.

Donald Lee DeAngelis. Model for the movement and distribution of fish in a body of water. Technical report, Oak Ridge National Lab., Tenn.(USA), 1978.

Timothy DelSole and Jagadish Shukla. Model fidelity versus skill in seasonal forecasting. *Journal of Climate*, 23(18):4794–4806, 2010.

AP Dobson. The population biology of parasite-induced changes in host behavior. *Quarterly Review of Biology*, pages 139–165, 1988.

Bret D Elderd, Jonathan Dushoff, and Greg Dwyer. Host-pathogen interactions, insect outbreaks, and natural selection for disease resistance. *The American Naturalist*, 172 (6):829–842, 2008.

Stefano Farolfi, Jean-Pierre Müller, and Bruno Bonté. An iterative construction of multi-agent models to represent water supply and demand dynamics at the catchment level. *Environmental modelling & software*, 25(10):1130–1148, 2010.

E.A. Fulton. Approaches to end-to-end ecosystem models. *Journal of Marine Systems*, 81(1):171–183, 2010.

E.A. Fulton, 2011. Personal communication.

E.A. Fulton, A.D.M. Smith, and A.E. Punt. Which ecological indicators can robustly detect effects of fishing. *ICES Journal of Marine Science*, 62(3):540–551, 2004.

EA Fulton, R Gray, M Sporcic, R Scott, and M Hepburn. Challenges of crossing scales and drivers in modelling marine systems. *18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, July*, pages 2108–2114, 2009.

E.A. Fulton, R. Gray, M. Sporcic, R. Scott, L.R. Little, M. Hepburn, B. Gorton, B. Hatfield, M. Fuller, T. Jones, W. De la Mare, F. Boschetti, K. Chapman, P. Dzidic, G. Syme, Dambacher J, and D. McDonald. Ningaloo collaboration cluster: Adaptive futures for ningaloo. Technical Report 5.3, Ningaloo Collaboration Cluster, Hobart, Tasmania, October 2011. URL `http://www.ningaloo.org.au/www/en/NingalooResearchProgram/Background.htm`.

R. Gray, E. Fulton, R. Little, and R. Scott. Ecosystem model specification within an agent based framework. Final Report 16, CSIRO Australia, Hobart, Tasmania, 2006a. ISBN 1 921061 80 4 (pbk), ISBN 1 921061 82 0 (pdf).

R Gray, EA Fulton, LR Little, and R Scott. *Operating model specification within an agent based framework. North West Shelf Joint Environmental Management Study Technical Report*. Number 16 in CSIRO-CMAR NWSJEMS Technical Reports. CSIRO, Hobart, Tasmania, Hobart, Tasmania, 2006b. ISBN 1 921061 80 4 (pbk), ISBN 1 921061 82 0 (pdf).

Randall Gray and Simon Wotherspoon. Increasing model efficiency by dynamically changing model representations. *Environ. Model. Softw.*, 30:115–122, April 2012. ISSN 1364-8152. doi: 10.1016/j.envsoft.2011.08.012.

Randall Gray, Elizabeth A. Fulton, and Richard Little. Human-ecosystem interaction in large ensemble-models. In Alexander Smajgl and Olivier Barreteau, editors, *Empirical Agent-Based Modelling - Challenges and Solutions*, pages 53–83. Springer New York, 2014a. ISBN 978-1-4614-6133-3. doi: 10.1007/978-1-4614-6134-0_4.

Randall Gray, Elizabeth A. Fulton, and Richard Little. Human-ecosystem interaction in large ensemble-models. In Alexander Smajgl and Olivier Barreteau, editors, *Empirical Agent-Based Modelling - Challenges and Solutions*, pages 53–83. Springer New York, 2014b. ISBN 978-1-4614-6133-3. doi: 10.1007/978-1-4614-6134-0_4.

V. Grimm and S.F. Railsback. *Individual-based Modeling and Ecology*. Princeton University Press, Princeton, New Jersey, 2005.

Volker Grimm, Uta Berger, Finn Bastiansen, Sigrunn Eliassen, Vincent Ginot, Jarl Giske, John Goss-Custard, Tamara Grand, Simone K. Heinz, Geir Huse, Andreas Huth, Jane U. Jepsen, Christian Jørgensen, Wolf M. Mooij, Birgit Müller, Guy Pe'er, Cyril Piou, Steven F. Railsback, Andrew M. Robbins, Martha M. Robbins, Eva Rossmanith, Nadja Rüger, Espen Strand, Sami Souissi, Richard A. Stillman, Rune Vabø, Ute Visser, and Donald L. DeAngelis. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198 (1-2):115 – 126, 2006. ISSN 0304-3800. doi: DOI:10.1016/j.ecolmodel.2006.04.023. URL `http://www.sciencedirect.com/science/article/B6VBS-4K606T7-3/2/1dad6192bec683f32fce6dee9d665b51`.

C.J. Harvey, S.P. Cox, T.E. Essington, S. Hansson, and J.F. Kitchell. An ecosystem model of food web and fisheries interactions in the baltic sea. *ICES Journal of Marine Science*, 60:939–950, 2003.

M. Huston, D.L. DeAngelis, and W. Post. New computer models unify ecological theory. *BioScience*, 38:682–691, 1988a.

Michael Huston, Donald DeAngelis, and Wilfred Post. New computer models unify ecological theory. *BioScience*, 38(10):682–691, 1988b.

L.R. Little, E.A. Fulton, R. Gray, D. Hayes, R. Scott, A.D. McDonald, and K. Sainsbury. Management strategy evaluation results and discussion for the north west shelf. Final Report 14, CSIRO Australia, Hobart, Tasmania, 2006. ISBN 1 921061 74 X (pbk), ISBN 1 921061 76 6 (pdf).

V. Lyne, R. Gray, K. Sainsbury, and R. Scott. Integrated biophysical model investigations. Final report, CSIRO Australia, Division of Fisheries, Hobart, Tasmania, 1994a.

V. Lyne, R. Gray, K. Sainsbury, and R. Scott. Integrated biophysical model investications. Final report, CSIRO Australia, Division of Fisheries, Hobart, Tasmania, 1994b.

Thomas Robert Malthus. *An Essay on the Principle of Population*. Library of Economics and Liberty, Internet, 16 feb 2010 edition, 1798. URL `http://www.econlib.org/library/Malthus/malPop.html`. First edition, originally published by J. Johnson, London.

L. Monte. A methodological approach to develop *contaminant migration-population effects* models. *Ecological Modelling*, 220:3280–3290, 2009.

Robert Poulin. Meta-analysis of parasite-induced behavioural changes. *Animal Behaviour*, 48(1):137–146, 1994.

Kenneth A. Rose, J. Icarus Allen, Yuri Artioli, Manuel Barange, Jerry Blackford, François Carlotti, Roger Cropp, Ute Daewel, Karen Edwards, Kevin Flynn, Simeon L. Hill, Reinier HilleRisLambers, Geir Huse, Steven Mackinson, Bernard Megrey, Andreas Moll, Richard Rivkin, Baris Salihoglu, Corinna Schrum, Lynne Shannon, Yunne-Jai Shin, S. Lan Smith, Chris Smith, Cosimo Solidoro, Michael St. John, and Meng Zhou. End-to-end models for the analysis of marine ecosystems: Challenges, issues, and next steps. *Marine and Coastal Fisheries: Dynamics, Management, and Ecosystem Science*, 2:115 – 130, 2010.

M. Scheffer, J.M. Baveco, D.L. DeAngelis, and K.A. Rose. Super-individuals: a simple solution for modelling large populations on an individual basis. *Ecological Modelling*, 80:161–170, 1995.

Gerry E Swan, Richard Cuthbert, Miguel Quevedo, Rhys E Green, Deborah J Pain, Paul Bartels, Andrew A Cunningham, Neil Duncan, Andrew A Meharg, J Lindsay Oaks, et al. Toxicity of diclofenac to gyps vultures. *Biology letters*, 2(2):279–282, 2006.

Jan C Thiele and Volker Grimm. Netlogo meets r: Linking agent-based models with a toolbox for their analysis. *Environmental Modelling & Software*, 25(8):972–974, 2010.

Christian Ernest Vincenot, Francesco Giannino, Max Rietkerk, Kazuyuki Moriya, and Stefano Mazzoleni. Theoretical considerations on the combined use of system dynamics and individual-based modeling in ecology. *Ecological Modelling*, 222(1):210 – 218, 2011a. ISSN 0304-3800. doi: DOI:10.1016/j.ecolmodel.2010.09.029. URL `http://www.sciencedirect.com/science/article/B6VBS-518MX70-2/2/199628e93c44d13863a48b3299472a32`.

Christian Ernest Vincenot, Francesco Giannino, Max Rietkerk, Kazuyuki Moriya, and Stefano Mazzoleni. Theoretical considerations on the combined use of system dynamics and individual-based modeling in ecology. *Ecological Modelling*, 222(1):210 – 218, 2011b. ISSN 0304-3800. doi: DOI:10.1016/j.ecolmodel.2010.09.029. URL `http://www.sciencedirect.com/science/article/B6VBS-518MX70-2/2/199628e93c44d13863a48b3299472a32`.

Carl Walters, Josh Korman, Lawrence E Stevens, and Barry Gold. Ecosystem modeling for evaluation of adaptive management policies in the grand canyon. *Conservation Ecology*, 4(2):1, 2000.

Carl J Walters and Steven JD Martell. *Fisheries ecology and management*. Princeton University Press, 2004.

Richard MP Ward and Charles J Krebs. Behavioural responses of lynx to declining snowshoe hare abundance. *Canadian Journal of Zoology*, 63(12):2817–2824, 1985.

webpage, August 2009. URL `http://www.ningaloo.org.au/www/en/NingalooResearchProgram/Background.htm`.

W. F. Wolff. An individual-oriented model of a wading bird nesting colony. *Ecological Modelling*, 72(1-2):75 – 114, 1994. ISSN 0304-3800. doi: DOI:10.1016/0304-3800(94)90146-5. URL `http://www.sciencedirect.com/science/article/B6VBS-48YNSFT-2X/2/09ef35a0a8e95b3261adfc540cd87a23`.

WS Yip and Thomas E Marlin. The effect of model fidelity on real-time optimization performance. *Computers & chemical engineering*, 28(1):267–280, 2004.

Sarah M Zala and Dustin J Penn. Abnormal behaviours induced by chemical pollution: a review of the evidence and new challenges. *Animal Behaviour*, 68(4):649–664, 2004.

Shu Zhang, Qi Zhu, and Amit K. Roy-Chowdhury. Adaptive algorithm and platform selection for visual detection and tracking. *CoRR*, abs/1605.06597, 2016. URL `http://arxiv.org/abs/1605.06597`.