Chapter 1

INTRODUCTION

Specific terms, types of model, motivation, basic concepts

1.1 New Section

1.2 New Section

Here we have started a new page to show how the headers work. The text in the header should be the last section title declared at the end of the current page.

This new paragraph shows how to set index items and subindex items.

1.2.1 New Subsection

Here's a subsection with some simple maths $a^2 + b^2 = c^2$.

subsubsection

Here's a subsubsection...ooooooohh....wow wee!!!!!!

Some more text to check indent and show how references work [1].

Here's how we place a figure (Figure 1.1) on the page.

Figure 1.1: The UTas logo

And finally, here's a table example (Table 1.1).

$n =$	2	3	4	5
c (rad/day)	1.67	0.52	0.06	-0.17
period (days)	3.75	12.00	100.00	37.50

Table 1.1: A simple table

Chapter 2

Increasing model efficiency by dynamically changing model representations

Randall Gray[1]

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.

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

2.1 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 (**?**], **?**]). 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 **?**], **?**] and **?**]). 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 of Population*** (**?**). 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 **?**], 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 (**?**], **?**], **?**] **?**], **?**], **?**]) 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 **?**], and **?**] 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. **?**] 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.

?] 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 **?**] and **?**] have demonstrated that large models of ecosystems can be modelled this way. **?** 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 (**?** ; **?** ; 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. **CiteauthorLyne94:1** assessed the potential of contaminants originating in industrial waste percolating through the food chain into commercially exploited fish stocks. **?** developed a regional model to assess managment 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 **?** and **?** include contaminant transport, uptake and depuration modelling, with behavioural sensitivity to contaminants. In **?** , 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. **?**] 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. **?** 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 (**?**] and **?**], for example).

2.2 Overview: an *ODD* model description

The *ODD* protocol [**?**] 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.

2.2.1 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 2.2.1). This model exhibits fundamental attributes of larger studies of pollutant/ecosystem interactions (**?** and **?**) 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.

2.2.2 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.

2.2.3 Process overview and scheduling

Simulations were run with 90 minute timesteps for a period representing twelve years. At each time-step, each instance of a submodel is rostered 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 timestep and an apply the uptake-depuration equation. At the end of a timestep 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 timestep 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.

2.3 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 **?**], 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.

2.4 Details

2.4.1 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.

2.4.2 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 $\mathbf{l}(t) = 10^5 (\cos(2t\pi/365.25), \sin(2t\pi/365.25))$.

2.4.3 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.

Table 2.1: Parameters associated with individual movemement

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

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

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

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

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

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 timestep to determine whether a change in representation is appropriate.

When a population disaggregates 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.

2.5 Results

The data presented in section 2.5.1 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 thet 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,

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

2.5.1 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.

2.5.2 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.	$r = i$	$r = m$	$r = p$
$\hat{\sigma}_{ir}$	0.0083	0.0084	0.0534
$\bar{\sigma}_{ir}$	0.0024	0.0024	0.0096
$\hat{\sigma}_{mr}$	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.

2.5.3 Sensitivity to the shape of the plume

We will use the same notation as Section 2.5.2 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

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

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.

2.5.4 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 supressing 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.

2.6 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.

2.6.1 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 2.1. 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}.

2.6.2 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 accomodate 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.

2.6.3 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 accomodate this when changing to a population-based representation.

In section 2.6.2, 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 2.5 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.

2.6.4 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[2] 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 er-

[2]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.

ror 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.

2.7 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 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 timesteps according to what they are simulating, or perhaps even adaptively set an appropriate timestep 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.

Chapter 3

A commutative rng over a set of trees with a metric space

3.1 Introduction

The subject of this paper arose in course of trying to find a means of using responses to survey questions to seed a population of "virtual individuals" in simulation models which attempt to incorporate the way public opinion changes in response to policy actions. This structure also has application in controlling simulation models which permit the representation of submodels to change in response to their own local state and the local states of other submodels.

More generally we will be considering trees comprised of labelled nodes with associated values. Each such node may have an arbitrary number of children.

3.2 Conventions and preliminary definitions

Generally, we will use lower case, boldface symbols to denote a node (or tree); upper case, boldface symbols to denote sets (particularly sets of nodes); and other symbols (such as x) will typically refer to numbers or rational polynomials. Elements of a node, $\boldsymbol{u}$ will be identified using an appropriate subscript, namely $\boldsymbol{u}_v$, for the node's value, $\boldsymbol{u}_{\mathsf{P}}$ for its label and $\boldsymbol{u}_{\boldsymbol{E}}$ for its set of extensions. We will take $\mathsf{P}_{\boldsymbol{A}}$ to be the field of rational multivariate polynomials with variables taken from a set of symbols $\boldsymbol{A}$.

Definition 3.2.1. *A node, $\boldsymbol{u}$, is a representative of a recursive structure of the form $(v, p, \boldsymbol{E})$ where $v \in \mathbb{F}, p \in \mathsf{P}_{\boldsymbol{A}}$, and its extension set, $\boldsymbol{E} \subset \boldsymbol{T}$, contains no two elements with the same label. Without necessarily limiting ourselves, we will take $\mathbb{F}$ to be $\mathbb{R}$. Nodes or trees with $\boldsymbol{E} = \emptyset$, the empty set, will be called* simple nodes, simple trees, *or* leaf nodes, *and simple nodes which also have scalar polynomials as their labels may be referred to as* scalar nodes *or* scalar trees. *The domain of trees, $\boldsymbol{T}$, is the collection of (acyclic) trees with a finite*

number of nodes of this form. We will make use of a special element in $\boldsymbol{T}$, $\mathbf{O} = (0, 0, \emptyset)$, *which is an analogue of zero we will refer to as the* zerotree.

Any two nodes are said to be *compatible* if they have the same label, or at least one of the nodes is the $\mathbf{O}$. Two trees are compatible if their root nodes are compatible. This property is largely analogous with compatibility in matrices. 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.

Definition 3.2.2. *For* $\boldsymbol{u} \in \boldsymbol{T}$ *we define the function*

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

which gives us the depth of the tree.

Definition 3.2.3. *We will also define for* $\boldsymbol{u} \in \boldsymbol{T}$,

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

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

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

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 3.2.5. *The* overlap *between two trees is defined*

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

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* $\|\boldsymbol{u}\|_{\mathsf{T}} = \|\boldsymbol{v}\|_{\mathsf{T}} = \operatorname{overlap}(\boldsymbol{u}, \boldsymbol{v})$.

3.3 Addition and Scalar Multiplication

We will define scalar multiplication of the trees in $\boldsymbol{T}$, 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.

Definition 3.3.1. *Given $a \in \mathbb{F}$ and $\boldsymbol{u} \in \boldsymbol{T}$, we define*

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

and we observe that the notation $-1\,\boldsymbol{u} \equiv -\boldsymbol{u}$ is consistent with common use.

Definition 3.3.2. *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*

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

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

and

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

Definition 3.3.3. *For convenience, we define analogues of several of the above relations for nodes to implicitly refer to the extension sets of those nodes.*

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

$$\boldsymbol{L}(\boldsymbol{u}) \equiv \boldsymbol{L}(\,\boldsymbol{u}_{\boldsymbol{E}})$$
$$\boldsymbol{u}|_{\boldsymbol{L}(\boldsymbol{v})} \equiv\ \boldsymbol{u}_{\boldsymbol{E}}|_{\boldsymbol{L}(\,\boldsymbol{v}_{\boldsymbol{E}})}$$
$$\boldsymbol{u}|_{\overline{\boldsymbol{L}}(\boldsymbol{v})} \equiv\ \boldsymbol{u}_{\boldsymbol{E}}|_{\overline{\boldsymbol{L}}(\,\boldsymbol{v}_{\boldsymbol{E}})}$$

Definition 3.3.4. *For compatible nodes $\boldsymbol{u}$ and $\boldsymbol{v} \in \boldsymbol{T}$,*

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

Definition 3.3.5 (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 first and second operands and the set obtained by applying the operator to compatible pairs from the first and second set. This notation is used to express the restriction of addition to compatible nodes in the extensions of the root nodes. Let*

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

3.4 Vector space

Proposition 3.4.1. *$\boldsymbol{T}$, 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 $a, b \in \mathbb{F}$, and that addition $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$ are 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 O is its own inverse, since $\mathsf{O} + \mathsf{O} = \mathsf{O}$ by definition.

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

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

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 extension nodes, that is to say depth$(\boldsymbol{v}) = 2$). Then

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

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 $-\boldsymbol{u}$.

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

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

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

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

Multiplicative identity element

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

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

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

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

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

and, if the extensions are all simple nodes,

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

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

Now suppose that 1 is the multiplicative inverse 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

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

.

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

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

Commutativity

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

The commutativity of addition involving 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

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

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

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

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

So,

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

since addition in $\boldsymbol{T}$ is commutative. If we can demonstrate that the expression for the extension set 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 $\{\boldsymbol{r}+\boldsymbol{s} : \boldsymbol{r} \in \boldsymbol{u}_{\boldsymbol{E}} \text{ and } \boldsymbol{r}_{\mathsf{P}} \in \boldsymbol{L}(\boldsymbol{v}) \text{ and } \boldsymbol{s} \in \boldsymbol{v}_{\boldsymbol{E}} \text{ and } \boldsymbol{s}_{\mathsf{P}} \in \boldsymbol{L}(\boldsymbol{u})\}$ 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 $\boldsymbol{u}|_{\overline{\boldsymbol{L}}(\boldsymbol{v})}$ and $\boldsymbol{v}|_{\overline{\boldsymbol{L}}(\boldsymbol{u})}$ doesn't affect the result, thus, addition must be commutative for all $\boldsymbol{u}$ where 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

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

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 $n+1$.

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

Recall that

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

so, letting

$$\boldsymbol{B} = \boldsymbol{u}_{\boldsymbol{E}} \boxplus \boldsymbol{v}_{\boldsymbol{E}}$$

we get

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

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

Notice that the elements of all the sets which comprise the extension set, those in $\boldsymbol{B}$ and in $\boldsymbol{w}_{\boldsymbol{E}}$, 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 $\mathbb{F}$

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

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

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

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

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

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

becomes

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

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,

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

.

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 $n = \min(j,k)$, $a \in \mathbb{F}$, 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

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

.

Notice that the component sets of the extension to $\boldsymbol{u}+\boldsymbol{v}$, namely $\{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{u}|_{\overline{\boldsymbol{L}}(\boldsymbol{v})}\}$, $\{a\,\boldsymbol{e} : \boldsymbol{e} \in \boldsymbol{v}|_{\overline{\boldsymbol{L}}(\boldsymbol{u})}\}$ and $\{a(\boldsymbol{r}+\boldsymbol{s}) : \boldsymbol{r} \in \boldsymbol{u}|_{\boldsymbol{L}(\boldsymbol{v})} \text{ and } \boldsymbol{s} \in \boldsymbol{v}|_{\boldsymbol{L}(\boldsymbol{u})}\}$ 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 $\mathbb{F}$

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:

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

Nodes with a depth of two are slightly more complicated,

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

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

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

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

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

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

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

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

3.5 Seminorms, norms and metrics

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.

3.5.1 $\boldsymbol{T}$ and its seminorm

Definition 3.5.1. *We define the absolute value of a node to be*

$$\langle\,\boldsymbol{u}\rangle = \begin{cases} 0 & \textit{if } \boldsymbol{u} = \mathbf{O} \\ |\,\boldsymbol{u}_v| & \textit{if } \boldsymbol{u}_{\boldsymbol{E}} = \emptyset \\ |\,\boldsymbol{u}_v| + \sum_{\boldsymbol{e}\in\,\boldsymbol{u}_{\boldsymbol{E}}}\langle\,\boldsymbol{e}\rangle & \textit{otherwise.}\end{cases}$$

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 $\langle\,\boldsymbol{u}\rangle \geq 0$ 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 3.5.1. *For $a\in\mathbb{F}$ and $\boldsymbol{u}\in\boldsymbol{T}, |a|\langle\,\boldsymbol{u}\rangle = \langle a\,\boldsymbol{v}\rangle$.*

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

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

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

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 depth$(\,\boldsymbol{u}) = n+1$, we

have

$$\begin{aligned}|a|\langle \boldsymbol{u}\rangle &= |a|((|\,\boldsymbol{u}_v| + \sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}} \langle \boldsymbol{e}\rangle))\\ &= |a||\,\boldsymbol{u}_v| + \sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}} |a|\langle \boldsymbol{e}\rangle\end{aligned}$$

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

$$\begin{aligned}&= ||a|\,\boldsymbol{u}_v| + \sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}} \langle |a|\,\boldsymbol{e}\rangle\\ &= |a\,\boldsymbol{u}_v| + \sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}} \langle a\,\boldsymbol{e}\rangle\\ &= \langle a\,\boldsymbol{u}\rangle.\end{aligned}$$

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

Proposition 3.5.2. *For* $\boldsymbol{u}$ *and* $\boldsymbol{v} \in \boldsymbol{T}, \langle \boldsymbol{u} + \boldsymbol{v}\rangle \leq \langle \boldsymbol{u}\rangle + \langle \boldsymbol{v}\rangle$.

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

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

$$\langle \boldsymbol{u} + \boldsymbol{v}\rangle = \langle(\boldsymbol{u}_v + \boldsymbol{v}_v,\, \boldsymbol{u}_{\mathsf{P}}, \emptyset)\rangle = |\,\boldsymbol{u}_v + \boldsymbol{v}_v|$$

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

$$|\,\boldsymbol{u}_v + \boldsymbol{v}_v| \leq |\,\boldsymbol{u}_v| + |\,\boldsymbol{v}_v| = \langle \boldsymbol{u}\rangle + \langle \boldsymbol{v}\rangle.$$

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

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

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

$$\sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}\boxplus \boldsymbol{v}_{\boldsymbol{E}}} \langle \boldsymbol{e}\rangle$$

has a depth of n or less, so

$$\sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}\boxplus \boldsymbol{v}_{\boldsymbol{E}}} \langle \boldsymbol{e}\rangle \leq \sum_{\boldsymbol{e}\in \boldsymbol{u}_{\boldsymbol{E}}} \langle \boldsymbol{e}\rangle + \sum_{\boldsymbol{e}\in \boldsymbol{v}_{\boldsymbol{E}}} \langle \boldsymbol{e}\rangle.$$

This implies that

$$\langle \boldsymbol{u} + \boldsymbol{v} \rangle \leq \Big[|\boldsymbol{u}_v| + |\boldsymbol{v}_v| + \sum_{\boldsymbol{e} \in \boldsymbol{u}_{\boldsymbol{E}}} \langle \boldsymbol{e} \rangle + \sum_{\boldsymbol{e} \in \boldsymbol{v}_{\boldsymbol{E}}} \langle \boldsymbol{e} \rangle \Big];$$

rearranging we get

$$\langle \boldsymbol{u} + \boldsymbol{v} \rangle \leq \Big[|\boldsymbol{u}_v| + \sum_{\boldsymbol{e} \in \boldsymbol{u}_{\boldsymbol{E}}} \langle \boldsymbol{e} \rangle \Big] + \Big[|\boldsymbol{v}_v| + \sum_{\boldsymbol{e} \in \boldsymbol{v}_{\boldsymbol{E}}} \langle \boldsymbol{e} \rangle \Big]$$

and hence

$$\langle \boldsymbol{u} + \boldsymbol{v} \rangle \leq \langle \boldsymbol{u} \rangle + \langle \boldsymbol{v} \rangle.$$

□

Corollary 3.5.1. *The absolute value forms a seminorm on $\boldsymbol{T}$.*

Proof. Propositions, 3.5.1 and 3.5.2, 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 $\mathfrak{O} = \{ \boldsymbol{o} \in \boldsymbol{T} : \langle \boldsymbol{o} \rangle = 0 \}$, and observe that for any $\boldsymbol{e} \in \boldsymbol{T}$, and $\boldsymbol{o} \in \mathfrak{O}$ the equation $\langle \boldsymbol{e} + \boldsymbol{o} \rangle = \langle \boldsymbol{e} \rangle$ 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

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

.

For the moment, we will satisfy ourselves with constructing a rng of these trees, and define a multiplicative operator without a corresponding identity. First we define the pairwise multiplication of the elements in two sets of nodes which is analogous to the $\boxplus$ operator used in the addition of trees.

Definition 3.5.2. *For sets of nodes $\boldsymbol{B}, \boldsymbol{C}$, we define*

$$\boldsymbol{B} \boxtimes \boldsymbol{C} = \{ \check{\boldsymbol{f}} \cdot \check{\boldsymbol{g}} : \check{\boldsymbol{f}} \in \boldsymbol{B}, \check{\boldsymbol{g}} \in \boldsymbol{C} \}$$

where $\cdot$ is the multiplicative operator for trees, and we also define

$$\sigma(p, \boldsymbol{B}, \boldsymbol{C}) = \{ \Sigma_{\check{\boldsymbol{f}}_{\mathsf{P}} = p} : \check{\boldsymbol{f}} \in \boldsymbol{B} \boxtimes \boldsymbol{C} \}$$

and

$$\boldsymbol{B} \boxdot \boldsymbol{C} = \{ \sigma(p, \boldsymbol{B}, \boldsymbol{C}) : p \in \boldsymbol{L}(\boldsymbol{B} \boxtimes \boldsymbol{C}) \}$$

Clearly, if either of the operands is the empty set, the result for both operators must also be the empty set. The purpose of $\boldsymbol{B} \boxdot \boldsymbol{C}$ is to collect the compatible

extension nodes within the product in much the same way as we collect terms in the product of two polynomials.

We can see that if tree multiplication is commutative, then $\boxtimes$ *must also be commutative. The operator* $\boxdot$ *is also commutative, since the addition of trees is commutative.*

Definition 3.5.3. *For nodes* $\check{\boldsymbol{u}}, \check{\boldsymbol{v}} \in \check{\boldsymbol{T}}$*, we define their product,* $\check{\boldsymbol{u}} \cdot \check{\boldsymbol{v}}$*, by*

$$\check{\boldsymbol{u}}, \check{\boldsymbol{v}} = \begin{cases} \check{\mathsf{O}} & \textit{if either of the nodes is } \check{\mathsf{O}} \\ (\check{\boldsymbol{u}}_{\nu}\, \check{\boldsymbol{v}}_{\nu},\, \check{\boldsymbol{u}}_{\mathsf{P}}\, \check{\boldsymbol{v}}_{\mathsf{P}},\, \check{\boldsymbol{u}}_{\boldsymbol{E}} \boxdot \check{\boldsymbol{v}}_{\boldsymbol{E}}) & \textit{otherwise} \end{cases} \tag{3.1}$$

commutative,

Proposition 3.5.3. *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 rational polynomials is commutative.

Suppose that one or both of the nodes has a depth of $n+1$. The extension sets of the nodes all have depths of n or less, so the elements of the extension set of the product must be independent of the order of the operators in the multiplication, and both scalar multiplication and polynomial multiplication 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 3.5.4. *Multiplication of trees in* $\check{\boldsymbol{T}}$ *is associative.*

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

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

In the case where all of the nodes are simple, it is evident that they must be associative, since scalar multiplication is associative, and the multiplication of rational polynomials is associative.

Suppose that one or more of the nodes has a depth of $n+1$ or less. Then

$$\check{\boldsymbol{u}} \cdot (\check{\boldsymbol{v}} \cdot \check{\boldsymbol{w}}) = (\check{\boldsymbol{u}}_{\nu},\, \check{\boldsymbol{u}}_{\mathsf{P}},\, \check{\boldsymbol{u}}_{\boldsymbol{E}}) \cdot (\check{\boldsymbol{v}}_{\nu}\, \check{\boldsymbol{w}}_{\nu},\, \check{\boldsymbol{v}}_{\mathsf{P}}\, \check{\boldsymbol{w}}_{\mathsf{P}},\, \check{\boldsymbol{v}}_{\boldsymbol{E}} \boxdot \check{\boldsymbol{w}}_{\boldsymbol{E}})$$

Suppose that one or both of the nodes has a depth of $n+1$. The extension sets of the nodes all have depths of n or less, so the elements in the extension set of their product must be independent of the order of the bracketting in the multiplication. Hence the multiplication of nodes with a depth of $n+1$ must be associative. By induction, we can say that multiplication of trees of arbitrary depth is associative. multiplication in $\check{\boldsymbol{T}}$. □

Proposition 3.5.5. *Tree-multiplication distributes over tree-addition in* $\check{\boldsymbol{T}}$.

Intuitively, the properties of operations on the scalar values and the rational polynomial labels of each node suggests that this must be true.

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.

Multiplication by nodes which have a depth less than two clearly distributes over addition, since the multiplication of scalars and of polynomials distributes over their addition, and there are no extension sets to complicate matters. The case of multiplication where one of the trees has a depth of less than two and the other has an arbitrary depth also distributes, since the extension set

Let us consider the case of multiplication involving a node, $\check{\boldsymbol{u}}$, with a depth of two, that is to say that $\check{\boldsymbol{u}}$ has an extension set that contains only simple nodes. Then for compatible summands with depths of two or less,

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

but $\check{\boldsymbol{u}}_\mathsf{P}$ must equal $\check{\boldsymbol{v}}_\mathsf{P}$, so

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

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 factors, $\check{\boldsymbol{u}}$, have depths of $n+1$ or less. Then

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

all of the nodes in the expression for the extension set are of depth less than or equal to n, so

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

□

Corollary 3.5.2. *$\check{\boldsymbol{T}}$ with tree addition and tree multiplication is a commutative rng.*

Proof. Propositions 3.5.3, 3.5.4, and 3.5.5 are sufficient to establish that it is a commutative rng. □

3.6 For the wildly optimistic

3.6.1 Our space is a complete metric space . . . Hah!

Proposition 3.6.1. *Every Cauchy sequence in $\check{\boldsymbol{T}}$ converges to a tree in $\check{\boldsymbol{T}}$.*

Proof. . . . □

3.7 Discussion

The way the extensions of a node influence the value of a node can be perverse if the data in these nodes aren't appropriate. The overriding rule is that extensions to a node *must be relevant to that node*. For extrapolation from surveys this is a matter associated with the coding of data, for simulation models (like MSE models or evaluating climate adaptation strategies) it is a matter of keeping track of influences and connections appropriately. I don't believe that this is entirely a trivial matter, but I also don't believe it is a complex one.

3.8 Application

MSE models incorporating public opinion (surveys)

Adaptive decision making (kind of like a Baysian net, I suppose)

Adaptive behaviour (things can learn by adjusting weightings or adding new branches)

Constructing or assessing phylogenetic trees?

Chapter 4

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.

[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)

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.

4.1 Introduction

The case has been made for developing systems with submodels that change their representation according to their state. [**?**] identify 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 [**?**] and [**?**]. 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 [**?** **?**]: 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 [**?**] are better suited to an *IB* approach. This argument is closely tied to the notion of model fidelity. Following [**?**], 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 [**?**] 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 [**?**]. 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 [**?**] 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 [**?**] and [**?**] 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. [**?**] 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.

4.2 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 [**?**]. 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 [**?**] 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.

4.2.1 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

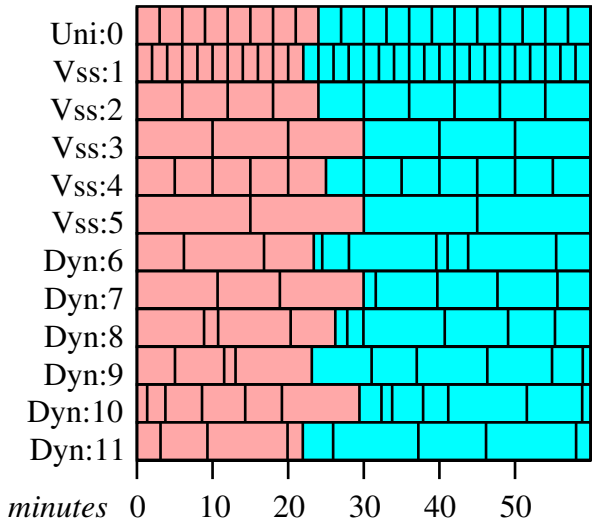

Figure 4.1: Time scheduling strategies. Red boxes represent time steps that have already passed, blue boxes represents scheduled time steps that have not yet been run. "Uni:" and "Vss:" submodels are members of a uniform or variable speed splitting submodels and require uniform time steps, and "Dyn:" submodels have adaptive time steps.

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.

[**?**] 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 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 [**?**] and many others. (Figure 4.1) 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 [**?**] 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 [**?**] 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.

[**?**] and [**?**] 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 4.1)

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, [**?** **?**], 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.

4.2.2 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.

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.

4.3 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 4.2). 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*

cell 1	*cell 2*	*cell 3*
cell 4	*cell 5*	*cell 6*
cell 7	*cell 8*	*cell 9*

Figure 4.2: 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.

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

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

and

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

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

Algorithm 2 Basic processing pass for fruit

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

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

$NewTreeCount \leftarrow Germination \cdot SeedCount$
$SeedCount \leftarrow SeedCount - (NewTreeCount$
$\quad + Spoilage_{\mathtt{S}} \cdot SeedCount)$
generate $NewTreeCount$ 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]

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

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

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

where $N_{\mathtt{P}}(t,x)$ 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.

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

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

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 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.

IB representations

Individual-based representations for plants, herbivores and carnivores follow the pattern in [**?**]; 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.

4.3.1 *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 ($\mathtt{P}_{M\Omega}$) 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.

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

Algorithm 5 Basic processing pass for plants

```
if (Mass ≥ P_Mature) ∧ (P_Fruits ≥ rnd_0,1) then
    AddFruit(P_ρ Mass^(2/3))
end if
if (Mass ≤ P_MΩ PkMass) ∨ (Ω_indP < rnd_0,1) then
    Die
else
    Mass ← Γ_P(δt, Mass)
    if Mass > PkMass then
        PkMass ← Mass
    end if
end if
```

4.3.2 *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.

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

Algorithm 6 Basic processing pass for herbivores and carnivores

```
if (Ω_indA > rnd_0,1) ∨ (Hungry ≥ A_Ω) then
    DIE
end if
PreyList ← PREYPRESENT_A(Locus, Mass)
if Sated ≥ 0 then
    Mass ← Mass + GROWTH_A(mass, δt)
else if (Hungry ≥ 0) ∧ (len(PreyList) > 0) then
    Sated ← EAT(PreyList, A_EatLimit, mass)
    Hungry ← 0
    ForageCt ← 0
else if (Hungry ≥ 0) ∧ len(PreyList) = 0) then
    FORAGE
    ForageCt ← ForageCt + 1
else if (Hungry ≥ A_moveT) ∨ CROWDED_A then
    MIGRATE_A(Locus)
else
    if (mass ≥ A_RepSize) ∧ (A_RepP ≥ rnd_0,1) then REPRODUCE_A(Locus)
    end if
end if
```

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 $\textsc{PreyPresent}_{\mathtt{H}}$ and $\textsc{Eat}_{\mathtt{H}}$ 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 $\textsc{Eat}_{\mathtt{C}}$ will handle this value appropriately.

4.3.3 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

$$\frac{\langle \overline{\text{mask}}(\boldsymbol{\tau}, \textit{confidence}, 0)\rangle}{\text{supp}(\overline{\text{mask}}(\boldsymbol{\tau}, \textit{confidence}, 0))},$$

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 4.3 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: $\boldsymbol{\tau}_{\mathsf{SD}(0),0}$–$\boldsymbol{\tau}_{\mathsf{SD}(10),0}$, the status tree for the lone carnivore, $\boldsymbol{\tau}_{IB(11),0}$, followed by the trees which represent alternative agents:

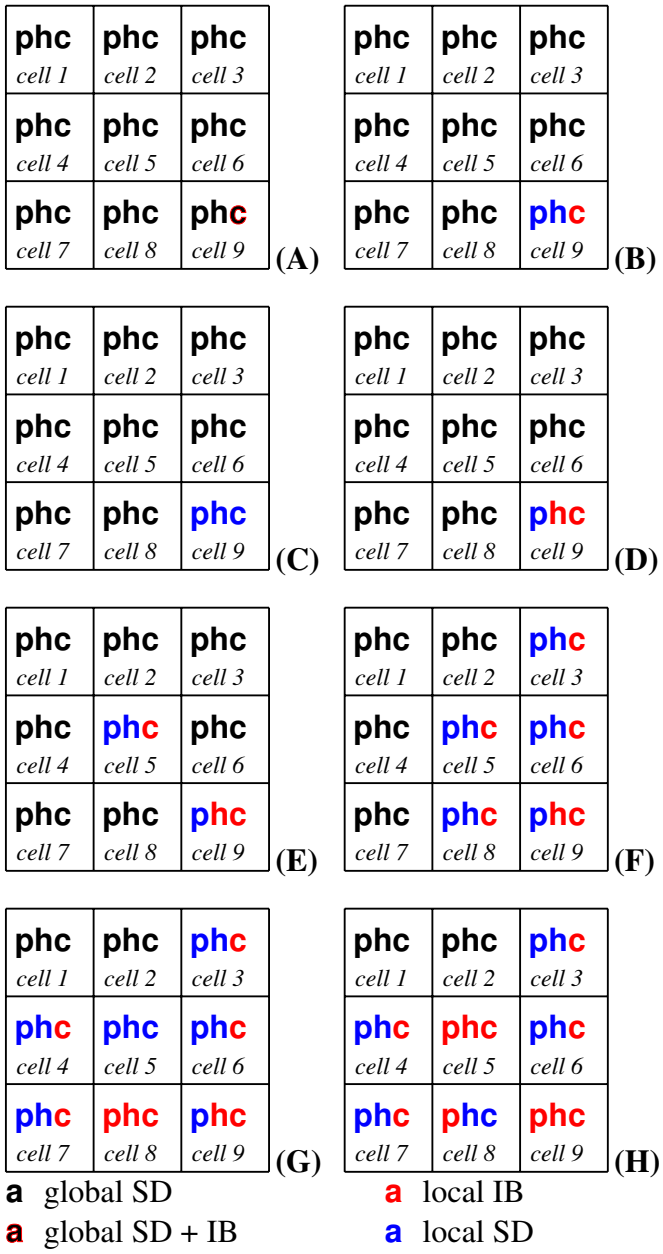

Figure 4.3: The color of the **p**,**h** and **c** indicate an agent's current representation within a cell at various points in the description of a simulation. In each, a black symbol indicates that the biomass of plants (**p**), herbivores (**h**) or carnivores (**c**) is modeled with the global *SD* 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.

$\hat{\boldsymbol{\tau}}_{SD(0),0}$– $\hat{\boldsymbol{\tau}}_{SD(10),0}$ 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

$$\text{ssim}(\boldsymbol{c},\, \boldsymbol{\tau_d}) = \frac{\text{overlap}(\boldsymbol{c},\, \boldsymbol{\tau_d})}{\max(\|c\|_{\mathsf{T}}, \|\tau_d\|_{\mathsf{T}})},$$

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 Δ(,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 $\overline{\text{mask}}$ 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 if $\boldsymbol{S}$ is the set of all serial numbers for candidates, $\check{\boldsymbol{\tau}}_d^{\Sigma}$ is the status tree fo the current domain, and is the , we calculate

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

and this is used to generate

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

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, $\check{\boldsymbol{\tau}}_d^{\Sigma}$, 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 $\boldsymbol{\tau}_0^{\Sigma}$ and $\boldsymbol{\tau}_{SD(9),0}^{\Sigma}$ 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, $\boldsymbol{\tau}_{11(0),}$, $\boldsymbol{\tau}_{IB(11),0}$ and $\hat{\boldsymbol{\tau}}_{IB(11),0}$, indicate that it is confident that it can represent the biomass, and that there are no immediate unmet requirements from other agents. Figure 4.3 (**A**)

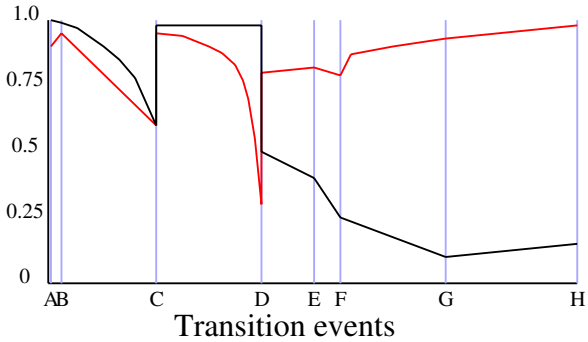

Figure 4.4: Normalized indexes of execution speed (black) and fidelity (red) the against configuration changes through time associated with Figure 4.3

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, $\check{\boldsymbol{\tau}}_9^{\Sigma}$, 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 4.3 (**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 4.4 is a result of a change in representation: the number of carnivores, recorded in $\boldsymbol{\tau}^{\Sigma}_{SD(C),t_4}$, reaches a point that prompts the monitor to convert them to an *SD* form. Figure 4.3 (**C**)

4. The biomass of carnivores has increased significantly by the time the model reaches **D** in Figure 4.4, 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 4.4 around **D** arise from the collapse of the carnivore's prey, followed by the increase in fidelity after the representation change at **D**. Figure 4.3 (**D**)

5. A carnivore, agent 43, has been hungry ($(Hungry \geq \mathtt{A}_{moveT})$ and has migrated to the cell 5 (noted in $\boldsymbol{\tau}_{IB(43),t_5}$). 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 4.3 (**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 $\boldsymbol{\tau}_{IB(n),t_7}$ 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 $\boldsymbol{\tau}_{IB(n),t_7}$ 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 4.3 (**F**)

7. The population of carnivores in cell 9 crashes as a result of migration and the scarcity of prey, (reported in $\check{\boldsymbol{\tau}}_9^{\Sigma}$) The *IB* juvenile herbivores are patchy and harder to find, so only a few carnivores are getting enough to eat. There will be many $\boldsymbol{\tau}_{IB(n),t_{10}}$ 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 $\boldsymbol{\tau}_{SD(4),0}^{\Sigma}$ and $\boldsymbol{\tau}_{SD(7),0}^{\Sigma}$). A fire begins in cell 8, moving through cell 5: biomass loss in all niches causes all niches to shift to *IB* representations. Figure 4.3 (**G**)

8. Juvenile herbivores are reappearing in cell 9, but the available plant biomass (recorded in $\boldsymbol{\tau}_{\mathsf{SD}(9),t_{11}}$) 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 4.4 shows a modest increase in execution speed between **G** and **H** due to the population losses associate with the fire. Figure 4.3 (**H**)

- ...the simulation continues

4.4 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 [**?**]. 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 [**? ? ?**] 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.

4.5 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.

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

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.

BIBLIOGRAPHY

[1] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and Swarztrauber P. N. A standard test set for numerical approximations to the shallow water equations in spherical geometry. *J. Comput. Phys.*, 102:211–224, 1992.

Appendix

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 .0.1. *Let $\mathcal{S}$ be a set of labels, and let $\mathbb{F}$ 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 $v \in \mathbb{F}$, $s \subset \mathcal{S}$, 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 $\mathsf{O} = (\emptyset, 0, \emptyset)$, 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 $\boldsymbol{u} = (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \boldsymbol{u}_{\boldsymbol{E}})$ in $\boldsymbol{T}$ correspond to its label*,* value*, and* extension set*. An element of $\boldsymbol{u}_{\boldsymbol{E}}$ 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 $\boldsymbol{u}_{\mathsf{P}} = \boldsymbol{v}_{\mathsf{P}}$ or at least one of $\boldsymbol{u}$ and $\boldsymbol{v}$ is O.

We define the *depth* of a tree with:

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

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

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

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

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

Simple nodes are the only nodes that have a cardinality of one, and 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.

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

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

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

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

The *overlap* between two trees is defined

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

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 $\|\boldsymbol{u}\|_{\mathsf{T}} = \|\boldsymbol{v}\|_{\mathsf{T}} = \operatorname{overlap}(\boldsymbol{u}, \boldsymbol{v})$.

We can now define scalar multiplication, and tree addition.

Definition .0.2. *Take $a \in \mathbb{F}$ and $\boldsymbol{u} \in \boldsymbol{T}$, then*

$$a\,\boldsymbol{u} = \begin{cases} \mathbf{O} & \textit{if } \boldsymbol{u} = \mathbf{O} \\ (\boldsymbol{u}_{\mathsf{P}}, a\,\boldsymbol{u}_v, \{a\,\boldsymbol{f} : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}\} \setminus \{\mathbf{O}\}) & \textit{otherwise.} \end{cases} \tag{1}$$

Definition .0.3. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be compatible elements of $\boldsymbol{T}$. Then taking the symbol $+$ to be addition in the field $\mathbb{F}$, we extend it to addition in $\boldsymbol{T}$ so that for nodes $\boldsymbol{u}$ and $\boldsymbol{v}$,*

$$\boldsymbol{u} + \boldsymbol{v} = \begin{cases} \mathbf{O} & \textit{if } \boldsymbol{u} = \boldsymbol{v} = \mathbf{O} \\ \boldsymbol{u} & \textit{if } \boldsymbol{u} \neq \mathbf{O} \textit{ and } \boldsymbol{v} = \mathbf{O} \\ \boldsymbol{v} & \textit{if } \boldsymbol{u} = \mathbf{O} \textit{ and } \boldsymbol{v} \neq \mathbf{O} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v + \boldsymbol{v}_v, \emptyset) & \textit{if } \boldsymbol{u}_{\boldsymbol{E}}, \boldsymbol{v}_{\boldsymbol{E}} = \emptyset \\ \Big(\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v + \boldsymbol{v}_v, (\{\boldsymbol{f} + \boldsymbol{g} : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}} \textit{ and } \boldsymbol{g} \in \boldsymbol{v}_{\boldsymbol{E}} \textit{ and } \boldsymbol{f}_{\mathsf{P}} = \boldsymbol{g}_{\mathsf{P}}\} & \\ \qquad \cup \{\boldsymbol{f} : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}} \textit{ and } \boldsymbol{f}_{\mathsf{P}} \neq \boldsymbol{g}_{\mathsf{P}} \forall \boldsymbol{g} \in \boldsymbol{v}\} & \\ \qquad \cup \{\boldsymbol{g} : \boldsymbol{g} \in \boldsymbol{v}_{\boldsymbol{E}} \textit{ and } \boldsymbol{g}_{\mathsf{P}} \neq \boldsymbol{f}_{\mathsf{P}} \forall \boldsymbol{f} \in \boldsymbol{u}\}) \setminus \{\mathbf{O}\}\Big) & \textit{otherwise.} \end{cases} \tag{2}$$

Definition .0.4. *Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be compatible elements of $\boldsymbol{T}$. Then we define inner-multiplication between the two nodes*

$$\boldsymbol{u} \cdot \boldsymbol{v} = (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v\, \boldsymbol{v}_v, \{\boldsymbol{f} \cdot \boldsymbol{g} : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}, \boldsymbol{g} \in \boldsymbol{v}_{\boldsymbol{E}} \textit{ and } \boldsymbol{f}_{\mathsf{P}} = \boldsymbol{g}_{\mathsf{P}}\} \setminus \{\mathbf{O}\}) \tag{3}$$

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

$$\langle \boldsymbol{u} \rangle = \begin{cases} 0 & \textit{if } \boldsymbol{u} = \mathbf{O} \\ |\boldsymbol{u}_v| & \textit{if } \boldsymbol{u}_{\boldsymbol{E}} = \emptyset \\ |\boldsymbol{u}_v| + \sum_{\boldsymbol{e} \in \boldsymbol{u}_{\boldsymbol{E}}} \langle \boldsymbol{e} \rangle & \textit{otherwise.} \end{cases}$$

It is clear that $\langle \boldsymbol{u} \rangle$ will always be non-negative, and the only shortcoming is that we can have a node $\boldsymbol{u}$ with $\langle \boldsymbol{u} \rangle = 0$, but $\boldsymbol{u} \neq \mathbf{O}$. In order to turn this into a normed vector space we take the set $\mathfrak{O} = \{ \boldsymbol{u} : u \in \boldsymbol{T} \text{ and} \langle \boldsymbol{u} \rangle = 0 \}$ and we construct an equivalence relation on $\boldsymbol{T}$ by the rule $[\boldsymbol{u}] \equiv [\boldsymbol{v}]$ if and only if there exist $\boldsymbol{z_u}, \boldsymbol{z_v} \in \mathfrak{O}$ such that $\boldsymbol{u} + \boldsymbol{z_u} = \boldsymbol{v} + \boldsymbol{z_v}$. 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 $\langle \boldsymbol{u} \rangle$ to be a norm and that it induces a distance function

$$\mathrm{d}(\boldsymbol{u}, \boldsymbol{v}) = \langle \boldsymbol{u} - \boldsymbol{v} \rangle$$

.

Definition .0.6. *We define the functions* mask *and* $\overline{\text{mask}}$ *that set the values associated with particular nodes in a tree to $v \in \mathbb{F}$. Specifically, if $\mathcal{L} \subset \boldsymbol{T}$, then*

$$\text{mask}(\boldsymbol{u}, \mathcal{L}, v) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, v, \{\text{mask}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}\}) & \textit{if } \boldsymbol{u}_{\mathsf{P}} \in \mathcal{L} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\text{mask}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}\}) & \textit{otherwise} \end{cases} \tag{4}$$

and

$$\overline{\text{mask}}(\boldsymbol{u}, \mathcal{L}, v) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, v, \{\overline{\text{mask}}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}\}) & \textit{if } \boldsymbol{u}_{\mathsf{P}} \notin \mathcal{L} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\overline{\text{mask}}(\boldsymbol{f}, \mathcal{L}, v) : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}}\}) & \textit{otherwise.} \end{cases} \tag{5}$$

$\text{mask}(\boldsymbol{u}, \mathcal{L}, 0)$ *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 $\boldsymbol{p}$,

$$\text{excise}(\boldsymbol{u}, \boldsymbol{p}) = \begin{cases} (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \{\text{excise}(\boldsymbol{f}, \boldsymbol{g}) : \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}} \textit{ and } \boldsymbol{g} \in \boldsymbol{p} \textit{ and } \boldsymbol{f}_{\mathsf{P}} = \boldsymbol{g}_{\mathsf{P}}\} \setminus \{\mathbf{O}\}) & \textit{if } \boldsymbol{u}_{\mathsf{P}} = \mathcal{L} \\ \mathbf{O} & \textit{if } \boldsymbol{u}_{\mathsf{P}} \neq \boldsymbol{p}_{\mathsf{P}} \\ (\boldsymbol{u}_{\mathsf{P}}, \boldsymbol{u}_v, \emptyset) & \textit{otherwise,} \end{cases} \tag{6}$$

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

$$\text{child}(\boldsymbol{u}, \ell) = \begin{cases} \boldsymbol{f} & \textit{if } \boldsymbol{f}_{\mathsf{P}} = \ell \textit{ and } \boldsymbol{f} \in \boldsymbol{u}_{\boldsymbol{E}} \\ \mathbf{O} & \textit{otherwise.} \end{cases} \tag{7}$$

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

Definition .0.7. *The degree of deviation between two trees,* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *is given by the expression*

$$\Delta(\boldsymbol{u}, \boldsymbol{v}) = \begin{cases} \|u\|_{\mathsf{T}} + \|v\|_{\mathsf{T}} - 2\,\mathrm{overlap}(\boldsymbol{u}, \boldsymbol{v}) + \langle \boldsymbol{u} - \boldsymbol{v} \rangle & \textit{if } \boldsymbol{u} \textit{ and } \boldsymbol{v} \textit{ are compatible} \\ \|u\|_{\mathsf{T}} + \|v\|_{\mathsf{T}} & \textit{otherwise.} \end{cases} \tag{8}$$

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.*