Dealing with high uncertainty in qualitative network models using Boolean analysis

Nadiah P. Kristensen[1,2], Ryan A. Chisholm[2], Eve McDonald-Madden[1,3]

[1]ARC Centre of Excellence for Environmental Decisions, The University of Queensland, St Lucia, Qld, Australia.
[2]Department of Biological Sciences, National University of Singapore, 14 Science Drive 4 Singapore 117543, Singapore.
[3]School of Geography, Planning and Environmental Management, the University of Queensland, St. Lucia Queensland, Australia.

nadiah@nadiah.org
ryan.chis@gmail.com
e.mcdonaldmadden@uq.edu.au

Abstract

1. Models for predicting ecological behaviour typically require large volumes of data for parameterisation, which is a problem because data are scarce. Qualitative modelling (QM) provides an alternative by exploring the entire range of possible parameter values. When a parameter value is completely unknown, QM typically invokes the Principle of Indifference (PoI), for example by sampling the parameter from a uniform prior distribution. However, if PoI is invoked in this probabilistic way, there may be multiple possible methods for defining a parameter space and sampling values from it, and, worryingly, two different but equally defensible methods can lead to different predictions about ecosystem responses.
2. We investigated how probabilistic PoI can give rise to problems in QM, and developed an alternative method based on Boolean PoI that does not suffer the same limitations. We used a case study that involved predicting the responses of multiple species to the suppression of a pest. The unknown model parameters were interaction strengths between species. For the standard probabilistic method, we drew the parameters randomly from uniform (PoI) and other distributions. For our new Boolean PoI method, we instead simply specified the ranges of "possible" parameter values, and developed a Boolean analysis technique to summarise model predictions.
3. As expected, invoking probabilistic QM yielded different predictions (species response probabilities) for different but equally defensible parameterisation and sampling schemes. Sometimes differences were large enough to impact decision making. In contrast, our new Boolean PoI approach simply classifies outcomes (species responses) as certain, possible, or impossible. Encouragingly, some species responses that were not consistently resolved under probabilistic PoI were shown by our method to be in fact governed by simple rules. Our method can also identify key species whose responses determine whole-system outcomes.
4. Our non-probabilistic representation of uncertainty circumvents the philosophical problems in standard implementations of PoI for QM. Our Boolean analysis method summarises results in a way that is interpretable and potentially useful to conservation decision makers. A priority for future research is to increase the efficiency of our Boolean approach to allow it to deal with problems of higher complexity (more interactions).

Key words: belief probabilism, Boolean minimisation, community matrix, ensemble ecosystem modelling, loop analysis, press perturbation, sensitivity matrix, weighted-predictions matrix

1 Introduction

Conservation decision makers face the problem that the management options available to them may lead to unexpected negative outcomes. For example, pest control can trigger mesopredator release (Courchamp et al., 1999), release of a competitor (Ruscoe et al., 2011), or more complex feedbacks between species, leading to even worse outcomes for native species. Predictions of these outcomes are needed for sound decisions to be made.

Complex outcomes can in principle be predicted with dynamical systems models (e.g. Levins, 1974; Courchamp et al., 1999), but in most applications such models are impossible to parameterise

Submitted to: Methods in Ecology and Evolution, 2019

due to a lack of data. Data is often lacking about which species and interactions are present, i.e. structural uncertainty, and the strengths of interactions between species, i.e. interaction-strength uncertainty. Both forms of uncertainty can have profound effects on model predictions (e.g. Novak et al., 2011; Marzloff et al., 2011). Structural uncertainty is typically resolved by expert elicitation, but interaction-strength uncertainty is more problematic. Interaction strengths are often completely unknown, and to quantify them empirically would require perturbing every species' population separately while monitoring every other species (e.g. Schmitz, 1997). This is rarely attempted because of the difficulty involved (e.g. Dambacher et al., 2003, reports only two examples).

Recently, modellers have turned to a suite techniques, which we refer to collectively as Qualitative Modelling (QM), that hold the promise of obtaining predictions from dynamical models even when the data required to parameterise them are lacking (Rochet and Rice, 2009; Rochette et al., 2009; Melbourne-Thomas et al., 2012). QM achieves this by exploring the range of parameter value uncertainty to obtain a range of predictions, then interpreting the predictions qualitatively or probabilistically. For example, if 89% of model simulations in the uncertainty range predict that pest control will increase petrel population size, then that is interpreted as high support for that prediction (Raymond et al., 2011).

Early QM approaches to interaction-strength uncertainty produced deterministic predictions, but they were later extended to produce probabilistic predictions as well. Loop analysis (Levins, 1974; Lane and Levins, 1977; Marzloff et al., 2011) assumes that population dynamics are near equilibrium, and that a press-perturbation can be used to simulate the scenario of interest (e.g. pest control) (Bender et al., 1984; Yodzis, 1988; Nakajima, 1992). Loop analysis can identify species responses that will occur regardless of what the interaction-strength values are, classifying responses as positive, negative, or indeterminate if either response is possible. The weighted-predictions matrix approach builds on loop analysis, by inferring probabilities for the indeterminate species responses (Dambacher et al., 2003; Metcalf et al., 2011; Ramos-Jiliberto et al., 2012; Lassalle et al., 2013; Wildermuth et al., 2018). Weightings in the matrix count the proportion of positive versus negative feedback loops between species, and high weightings are interpreted as evidential support for an indeterminate response taking a particular sign. More recently, Monte Carlo simulations have been used (Raymond et al., 2011; Reum et al., 2015; Harvey et al., 2016; Sobocinski et al., 2018), where the proportions of random parameterisations giving a positive or negative species response are interpreted probabilistically. The probabilistic approach can also address structural uncertainty, by comparing the likelihoods with which candidate network structures reproduce past observations (Hosack et al., 2008; Montaño-Moctezuma et al., 2007; Melbourne-Thomas et al., 2012; Hansen et al., 2017). Conceptually related approaches may include fuzzy cognitive maps (e.g. Baker et al., 2018), ensemble ecosystem modelling (e.g Baker et al., 2017), and Monte Carlo simulations in other types of models (e.g. Lassalle et al., 2013). To varying degrees, these methods express a philosophical idea: ignorance can be expressed in probabilistic terms, and the outcomes of doing so, also expressed as probabilities, quantify one's confidence in a prediction.

When there is complete ignorance of parameter values, QM implicitly invokes the Principle of Indifference (PoI) (Keynes, 1921). PoI encompasses two ideas (Norton, 2008). The first idea (PoI 1) is a truism of evidence: if we have no grounds for distinguishing possible scenarios, then we should favour them equally. The choice to favour scenarios equally can be justified on epistemic grounds, expressing our degree of belief, or objective grounds, expressing the degree of support from evidence. PoI 1 is invoked, for example, when equal weighting is given to each positive and negative feedback loop in the weighted-predictions matrix. The second idea (PoI 2) is that the extent of favouring can be represented by a probability. PoI 2 is invoked, for example, by sampling unknown interaction strengths from a uniform prior, and by interpreting predictions with higher probabilities as having higher support.

The two ideas of PoI work well in canonical examples (e.g. tossing a die), however we encounter philosophical and technical difficulties when background information is lacking (Box 1). The difficulty arises because we must describe a parameter space to represent the equally favoured scenarios, and there are usually multiple ways to do so. In cases of very little background knowledge, we may have no means to privilege one description of the parameter space as the correct one. This can lead to a paradoxical result: different but equally defensible descriptions predict different probabilities for the same outcome (e.g. Bertrand's paradox (reviewed in Shackel, 2007); the squares factory (Smith, 2014)). This general problem with PoI raises the possibility that QM that implicitly invokes PoI may

suffer from similar ambiguity problems, and that this may impact conservation decision-making.

Box 1: Principle of Indifference examples

Example 1: Tossing a die

A die is tossed. We have no reason to favour any one outcome over any other, therefore by PoI 1 we must treat each outcome $1,\ldots,6$ equally. By PoI 2 we assign the same probability, $\frac{1}{6}$, to each. This gives the correct (observed) probabilities.

Example 2: The von Mises wine/water paradox (reviewed in Deakin, 2006)

All that is known about a bottle of liquid is that it contains a mix of wine and water, and that the ratio of wine to water is between 1/3 and 3. What is the probability that the ratio of wine to water is less than or equal to 2? Assigning the same probability to each value is equivalent to sampling from a uniform distribution. If we define $x = \frac{\text{wine}}{\text{water}}$, then $P(x \leq 2) = (2-1/3)/(3-1/3) = 5/8$. However we can reframe the problem in terms of $y = \frac{\text{water}}{\text{wine}}$. Then applying PoI gives $P(y \geq 1/2) = (3-1/2)/(3-1/3) = 15/16$. These are equivalent and equally defensible descriptions of the parameter space, but applying PoI to each of them gives contradictory probabilities.

Example 3: Modelling for conservation decision-making

An expert elicitation workshop is being conducted, to develop a model (similar to Fig. 1) to predict the effects of pest suppression on other species in an ecosystem. Experts have produced an interaction network structure, and conservation decision-makers have agreed to use a population dynamic model and its near-equlibrium assumptions. However, the experts state that no data exists to quantify the interactions between the species. They have no reason to prefer any values over any others, therefore the modellers propose that they will treat all possible values equally.

The modellers consider two approaches: (1) the community matrix approach, where a uniform prior is used to sample interaction-strength values equally; or (2) an approach similar to the weighted-predictions matrix, where each positive and negative feedback loop is weighted equally. Which should they use?

The modellers could themselves choose an approach. Both are equally valid representations of the same model and ways to represent the interactions. However, the modellers know that the different approaches will produce different prediction probabilities, which can be different enough to impact conservation decision-making. The modellers could clarify that the choice is an additional assumption underlying the model predictions. However, the conservation decision-makers find the assumption's meaning obscure: if both are equally valid, then what does it mean to choose one or the other, and how do they account for that in their decision-making?

The modellers could instead ask the experts to choose. However, when the experts said that they had no data about interaction-strength values, that included not knowing which parameter space would be best to represent that lack of data. Further, the conservation decision-makers say that this ambiguity is part of the uncertainty that they need the modelling approach to account for, in order to make their decision.

The modellers decide to present predictions from both approaches separately — and this seems to address the issues above. They have captured experts' uncertainty about both the parameter value and parameter space. Where the predictions agree, the decision-makers feel more confident that they understand the risks of their management interventions; and where they disagree, it suggests that the combined approach is doing a good job of capturing all of the uncertainty present.

However, the exercise has given the modellers pause. They know that there are many additional model representations, also having mathematical equivalence to the representations above, potentially including ones that they are not aware of. Each has a different parameter space and can therefore produce different predictions. Further, it is paradoxical that two equally valid ways of representing the same thing – 'ignorance' – should produce different results (c.f. wine/water paradox); it suggests that information is missing from the problem specification. In simple examples (e.g. die roll), a parameter space can be chosen using background information about the mechanism of randomisation, however the mechanisms determining species interaction strengths are too complex to resolve in this way.

In this paper, we will use a case study involving interaction-strength uncertainty (Raymond et al., 2011) to illustrate how different but equally defensible random parameterisation methods lead to contradictory predictions. We will then turn to the philosophy literature for a non-probabilistic method of expressing uncertainty under PoI, which circumvents the philosophical problems described above. We will demonstrate how Boolean analysis can be used to summarise non-probabilistic model predictions into a form that is useful to conservation decision makers.

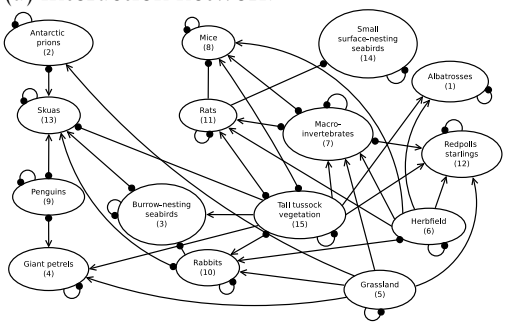
2 Background

2.1 Background to the Macquarie Island case study

Raymond et al. (2011) used a population dynamic model to predict the responses of species on Macquarie Island, Australia, to pest control of rabbits, rats, and mice. They simulated pest control as a negative press perturbation using the sensitivity matrix approach (Bender et al., 1984; Yodzis, 1988; Nakajima, 1992). The sensitivity matrix predicts the change in the population sizes of other species, and the changes can be summarised by their signs (Fig. 1).

In order to obtain predictions of species responses in this model, prior knowledge of the interaction strengths between all species is needed, but because these were not known, Raymond et al. (2011) used Monte Carlo simulations. For each network structure (obtained from expert elicitation, e.g. Fig. 1a), they sampled interaction-strength magnitudes from a uniform distribution and created an ensemble of community matrices (e.g. Fig. 1b). This ensemble was filtered using plausibility constraints: the population dynamics must be locally stable, and the model must reproduce past observations (e.g. suppression of rabbits increases tall tussock vegetation). For each plausible community matrix, they derived a sensitivity matrix (e.g. Fig. 1c) predicting species responses to pest control (e.g. Fig. 1d). Then, for each species, they interpreted the proportion of responses of a particular sign as a probability indicating the level of support for that response occurring in the real system.

(a) interaction network



(b) community matrix A

$$\begin{bmatrix} -a_{1,1} & 0 & 0 & 0 & 0 & \cdots & a_{1,15} \\ 0 & -a_{2,2} & 0 & 0 & a_{2,5} & & 0 \\ 0 & 0 & -a_{3,3} & 0 & 0 & & a_{3,15} \\ 0 & 0 & 0 & -a_{4,4} & a_{4,5} & & a_{4,15} \\ 0 & 0 & 0 & 0 & -a_{5,5} & & 0 \\ \vdots & & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & -a_{15,15} \end{bmatrix}$$

(c) sensitivity matrix $S = (-A)^{-1}$

$$\begin{bmatrix} s_{1,1} & s_{1,2} & s_{1,3} & s_{1,4} & s_{1,5} & \cdots & s_{1,15} \\ s_{2,1} & s_{2,2} & s_{2,3} & s_{2,4} & s_{2,5} & & s_{2,15} \\ s_{3,1} & s_{3,2} & s_{3,3} & s_{3,4} & s_{3,5} & & s_{3,15} \\ s_{4,1} & s_{4,2} & s_{4,3} & s_{4,4} & s_{4,5} & & s_{4,15} \\ s_{5,1} & s_{5,2} & s_{5,3} & s_{5,4} & s_{5,5} & & s_{5,15} \\ \vdots & & & & & \ddots & \vdots \\ s_{15,1} & s_{15,2} & s_{15,3} & s_{15,4} & s_{15,5} & \cdots & s_{15,15} \end{bmatrix}$$

(d) species response

- If $\operatorname{sgn}(s_{1,10}) = +$, then albatrosses predicted to respond negatively to rabbit control;
- If $\operatorname{sgn}(s_{1,10}) = -$, then albatrosses predicted to respond positively to rabbit control;
- If $\operatorname{sgn}(s_{2,10}) = +$, then antarctic prions predicted to respond negatively to rabbit control;
- If $\operatorname{sgn}(s_{2,10}) = -$, then antarctic prions predicted to respond positively to rabbit control;
- ...

Figure 1: General procedure for modelling the effect of a pest control programme on species in an ecosystem. The interaction network describes the direct interactions between species, where positive effects are indicated by an arrow and negative effects by a filled circle (a). In the Macquarie Island case study, interaction networks were obtained using expert elicitation. The corresponding community matrix has elements that quantify the strengths of the interactions between species (b). In the case study, the interaction-strengths were unknown, so numerical values were randomly sampled. Pest control is modelled as a negative press perturbation on the pest species, therefore the sensitivity matrix, which is calculated as the inverse of the community matrix, determines the response of species to pest control (c). Species responses to pest control are summarised by their sign i.e. 'positive' or 'negative' (d).

2.2 Background to the Boolean approach

The problems of PoI cannot be resolved by discarding PoI 1 (equal favouring of indistinguishable scenarios) because it is a truism, therefore the solution must be found by relinquishing PoI 2 and using a non-probabilistic representation of favouring instead (Norton, 2008). When the full continuous-valued probability framework is replaced with a three-valued system, such that scenarios are classified as simply (1) possible, (2) necessary, or (3) impossible, then the PoI paradoxes no longer occur (Norton, 2010a,b). The next challenge is to turn this impoverished description into useful outputs for decision makers.

When the case-study's model inputs — the interaction strengths — are unknown, our non-probabilistic approach will treat every input value as possible; however, this does not mean that every output — every species-response combination (e.g. albatrosses respond positively, prions respond negatively, etc.) — is possible. Some combinations will be impossible, regardless of what the interaction-strength values are, because of the structure of the interaction network. The classic example is a food chain: a negative press-perturbation of the apex predator always has a positive effect on its prey, which in turn has a negative effect on that prey's prey, and so on with alternating signs down the chain (Usmani, 1994); therefore it is impossible for a predator–prey pair to have the same response sign. More complex interaction networks will have more complex rules, but the principle remains the same: the network structure determines the ecological feedbacks between species, which makes certain species-response combinations impossible. Plausibility constraints, e.g. representing past observations, will further increase the set of impossible combinations.

Our approach begins with a truth table (e.g. Fig. 2a), which lists every species-response combination that is combinatorially possible. We then classify each as possible in the model if it can arise under at least one parameterisation, or impossible otherwise. As the number of species responses grows, the truth table quickly becomes very long (n responses gives 2^n rows) and difficult to interpret. A method for summarising the truth table is needed.

(a) truth table

Negative perturbation of spp 3 causes positive response in:			Species-response combination impossible in the model?
species 3	species 4	species 5	
False	False	False	True
False	False	True	True
False	True	False	True
False	True	True	False
True	False	False	False
True	False	True	True
True	True	False	True
True	True	True	True

(b) minimised truth table

Negative perturbation of spp 3 causes positive response in:			Species-response combination impossible in the model?	
species 3	species 4	species 5		
False	-	False	True	*i*
-	False	True	True	*j*
True	True	-	True	*k*

(c) logical relationships between species-responses

i: if spp 3 responds negatively then spp 5 responds positively
if spp 5 responds negatively then spp 3 responds positively

j: if spp 4 responds negatively then spp 5 responds negatively
if spp 5 responds positively then spp 4 responds positively

k: if spp 3 responds positively then spp 4 responds negatively
if spp 4 responds positively then spp 3 responds negatively

(d) logical implication network

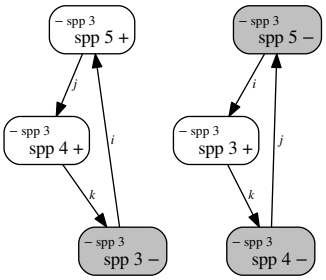


Figure 2: An example of Boolean analysis, using the results from model simulations of three species' responses to a negative press perturbation of 'species 3' (details in SI A). Every species-response combination that is combinatorially possible was classified as either possible or impossible in the model and encoded in a truth table (a). Boolean minimisation of the truth table produces its minimal representation, where '-' indicates species responses that can take either truth value (b). Each row of the minimised truth table corresponds to a set of logical implication statements (c). Statements are chained together to create a logical implication network, which summarises the deterministic relationships between species responses as predicted by the model (d). The nodes represent species responses, and the arrows between nodes represent the direction of the logical implication, e.g. 'spp 3 − → spp 5 +' reads 'if species 3 responds negatively then species 5 responds positively'. The small '− spp 3' in each node indicates that species are responding to a negative press-perturbation of species 3.

The truth table can be summarised using Boolean analysis, which is a technique for analysing logical relationships between dichotomous variables (Theuns, 1994). It is used in sociology for analysis of questionnaire data, to determine if a 'yes' or 'no' answer to a question is always predicted by some combination of 'yes' or 'no' answers to another set of questions (e.g. Degenne and Lebeaux, 1996). In the context of pest control, this is analogous to determining if a positive or negative response in a species is always predicted by some combination of positive or negative responses in another set of species.

The first step of Boolean analysis is Boolean minimisation, which reduces the truth table to the minimal table needed to provide its complete description (e.g Fig. 2b). Boolean minimisation is commonly used in circuit design, therefore algorithms are readily available to find a solution that is minimal (e.g. McCluskey, 1956) or close-to-minimal (e.g. Brayton et al., 1984). The second step is to use each row of the minimised table to derive a set of logical implication statements (e.g Fig. 2c), which are deterministic rules governing the relationships between species responses. The final step is to connect implications together to construct a logical implication network (e.g Fig. 2d). It can be shown (Theuns, 1994) that if at least one statement from every row of the minimised table is included, then the network gives a complete description of all species-response relationships in the model.

See SI C for tutorials on implementing the Boolean approach, and SI D for guidance on constructing and interpreting logical implication networks.

3 Materials and methods

We analysed the predictions of the Macquarie Island model using two alternative methods: (1) Monte Carlo simulations (a standard QM approach expressing probabilistic PoI), which were used to test the effect of sampling method on the predictions, and (2) our new Boolean approach. We tested the ability of both approaches to handle interaction-strength uncertainty only: the structure of the ecological interaction network was fixed as shown in Fig. 1a. We predicted species responses to rabbit control, using the sensitivity matrix approach, subject to three plausibility constraints: (1) the system is stable, (2) rabbits respond negatively to rabbit control, and (3) tall tussock vegetation responds positively to rabbit control.

3.1 Monte Carlo simulations

The sampling method from Raymond et al. (2011) was used as a baseline for comparisons, and we evaluated the robustness of its predictions to three variations in the sampling method (Table 1). Tests 1 and 2 represent alternative but arguably equally defensible ways to sample interaction-strength values: test 1 is a reparameterisation of the method used in Raymond et al. (2011) and test 2 incorporates additional background information. Test 3 explores the effect of the sampling method on the stability constraint. Summary results were calculated from 500 plausible community matrices for tests 1 and 2, and 100 matrices per subsample for test 3.

3.1.1 Test 1: Sampling in the Lotka-Volterra parameter space

Raymond et al. (2011) performed the Monte Carlo sampling on the elements of the community matrix. However, the community matrix represents the linearisation of some population dynamic model around its steady state, and is therefore a description of the parameter space in terms of derivative properties (SI B). When the parameter spaces have a physical meaning, a philosophical argument can be made that PoI should be applied to the primary description (Mikkelson, 2004). This raises Question 1: *do the species-response predictions change when the parameter space sampled is that of the primary description?*

To test this, we assumed that the community matrix is derived from the Lotka–Volterra model, and we performed the Monte Carlo simulations on the values of this model's parameters. The population dynamics are

$$\frac{dn_i}{dt} = n_i\left(r_i + \sum_{j=1}^{n} b_{ij}n_j\right), \tag{1}$$

name of test	description of parameter space sampling method
Raymond baseline	Baseline method from Raymond et al. (2011). · $a_{ii} \sim \mathcal{U}(-1,-0.25)\ \forall i$. · $\lvert a_{ij}\rvert \sim \mathcal{U}(0.01,1)\ \forall i \neq j$.
test 1	Based on recommendation of Mikkelson (2004), which applies the Principle of Indifference to the Lotka–Volterra parameter space. · $b_{ii} \sim \mathcal{U}(-1,0)\ \forall i$. · $\lvert b_{ij}\rvert \sim \mathcal{U}(0,1)\ \forall i \neq j$. · $r_i \sim \mathcal{U}(-1,1)$. · Additional plausibility constraint: $n_i^* > 0\ \forall i$.
test 2	Invokes background information that the interaction-strength distribution is skewed (e.g Neutel et al., 2002). · $a_{ii} \sim \mathcal{U}(-1,0)\ \forall i$. · If species i is a prey and species j is a predator: · $a_{ij} \sim \mathcal{U}(-1,0)$, and · $a_{ji} \sim \text{Beta}(\alpha,1)$ such that $\mathbb{E}[a_{ji}] = 10^{-2}a_{ij}$, otherwise: $\lvert a_{ij}\rvert \sim \mathcal{U}(0,1)$.
test 3	Explores the effect of 'strengthening' the stability constraint. · If i is basal: $a_{ii} \sim \mathcal{U}(-1,0)$, otherwise: $a_{ii} \sim -\text{Beta}(1,\beta)$, and explore effect of $\mathbb{E}[a_{ii}]$. · $\lvert a_{ij}\rvert \sim \mathcal{U}(0,1)\ \forall i \neq j$.

Table 1: The different Monte Carlo sampling methods that were applied to the Macquarie Island case study, to test the robustness of predictions. All tests included the following plausibility constraints: A is stable, $-S_{\text{rabbits,rabbits}} < 0$, $-S_{\text{tussock,rabbits}} > 0$. The parameters a_{ij} are non-zero elements of the community matrix, and b_{ij} and r_i are parameters from the equivalent Lotka-Volterra model.

where n_i is the population density of species i, r_i is the intrinsic growth rate of species i, and b_{ij} is the effect that species j has on species i. The parameter values sampled were r_i and b_{ij}. The timescale t in Eq. 1 can be chosen such that all $|r_i| \leq 1$ and $|b_{ij}| \leq 1$. Sampling on these parameters can produce a system with negative population density at steady state i.e. $n_i^* < 0$. Therefore we added the plausibility constraint $n_i^* > 0\ \forall i$.

3.1.2 Test 2: Sampling from a skewed interaction-strength distribution

Raymond et al. (2011) sampled interaction strengths from a uniform distribution. However, detailed studies of real food webs show that interaction-strength distributions are not random and uniform, but rather have skewed distributions with many weak interactions (Wootton and Emmerson, 2005). This is often attributed to the size difference between between predators and prey and feeding inefficiencies (Williams and Martinez, 2000; Emmerson and Raffaelli, 2004; Loeuille and Loreau, 2005). Background information like this is typically incorporated into the sampling distribution (Kass and Wasserman, 1996), and previous QM studies have also sampled interaction strengths from a skewed distribution (e.g. Hosack et al., 2008). This raises Question 2: *do the species-response predictions change when interaction strengths are sampled from a skewed distribution?*

To test this, we sampled the predator-to-prey interaction-strength magnitudes from a standard uniform distribution. Then prey-to-predator magnitudes were sampled from a beta distribution, with a mean scaled to the predator-to-prey magnitude to reflect the heuristic that the former is approximately two orders of magnitude larger than the latter (Neutel et al., 2002).

3.1.3 Test 3: Exploring the effect of stability strength

Raymond et al. (2011) used the stability of the community matrix as a plausibility constraint. This is justified by studies showing that webs have an idiosyncratic structure peculiarly suited to promoting stability (Yodzis, 1981; De Ruiter et al., 1995). However, the 'strength' of the stability constraint can be manipulated by changing the magnitude of the diagonal elements of the community matrix (representing species' self-limitation) relative to the off-diagonal elements. Negative diagonal elements stabilise the system; an unstable system can be converted to a stable one by arbitrarily increasing the magnitude of negative diagonal elements (c.f. Neutel et al., 2002; James et al., 2015). Raymond et al. (2011) (and also Harvey et al., 2016) sampled the off-diagonal elements from $|a_{ij}| \sim \mathcal{U}(0.01,1)$, but

the diagonal elements were sampled from higher magnitudes $a_{ii} \sim \mathcal{U}(-1, -0.25)$. This raises Question 3: *do the species-response predictions change if the magnitude of the diagonal elements, and hence the 'strength' of the stability constraint, is changed?*

To test this, we sampled the consumer species' diagonal elements from a beta distribution, and explored the effect of increasing the strength of the stability constraint by decreasing that distribution's expected value. We chose the consumer species because they generally have low self-regulation compared to basal species and compared to other interaction strengths generally (by roughly one order of magnitude, see Christianou and Kokkoris, 2008; Emmerson and Raffaelli, 2004).

3.2 Boolean approach

To construct the truth table, we performed a parameter-value sweep of the same Macquarie Island model that was used in the Monte Carlo simulations above. We performed the parameter-value sweep on the magnitudes of the interaction-strengths, by sampling $|a_{ij}| \sim \mathcal{U}(0, 1)$, to create a large sample of possible community matrices. We tested each matrix against the three plausibility constraints, and if the matrix passed, we used the sensitivity matrix to predict species responses to rabbit control. Every species-response combination that was observed at least once was identified as possible, and we assumed that combinations that were never seen were impossible. We sampled a total of 1×10^5 community matrices, and no new species-response combinations were found after the first 763 samples. This suggests that we found every combination that was possible, but it does not guarantee it (see Discussion).

To minimise the truth table, we used the the Espresso algorithm implemented in the PyEDA 0.28 package (Drake, 2013–) in Python 3 (see *Data accessibility*). Every logical implication statement with a single antecedent was included in the logical implication network.

Note that the distribution we used to perform the parameter-value sweep does not represent a prior, and the fact that it is uniform is not an expression of PoI. The uniform distribution was chosen for convenience only. Provided that every possible species-response combination is found, then the method used to find them does not influence the Boolean analysis.

4 Results

Alternative Monte Carlo sampling methods yielded quite different predictions for the same species response (selected examples in Fig. 3, other species in SI E). For example, the proportion of negative responses in penguins decreased from 81% when the elements of the community matrix were sampled uniformly (test 3 with $\mathbb{E}[a_{ii}] = 0.5$), to 70% in the reimplementation of Raymond et al. (2011) sampling (Raymond baseline), to 64% when the Lotka–Volterra coefficients were sampled instead (test 1). And while under the baseline parameterisation both skuas and Antarctic prions had ambiguous responses, under parameterisation via the skewed distribution (test 2) there was high evidential support for a negative response by skuas and a positive response by prions. When the stability constraint was 'strengthened' (test 3), the proportion of matrices rejected increased as expected (Fig. E.13), and some species' responses went from having high to ambiguous support (e.g. redpolls starlings from 90% to 46%), while others went from having ambiguous to high support (e.g. skuas from 42% to 94%).

The Boolean approach produced a minimised truth table with 12 rows, from which the logical implication network in Fig. 4 was derived. Herbfield, albatrosses, and burrow-nesting seabirds respond positively to rabbit control (Fig. 4a), consistent with the 100% probabilities for those responses obtained in Monte Carlo simulations (SI E). For some responses where Monte Carlo results were conflicting, the Boolean approach elucidated simple rules: penguin response is partially determined, such that if skuas respond positively then penguins respond negatively (Fig. 4c); and the response of skuas is always opposite in sign to that of prions (Fig. 4d).

Table 2 notes the computational time for both approaches.

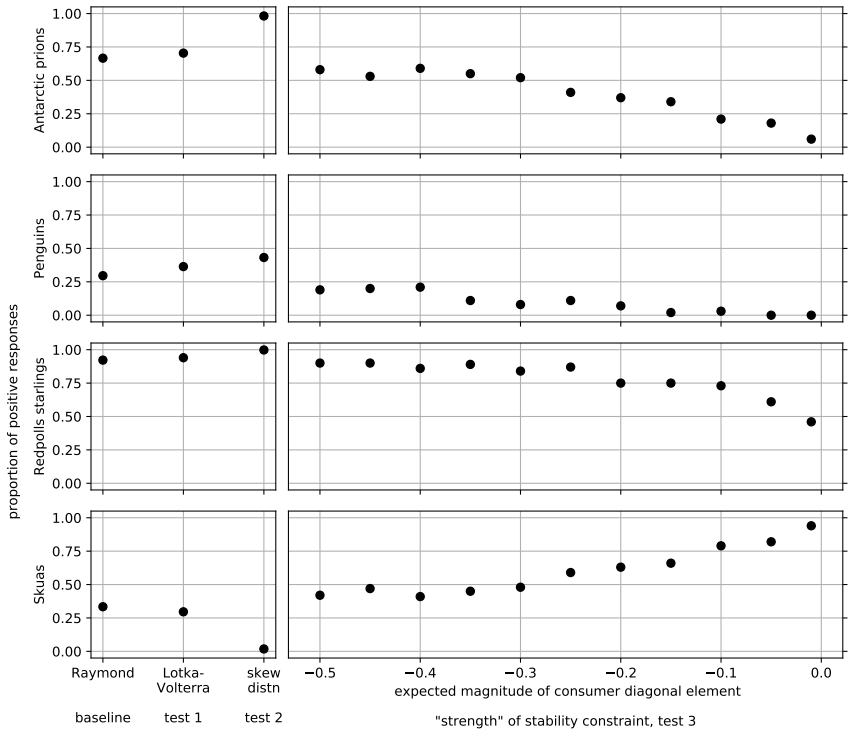


Figure 3: Selected results for Macquarie Island from the different Monte Carlo simulation methods in Table 1 (probabilistic QM). Predictions for some species differed between methods and was sensitive to the strength of the stability constraint. Additional results in SI E.

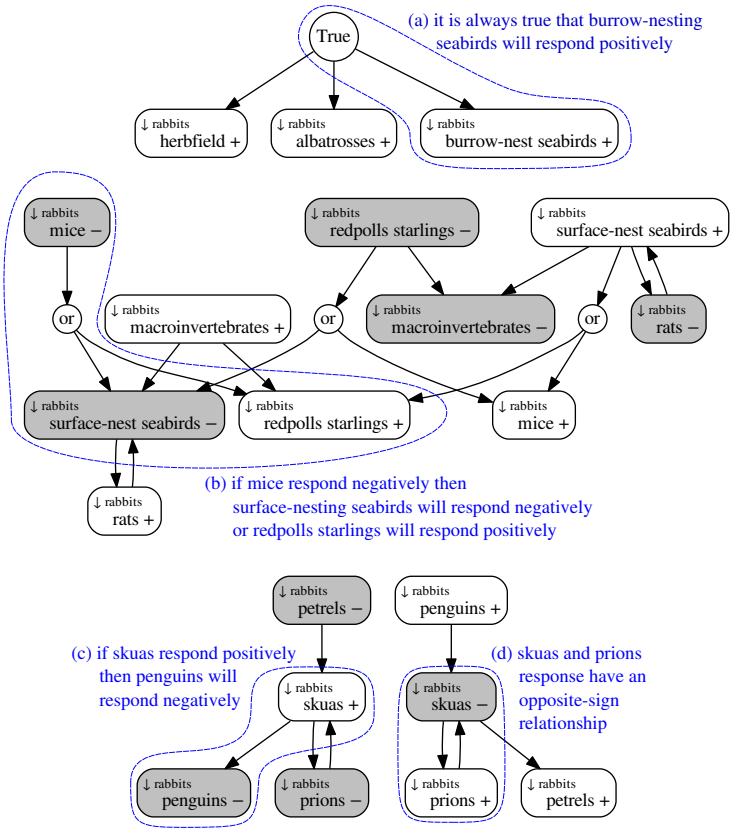


Figure 4: Logical implication network for Macquarie Island from the Boolean approach (non-probabilistic PoI). Species whose responses were ambiguous according to the Monte Carlo simulations or differed between sampling methods (Fig. 3) are governed by simple deterministic rules. See SI D for more guidance interpreting network.

Table 2: A comparison ranking various aspects of four QM methods' performance for modelling with interaction-strength uncertainty (impossible < worst < poor < good < best), and computation times at key algorithmic bottlenecks from the Macquarie Island case study. Computation times for Lenovo ThinkPad X1 Carbon with Intel i7 CPU (2 GHz) and 8 GB RAM.

Aspect	Loop analysis (e.g. Lane and Levins, 1977)	Weighted-predictions matrix (e.g. Dambacher et al., 2003)	Monte Carlo simulations (e.g. Raymond et al., 2011)	Boolean approach (this paper)
Large models and computational time constraints?	*worst to impossible* Requires symbolic calculation of matrix adjugate plus direct analysis of result.	*poor to *impossible* Matrix permanent cannot be calculated for large problems.	*best* Approximately 1000 plausible-matrix samples are sufficient to estimate expected value.	*poor to *impossible* Cannot perform Boolean minimisation for large problems. If using large parameter-value sweep, then cannot know when all response combinations found, and problem is worse for larger models.
Use past responses as plausibility constraints?	*good* Can incorporate constraints into direct analysis.	*impossible* Cannot use.	*best* Can easily filter matrices sampled against constraints.	*best* Can easily filter matrices accepted against constraints.
If high uncertainty, no data available?	*best* No PoI problems. Usefulness of predictions depends upon context and analytic tractability.	*poor* PoI problems for indeterminate responses. Single-species predictions only.	*worst* PoI problems (see text).	*best* No PoI problems. Multi-species relationships predicted.
If low uncertainty, data available?	*poor* Relative constraints (e.g. $a_{ij} > a_{kl}$) easily incorporated, but only prior distribution's support can be used.	*impossible* Neither relative constraints nor prior distribution can be used.	*best* Relative constraints easily incorporated, and natural Bayesian (prior-posterior) interpretation.	*poor* Relative constraints easily incorporated, but only prior distribution's support can be used.
Computation times	NA	*T matrix: 14 s	1×10^3 matrices: 0.6 s	*1×10^5 matrices: 1 min; *Minimisation: 1 s

*Computation time grows exponentially with size of model (i.e. matrix permanent (Valiant, 1979) and Boolean minimisation (Buchfuhrer and Umans, 2011) are NP hard problems).

5 Discussion

Qualitative Modelling (QM) can be used to predict ecosystem behaviour even when species interaction strength data are lacking. However, philosophical and technical difficulties can arise if QM invokes the probabilistic form of the Principle of Indifference (PoI). To illustrate this, we used a case study from Macquarie Island, with the goal of predicting species responses to pest control. We found that different but equally defensible ways of representing interaction-strength uncertainty can produce contradictory predictions. We drew from the philosophy literature to develop a non-probabilistic approach that eliminated the cause of these contradictions. We used Boolean analysis to reveal simple rules governing the model predictions that were not apparent from probabilistic QM.

Under a probabilistic PoI, alternative ways of specifying a QM problem can give rise to contradictory predictions, with differences that can be large enough to impact conservation decision-making. We demonstrated this in the Macquarie Island case study, where some species' responses to pest control were predicted with strong support by one Monte Carlo sampling method but were ambiguous according to another (Fig. 3). We also found that the 'strength' of the stability constraint, which is necessary to capture the idiosyncratic interaction-strength patterns in real webs (e.g. De Ruiter et al., 1995), changed predictions (c.f. Section 2 of Kirk et al., 2015). The point of creating alternative sam-

pling methods was not to criticise the method that Raymond et al. (2011) chose; we do not believe that their choice was unreasonable. Rather, the point is that — under high uncertainty — many different yet arguably equally defensible sampling methods can be devised, and they can lead to very different predictions. Previous authors have argued that modelling uncertainty using a distribution is paradoxical because more information is needed to describe a distribution than an expected value (Rochet and Rice, 2009). Here we have shown how trying to specify some 'neutral' distribution, expressing ignorance of the parameter values, can lead to paradoxical results (c.f. Kraak et al., 2010).

It is not surprising that different sampling methods will produce different predictions, and under a Bayesian framework, these differences have a natural prior–posterior interpretation. This interpretation is useful for certain research questions. For example, May (1972) found that complex food-web models with randomly-sampled interaction strengths have a low probability of being stable, which prompted the wide-ranging complexity-stability debate, where the probabilistic approach remains a key tool (Landi et al., 2018). However, in the context of conservation decision-making, the goal is to predict the behaviour of a particular system and in a way that accounts for experts' uncertainty. When experts' uncertainty is very high, then the best choice of parameter space and sampling distribution can also be uncertain; this aspect of uncertainty is what our Boolean approach addresses (Example 3 in Box 1).

In contrast to probabilistic QM, the Boolean approach produces deterministic predictions that are always true in the model, regardless of the parameter values. By invoking PoI in a non-probabilistic manner, our method generates robust predictions that subsume the contradictory predictions produced by different probabilistic methods. In the case study, different Monte Carlo sampling methods gave different predictions for Antarctic prions and skuas, but they always predicted that the percentage of positive responses in Antarctic prions was 100% minus the percentage of positive responses in skuas. Our Boolean approach revealed why: the response of prions is always opposite in sign to that of skuas. As another example, the response of penguins was ambiguous and depended upon the parameter space sampled. The Boolean analysis revealed more specifically that penguins' response is partly determined: if skuas respond positively then penguins will respond negatively.

The Boolean approach has pragmatic and philosophical advantages over probabilistic QM. One pragmatic advantage is that a deterministic prediction may be easier to interpret, particularly when the probabilistic prediction is ambiguous. Simple relationships between species responses may also be used to identify indicator species for a post-intervention monitoring, or as candidates for additional intervention. For example, the determining effect of the rat response upon surface-nesting seabirds makes rats a good candidate as an indicator species. Alternatively, the positive effect of decreasing rats may inspire additional Boolean analysis, to investigate the outcome of a rat control programme (SI C.3). The Boolean approach can also identify key species that determine the overall system outcomes. In the theoretical example (Fig. 2), species responses that were maximally ambiguous according to a weighted-predictions matrix analysis (SI A) were shown by the Boolean approach to constitute two separate regimes of system behaviour. In the Macquarie Island case study, clusters of species that are linked in the implication network indicate those that respond together. The Boolean approach can also produce results that are difficult to interpret, when there are few rules or all rules are long (SI D); however, this result useful in its own right because it indicates that the system is intrinsically difficult to predict.

The philosophical advantage of the Boolean approach is that it addresses the root cause of contradictory predictions in probabilistic QM. Contradictory predictions occur for the same reason as in philosophical thought-experiments: there is insufficient background information about the system. The missing background information is not merely ignorance of the parameter values, which PoI can work well to address, but rather a more fundamental ignorance about the nature of the system itself. PoI works well when the mechanism of randomisation is well understood, e.g. the toss of a die implies randomisation over the parameter space of the 6 faces. However, the mechanism that determines interaction-strength values involves an interplay between physical, biological, ecological, and evolutionary processes, and these processes are poorly understood. Probabilistic QM forces us to specify the parameter space and randomisation anyway — in order to proceed with the modelling exercise — whereas the Boolean approach provides a way to produce predictions without implicitly overstating our knowledge.

The idea of obtaining deterministic predictions from community matrix models is not new. Loop analysis and the weighted-predictions matrix can identify if a species' response is guaranteed to be

positive or negative or if it is indeterminate (i.e. both signs possible). However, these methods typically consider each species' deterministic response independently from other species. When species are considered independently, even for modestly large networks, most responses will be indeterminate (this motivated the probabilistic interpretation (Dambacher et al., 2002)). In contrast, our Boolean approach considers the interdependencies between species responses, which uncovers new deeper levels of determinacy (e.g. SI A).

The Boolean approach can only tell us about the predictions of a model, and we have not explored model validation. For example, we did not consider variations on the interaction network structure (i.e. variations on Fig. 1a), which is an area for future work. We also treated the model assumptions as given; the Boolean approach cannot tell us if the 'near-equilibrium' assumptions (Abrams, 2001; Justus, 2006) are sufficient for modelling pest control. However, the separation between Boolean analysis and the model means that it can be applied to any model or system where the response variables have a natural dichotomisation, including future improvements to pest control models, and also other kinds of models (e.g. social-ecological, SI C.4).

Finally, we highlight two caveats to the Boolean approach: (1) the assumption that the parameter-value sweep uncovered *all* possible response combinations for input to the Boolean analysis, and (2) the fact that Boolean minimisation is itself computationally expensive. Both caveats mean that, for sufficiently large models, a Boolean approach will be impossible (Table 2).

The first caveat arises because our Boolean analysis assumes that we sampled the parameter space sufficiently to obtain all possible response combinations. In our case study, we strongly suspect that we achieved this because the number of samples was two orders of magnitude larger than was needed to find the final new combination. However, when the parameter space is continuous, then the number of possible parameter values is infinite, therefore there is no guarantee that a naive parameter-sweep will find all possible combinations. It is possible, for certain kinds of problems, to identify the finite number of points at which the parameter values must be sampled to obtain all response combinations (e.g. Sontag, 2014; Giordano et al., 2016). Perhaps these methods can be extended to more general cases in the future. In the meantime, however, because this problem gets worse the larger and more connected the interaction network, it remains a limitation of our approach and may prevent its use for complex models.

Second, Boolean minimisation is an NP hard problem; roughly speaking, this means that the computation time required increases exponentially with the size of the problem. The practical consequence is that it is not possible to minimise the full truth table and obtain the logical implication network when the number of responses of interest is 'large'. To give an example of what can be considered 'large', minimising 16 species responses in a Christmas Island food web (unpublished data) was the approximate computational limit. Further, in multi-species pest control, the effects of each pest's suppression on a species cannot be summed (SI F) but must instead be treated separately, and therefore the problem size doubles with each additional pest suppressed. This limitation is intrinsic to Boolean minimisation.

We are faced with an urgent biodiversity crisis, and predicting the complex ecosystem behaviour to stem further species losses is difficult when data are lacking. The methods we have examined were a constructive response to this pressing need, and it was only by merit of their transparency and theoretically rigorous basis that our work was possible. We hope that highlighting their potential problems will not be discouraging, but instead inspire further technical development to meet the important conservation challenges ahead.

6 Authors' contributions

EMM conceived problem and acquired funding; EMM and RAC contributed to writing; NPK conceived Boolean approach, analysed all models, and led writing. All authors gave final approval for publication.

7 Acknowledgements

The authors thank Sam Thompson and Matthew Luskin for comments on the manuscript, Jeff M. Mikkelson and Alan Hájek for discussions, and John D. Norton for his help to understand his work

and the philosophical problem. We acknowledge funding and support from Research Computing Centre (RCC), University of Queensland. EMM was funded by an ARC future fellowship.

8 Data accessibility

Code and tutorials are archived at DOI: 10.5281/zenodo.2574324

References

Abrams, P. A. (2001). Describing and quantifying interspecific interactions: a commentary on recent approaches, *Oikos* **94**(2): 209–218.

Baker, C. M., Gordon, A. and Bode, M. (2017). Ensemble ecosystem modeling for predicting ecosystem response to predator reintroduction, *Conservation Biology* **31**(2): 376–384.

Baker, C. M., Holden, M. H., Plein, M., McCarthy, M. A. and Possingham, H. P. (2018). Informing network management using fuzzy cognitive maps, *Biological Conservation* **224**: 122–128.

Bender, E. A., Case, T. J. and Gilpin, M. E. (1984). Perturbation experiments in community ecology: theory and practice, *Ecology* **65**(1): 1–13.

Brayton, R. K., Hachtel, G. D., McMullen, C. T. and Sangiovanni-Vincentelli, A. L. (1984). *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, MA.

Buchfuhrer, D. and Umans, C. (2011). The complexity of boolean formula minimization, *Journal of Computer and System Sciences* **77**(1): 142–153.

Christianou, M. and Kokkoris, G. D. (2008). Complexity does not affect stability in feasible model communities, *Journal of Theoretical Biology* **253**(1): 162–169.

Courchamp, F., Langlais, M. and Sugihara, G. (1999). Cats protecting birds: modelling the mesopredator release effect, *Journal of Animal Ecology* **68**(2): 282–292.

Dambacher, J. M., Li, H. W. and Rossignol, P. A. (2002). Relevance of community structure in assessing indeterminacy of ecological predictions, *Ecology* **83**(5): 1372–1385.

Dambacher, J. M., Li, H. W. and Rossignol, P. A. (2003). Qualitative predictions in model ecosystems, *Ecological Modelling* **161**(1): 79–93.

De Ruiter, P. C., Neutel, A.-M. and Moore, J. C. (1995). Energetics, patterns of interaction strengths, and stability in real ecosystems, *Science* **269**(5228): 1257–1257.

Deakin, M. A. (2006). The wine/water paradox: background, provenance and proposed resolutions, *Australian Mathematical Society Gazette* **33**(3): 200–205.

Degenne, A. and Lebeaux, M.-O. (1996). Boolean analysis of questionnaire data, *Social Networks* **18**(3): 231–245.

Drake, C. (2013–). PyEDA: a Python library for electronic design automation.
URL: *https://github.com/cjdrake/pyeda*

Emmerson, M. C. and Raffaelli, D. (2004). Predator–prey body size, interaction strength and the stability of a real food web, *Journal of Animal Ecology* **73**(3): 399–409.

Giordano, G., Samaniego, C. C., Franco, E. and Blanchini, F. (2016). Computing the structural influence matrix for biological systems, *Journal of Mathematical Biology* **72**(7): 1927–1958.

Hansen, G. J., Tunney, T. D., Winslow, L. A. and Vander Zanden, M. J. (2017). Whole-lake invasive crayfish removal and qualitative modeling reveal habitat-specific food web topology, *Ecosphere* **8**(2).

Harvey, C. J., Reum, J. C., Poe, M. R., Williams, G. D. and Kim, S. J. (2016). Using conceptual models and qualitative network models to advance integrative assessments of marine ecosystems, *Coastal Management* **44**(5): 486–503.

Hosack, G. R., Hayes, K. R. and Dambacher, J. M. (2008). Assessing model structure uncertainty through an analysis of system feedback and Bayesian networks, *Ecological Applications* **18**(4): 1070–1082.

James, A., Plank, M. J., Rossberg, A. G., Beecham, J., Emmerson, M. and Pitchford, J. W. (2015). Constructing random matrices to represent real ecosystems, *The American Naturalist* **185**(5): 680–692.

Justus, J. (2006). Loop analysis and qualitative modeling: limitations and merits, *Biology and Philosophy* **21**(5): 647–666.

Kass, R. E. and Wasserman, L. (1996). The selection of prior distributions by formal rules, *Journal of the American Statistical Association* **91**(435): 1343–1370.

Keynes, J. M. (1921). *A treatise on probability*, MacMillan and Co. Ltd., London.
URL: *https://www.gutenberg.org/files/32625/32625-pdf.pdf*

Kirk, P., Rolando, D. M. Y., MacLean, A. L. and Stumpf, M. P. H. (2015). Conditional random matrix ensembles and the stability of dynamical systems, *New Journal of Physics* **17**(8): 083025.
URL: *http://stacks.iop.org/1367-2630/17/i=8/a=083025*

Kraak, S. B., Kelly, C. J., Codling, E. A. and Rogan, E. (2010). On scientists' discomfort in fisheries advisory science: the example of simulation-based fisheries management-strategy evaluations, *Fish and Fisheries* **11**(2): 119–132.

Landi, P., Minoarivelo, H. O., Brännström, Å., Hui, C. and Dieckmann, U. (2018). Complexity and stability of ecological networks: a review of the theory, *Population Ecology* **60**(4): 319–345.

Lane, P. and Levins, R. (1977). The dynamics of aquatic systems. 2. The effects of nutrient enrichment on model plankton communities, *Limnology and Oceanography* **22**(3): 454–471.

Lassalle, G., Nelva Pasqual, J.-S., Boët, P., Rochet, M.-J., Trenkel, V. M. and Niquil, N. (2013). Combining quantitative and qualitative models to identify functional groups for monitoring changes in the bay of biscay continental shelf exploited foodweb, *ICES Journal Of Marine Science* **71**(1): 105–117.

Levins, R. (1974). Discussion paper: the qualitative analysis of partially specified systems, *Annals of the New York Academy of Sciences* **231**(1): 123–138.

Loeuille, N. and Loreau, M. (2005). Evolutionary emergence of size-structured food webs, *Proceedings of the National Academy of Sciences of the United States of America* **102**(16): 5761–5766.

Marzloff, M. P., Dambacher, J. M., Johnson, C. R., Little, L. R. and Frusher, S. D. (2011). Exploring alternative states in ecological systems with a qualitative analysis of community feedback, *Ecological Modelling* **222**(15): 2651–2662.

May, R. M. (1972). Will a large complex system be stable?, *Nature* **238**: 413–414.

McCluskey, E.J., J. (1956). Minimization of Boolean functions, *Bell System Technical Journal* **35**(6): 1417–1444.

Melbourne-Thomas, J., Wotherspoon, S., Raymond, B. and Constable, A. (2012). Comprehensive evaluation of model uncertainty in qualitative network analyses, *Ecological Monographs* **82**(4): 505–519.

Metcalf, S. J., Pember, M. B. and Bellchambers, L. M. (2011). Identifying indicators of the effects of fishing using alternative models, uncertainty, and aggregation error, *ICES Journal of Marine Science* **68**(7): 1417–1425.

Mikkelson, J. M. (2004). Dissolving the wine/water paradox, *British Journal for the Philosophy of Science* **55**(1): 137–145.

Montaño-Moctezuma, G., Li, H. W. and Rossignol, P. A. (2007). Alternative community structures in a kelp-urchin community: a qualitative modeling approach, *Ecological Modelling* **205**(3): 343–354.

Nakajima, H. (1992). Sensitivity and stability of flow networks, *Ecological Modelling* **62**(1): 123–133.

Neutel, A.-M., Heesterbeek, J. A. and de Ruiter, P. C. (2002). Stability in real food webs: weak links in long loops, *Science* **296**(5570): 1120–1123.

Norton, J. D. (2008). Ignorance and indifference, *Philosophy of Science* **75**(1): 45–68.

Norton, J. D. (2010a). Cosmic confusions: not supporting versus supporting not-, *Philosophy of Science* **77**(4): 501–523.

Norton, J. D. (2010b). There are no universal rules for induction, *Philosophy of Science* **77**(5): 765–777.

Novak, M., Wootton, J. T., Doak, D. F., Emmerson, M., Estes, J. A. and Tinker, M. T. (2011). Predicting community responses to perturbations in the face of imperfect knowledge and network complexity, *Ecology* **92**(4): 836–846.

Ramos-Jiliberto, R., Garay-Narváez, L. and Medina, M. H. (2012). Retrospective qualitative analysis of ecological networks under environmental perturbation: a copper-polluted intertidal community as a case study, *Ecotoxicology* **21**(1): 234–243.

Raymond, B., McInnes, J., Dambacher, J. M., Way, S. and Bergstrom, D. M. (2011). Qualitative modelling of invasive species eradication on subantarctic Macquarie Island, *Journal of Applied Ecology* **48**(1): 181–191.

Reum, J. C., McDonald, P. S., Ferriss, B. E., Farrell, D. M., Harvey, C. J. and Levin, P. S. (2015). Qualitative network models in support of ecosystem approaches to bivalve aquaculture, *ICES Journal of Marine Science* **72**(8): 2278–2288.

Rochet, M.-J. and Rice, J. C. (2009). Simulation-based management strategy evaluation: ignorance disguised as mathematics?, *ICES Journal of Marine Science* **66**(4): 754–762.

Rochette, S., Lobry, J., Lepage, M. and Boët, P. (2009). Dealing with uncertainty in qualitative models with a semi-quantitative approach based on simulations. Application to the Gironde estuarine food web (France), *Ecological Modelling* **220**(2): 122–132.

Ruscoe, W. A., Ramsey, D. S., Pech, R. P., Sweetapple, P. J., Yockney, I., Barron, M. C., Perry, M., Nugent, G., Carran, R., Warne, R. et al. (2011). Unexpected consequences of control: competitive vs. predator release in a four-species assemblage of invasive mammals, *Ecology Letters* **14**(10): 1035–1042.

Schmitz, O. J. (1997). Press perturbations and the predictability of ecological interactions in a food web, *Ecology* **78**(1): 55–69.

Shackel, N. (2007). Bertrand's Paradox and the Principle of Indifference, *Philosophy of Science* **74**(2): 150–175.

Smith, M. (2014). Evidential incomparability and the Principle of Indifference, *Erkenntnis* **80**(3): 605–616.

Sobocinski, K. L., Greene, C. M. and Schmidt, M. W. (2018). Using a qualitative model to explore the impacts of ecosystem and anthropogenic drivers upon declining marine survival in pacific salmon, *Environmental Conservation* **45**(3): 278–290.

Sontag, E. D. (2014). A technique for determining the signs of sensitivities of steady states in chemical reaction networks, *IET Systems Biology* **8**(6): 251–267.

Theuns, P. (1994). A dichotomization method for Boolean analysis of quantifiable co-occurrence data, *in* G. H. Fischer and D. Laming (eds), *Contributions to Mathematical Psychology, Psychometrics, and Methodology*, Springer, New York, pp. 389–402.

Usmani, R. A. (1994). Inversion of a tridiagonal Jacobi matrix, *Linear Algebra and its Applications* **212**: 413–414.

Valiant, L. G. (1979). The complexity of computing the permanent, *Theoretical Computer Science* **8**(2): 189–201.

Wildermuth, R. P., Fay, G. and Gaichas, S. (2018). Structural uncertainty in qualitative models for ecosystem-based management of georges bank, *Canadian Journal of Fisheries and Aquatic Sciences* **75**(10): 1635–1643.

Williams, R. J. and Martinez, N. D. (2000). Simple rules yield complex food webs, *Nature* **404**(6774): 180–183.

Wootton, J. T. and Emmerson, M. (2005). Measurement of interaction strength in nature, *Annual Review of Ecology, Evolution, and Systematics* **36**: 419–444.

Yodzis, P. (1981). The stability of real ecosystems, *Nature* **289**(5799): 674–676.

Yodzis, P. (1988). The indeterminacy of ecological interactions as perceived through perturbation experiments, *Ecology* **69**(2): 508–515.

Supplementary Information

Dealing with high uncertainty in qualitative network models using Boolean analysis

Kristensen, et al. (2019)

Table of Contents

A An example showing the links between weighted-predictions matrix analysis, loop analysis, and Boolean analysis

The purpose of this appendix is to use the example network in Fig. A.5 to show the relationship between the weighted-predictions matrix (Dambacher et al., 2002), loop analysis (Lane and Levins, 1977), and Boolean analysis. According to the weighted-predictions matrix analysis, the species responses of interest are maximally ambiguous (A.1), however loop analysis shows that the species responses are nevertheless fully determined by the relationships between them (A.2). Boolean analysis uncovers these same deterministic relationships (A.3), and we show how to perform a Boolean minimisation by hand. The tutorial in SI C.1 shows how the example would be implemented in Python.

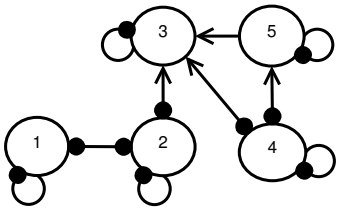


Figure A.5: The ecological interaction network (from Melbourne-Thomas et al., 2012) used for the example in this appendix. Nodes represent species, direct positive effects between species are indicated by an arrow, and negative effects by a filled circle.

A.1 Weighted-predictions matrix analysis

The weighted-predictions matrix W provides a summary of the positive and negative effects between species in a network in a press-perturbation scenario (Dambacher et al., 2002). The elements of the weighted-predictions matrix W_{ij} count the proportion of feedback cycles that conform to the sign given by the adjoint; therefore $W_{ij} = 1$ means that Species i will always respond to a negative press-perturbation of Species j in the direction given by the sign of the corresponding element of the adjoint, and $W_{ij} < 1$ indicates that a mixture of positive and negative feedback cycles are present and so the response direction is ambiguous.

Following Dambacher et al. (2002) and using the same notation, the weighted-predictions matrix for the interaction network in Fig. A.5 is found as follows.

From the community matrix A we obtain the qualitative community matrix $^\circ A$ such that:

$$^\circ A_{ij} = \begin{cases} 0 & \text{if } A_{ij} = 0 \\ -1 & \text{if } A_{ij} < 0 \\ +1 & \text{if } A_{ij} > 0. \end{cases} \tag{A.2}$$

For the network in Fig. A.5, the qualitative community matrix is

$$^\circ A = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & +1 & -1 & +1 & +1 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 0 & +1 & -1 \end{bmatrix} \tag{A.3}$$

By taking the classical adjoint or adjugate of the qualitative community matrix, $\text{adj}(-^\circ A)$, an analogue of the sensitivity matrix is obtained

$$-^\circ A^{-1} = \frac{\text{adj}(-^\circ A)}{\det(-^\circ A)} \tag{A.4}$$

where det() is the determinant, and adj() is the classical adjoint or adjugate. The determinant scales the magnitude of response to the press perturbation, and if the system is stable then $\det(-^\circ A)$ is

positive, while the adjugate gives the sum of feedback cycles that contribute to that species' positive or negative response. For the network in Fig. A.5, the adjoint is

$$\mathrm{adj}(-^{\circ}A) = \begin{bmatrix} +6 & -4 & +2 & +2 & 0 \\ -4 & +4 & -2 & -2 & 0 \\ -2 & +2 & 0 & 0 & 0 \\ +1 & -1 & 0 & +1 & -1 \\ +1 & -1 & 0 & +1 & +1 \end{bmatrix} \tag{A.5}$$

Each i, j element of the adjoint is the sum the positive feedback cycles minus the sum of the negative feedback cycles from j to i. In order to interpret its value as a proportion, one must also find the absolute number of positive and negative feedback cycles in total.

From the community matrix A we also obtain the absolute feedback matrix T, which counts the total number of cycles. This is done in two steps. First, we find the binary community matrix ${}^{\bullet}A$ such that:

$${}^{\bullet}A_{ij} = \begin{cases} 0 & \text{if } A_{ij} = 0 \\ +1 & \text{if } A_{ij} \neq 0. \end{cases} \tag{A.6}$$

Then the absolute feedback matrix T can be found

$$T_{ji} = \mathrm{per}(\min {}^{\bullet}A_{ij}), \tag{A.7}$$

where 'per' is the matrix permanent, and where $\min {}^{\bullet}A_{ij}$ is the submatrix found by removing row i and column j of ${}^{\bullet}A$. For the network in Fig. A.5, the absolute feedback matrix is

$$T = \begin{bmatrix} 6 & 4 & 2 & 2 & 2 \\ 4 & 4 & 2 & 2 & 2 \\ 2 & 2 & 4 & 4 & 4 \\ 1 & 1 & 2 & 3 & 5 \\ 1 & 1 & 2 & 3 & 5 \end{bmatrix} \tag{A.8}$$

Finally the weighted-predictions matrix can be found

$$W = \mathrm{abs}(\mathrm{adj}(-^{\circ}A))./T \tag{A.9}$$

where ./ indicates element-wise division, and by convention W_{ij} is set to 1 when $T_{ij} = 0$. For the network in Fig. A.5, the weighted-predictions matrix is

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1/3 & 1/5 \\ 1 & 1 & 0 & 1/3 & 1/5 \end{bmatrix} \tag{A.10}$$

If all feedback cycles are given equal weighting, then the elements of the weighted-predictions matrix can be interpreted probabilistically, and the magnitude of the W_{ij} elements can be interpreted as the degree of evidential support for the response-sign indicated by the adjoint (e.g. Metcalf et al., 2011; Ramos-Jiliberto et al., 2012). According to the weighted-predictions matrix in Eq. A.10, the effects of a press-perturbation of Species 3 upon Species 3, 4, and 5 are maximally ambiguous ($W_{3,3} = W_{4,3} = W_{5,3} = 0$), which is because the number of positive and negative feedbacks is equal. Therefore a probabilistic interpretation would assign a 50% probability of each outcome being either positive or negative.

A.2 Loop analysis

It is possible to obtain information about the species responses by analysing the adjoint of the community matrix directly (c.f. Lane and Levins (1977, p. 464), and Dambacher et al. (2003, p. 82)).

Recall that the sensitivity matrix $S = (-A)^{-1}$. Given that $(-A)^{-1} = \text{adj}(-A)/\det(-A)$, and given that in a stable system $\det(-A)$ is positive, then the signs of the elements of $\text{adj}(-A)$ are equal to the signs of the elements of S.

According to the weighted-predictions matrix analysis above, the effects of a press-perturbation of Species 3 upon Species 3, 4, and 5 are maximally ambiguous; however direct analysis reveals that they share a simple relationship. The elements of interest in the adjoint are

$$\text{adj}(-A)_{3,3} = -(a_{45}a_{54} + a_{44}a_{55})(a_{12}a_{21} - a_{11}a_{22}), \tag{A.11a}$$
$$\text{adj}(-A)_{4,3} = a_{43}a_{55}(a_{12}a_{21} - a_{11}a_{22}), \tag{A.11b}$$
$$\text{adj}(-A)_{5,3} = a_{43}a_{54}(a_{12}a_{21} - a_{11}a_{22}). \tag{A.11c}$$

From this, a relationship between the signs of the elements of the community matrix can be obtained

$$\text{sign}(S_{3,3}) = -\text{sign}(S_{4,3}) = -\text{sign}(S_{5,3}). \tag{A.12}$$

In contrast to the weighted-predictions matrix analysis, Eq. A.12 implies that the species responses are not so much 'ambiguous' as contingent upon one another. The relationships between the species responses can be expressed as logical implication statements, for example *if Species 3 responds positively then Species 4 responds negatively*.

A.3 Boolean analysis

The logical relationship between species responses revealed by direct analysis above can also be found using Boolean analysis. Let us assume that a large-enough sampling of the parameter space of the community matrix is performed so that *all* possible species-response combinations in Species 3, 4, and 5 to the negative perturbation of Species 3 are obtained. For simplicity, let us assume that the responses of interest are always positive or negative. Then, because the species-responses are dichotomous, the species-response combinations can be encoded as a truth table, as shown in Table A.3.

Table A.3: A truth table listing all possible species response-combinations, and whether they were never observed or observed. The values '0' correspond to 'false' and '1' to 'true'.

	positive response to negative perturbation of spp 3			never
ID	spp 3	spp 4	spp 5	observed
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

The first step take is to reduce the length of the table by focusing only on those response-combinations that were never observed, resulting in Table A.4.

Table A.4: After reducing the truth table to only those species response-combinations that were never observed, then two rows are selected that can be simplified using Boolean algebra.

ID	positive response to negative perturbation of spp 3			never observed
	spp 3	spp 4	spp 5	
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

The two rows highlighted in Table A.4 can be simplified using Boolean algebra as follows

$$(\text{spp 3} \wedge \text{spp 4} \wedge \overline{\text{spp 5}}) \vee (\text{spp 3} \wedge \text{spp 4} \wedge \text{spp 5}) = \text{spp 3} \wedge \text{spp 4} \wedge (\overline{\text{spp 5}} \vee \text{spp 5}) = \text{spp 3} \wedge \text{spp 4} \quad \text{(A.13)}$$

where we have used the standard notation of a bar to indicate 'false' (a negative species response), and no bar to indicate 'true' (a positive species response). The result of the simplification is given in Table A.5, where a '-' is used to indicate species responses that can take either truth value.

Table A.5: Combining rows 6 and 7 gives a new final row in the truth table, where '-' indicates that the entry can take either truth value.

ID	positive response to negative perturbation of spp 3			never observed
	spp 3	spp 4	spp 5	
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
5	1	0	1	1
	1	1	-	1

In a similar way, one can simplify rows 1 and 5 of Table A.5, and then rows 0 and 2 of the resulting table, until one obtains the truth table in Table A.6 that cannot be simplified further.

Table A.6: The final truth table.

ID	positive response to negative perturbation of spp 3			never observed
	spp 3	spp 4	spp 5	
	0	-	0	1
	-	0	1	1
	1	1	-	1

A rows of the minimised truth table give the 'PCUs' (short for the French 'projection canonique ultime' or 'ultimate canonical projection'), which in this context are the minimal sets describing the never-observed species-response combinations (Theuns, 1994, Section 28.2). From the PCUs, the entire original truth table and all species-response combinations can be recovered. The three rows in Table A.6 describe three sets of never-observed species responses:

1. $\{\overline{\text{spp 3}}, \overline{\text{spp 5}}\}$,
2. $\{\overline{\text{spp 4}}, \text{spp 5}\}$, and

3. {spp 3, spp 4}.

These three never-observed responses have the corresponding meanings, "we never observe ...":

1. "... a decrease in Species 3 and a decrease in Species 5",
2. "... a decrease in Species 4 and an increase in Species 5", and
3. "... an increase in Species 3 and an increase in Species 4".

The truth table in Table A.6 is also the minimal representation of the original table, and the three sets of never-observed species responses are the minimal description of all species response relationships in the system. This table is minimal due to the order in which the simplifications above were performed, however in general a technique like the Quinne-McCluskey method is needed to perform the operations in the right order to get the minimal table.

When the description is minimal, these sets of never-observed species responses are known as the 'PCUs' (Theuns, 1994). Each PCU of length k can be turned into $2^k - 2$ logical implications about response-combinations that were observed:

1. $\overline{\text{spp 3}} \rightarrow \text{spp 5}$, $\overline{\text{spp 5}} \rightarrow \text{spp 5}$;
2. $\overline{\text{spp 4}} \rightarrow \overline{\text{spp 5}}$, $\text{spp 5} \rightarrow \text{spp 4}$; and
3. $\text{spp 3} \rightarrow \overline{\text{spp 4}}$, $\text{spp 4} \rightarrow \overline{\text{spp 3}}$,

with the corresponding meanings:

1. "a decrease in Species 3 implies an increase in Species 5", "a decrease in Species 5 implies an increase in Species 5";
2. "a decrease in Species 4 implies a decrease in Species 5", "an increase in Species 5 implies an increase in Species 4"; and
3. "an increase in Species 3 implies a decrease in Species 4", "an increase in Species 4 implies a decrease in Species 3".

By selecting one logical implication statement from each PCU, and by linking statements with shared terms where possible, all species-response combinations can be concisely described by a network of logical implications (Fig. 2d). The minimal table and the logical implications network derived from it holds all of the information needed to reconstruct a complete description of the species-response relationships.

B The community matrix as a derivative description of a Lotka-Volterra system

The purpose of this appendix is to show how the community matrix is a derivative description of the population dynamics compared to the Lotka-Volterra description, and also to describe some of the special properties of the community matrix.

B.1 Background: primary versus derivative properties and PoI

Mikkelson (2004) presents a philosophical argument that PoI should be applied to the primary description of a problem. The distinction between "primary" and "derivative" is that two things cannot differ with respect to their derivative properties without also differing with respect to their primary properties. For example, if two coins are flipped, then the property space may be described as $D_1 = \{TT; HT; TH; HH\}$ or $D_2 = \{2T; 1H, 1T; 2H\}$. Outcomes cannot differ in D_2 without also differing in D_1, therefore D_1 is the more primary description. When PoI is applied to D_1 it leads to the correct (observed) probabilities.

B.2 Showing that the community matrix is a derivative description

The community matrix is derived from the Lotka-Volterra description as follows. Eq. 1 has a steady-state at

$$\mathbf{n}^* = B^{-1}(-\mathbf{r}), \tag{B.14}$$

and a community matrix (also known as the Jacobian matrix)

$$J = \left[\left. \frac{\partial \dot{n}_i}{\partial n_j} \right|_{\mathbf{n}^*} \right] = N^* B \tag{B.15}$$

where N^* is the diagonal matrix formed from $\mathbf{n}^*$. It is assumed that all $n_i^* > 0$, therefore the sign structure in J is the same as the sign structure in B i.e. determined by the signs of the direct interactions in the interaction network (e.g. Fig. 1a). To randomly sample parameter values, a positive scaling factor may be chosen and multiplied by J in Eq. B.15 so that the magnitude of all non-zero elements of J are between 0 and 1.

Another common way to define the community matrix (e.g. Appendices Melbourne-Thomas et al., 2012) is to scale the population-size state variables first

$$\hat{n}_i = \frac{n_i}{n_i^*}, \tag{B.16}$$

so that the steady state

$$\hat{\mathbf{n}}^* = \mathbf{1}. \tag{B.17}$$

Then

$$\frac{d\hat{n}_i}{dt} = \hat{n}_i \left(r_i + \sum_j b_{ij} \hat{n}_i n_j^* \right), \tag{B.18}$$

and the community matrix

$$A = \left[\left. \frac{\partial \dot{\hat{n}}_i}{\partial \hat{n}_j} \right|_{\hat{\mathbf{n}}^* = [1]} \right] = BN^* = (N^*)^{-1} J N^*. \tag{B.19}$$

Again a scaling can be applied so that all elements of A are between 0 and 1.

The community matrix A has three special properties that make it useful for Monte Carlo simulations and predicting species responses to pest control. First, because it is assumed that all species

have a positive steady state $n_i^* > 0$, then it preserves the sign structure of J, which is ultimately determined by the signs of the direct interactions. Second, Eq. B.19 is a similarity transform, therefore the eigenvalues (but not eigenvectors) of A and J are equal, and stability can be determined from either. Third, the sensitivity matrix $S = -A^{-1}$ has the same sign structure as $-J^{-1}$, therefore the signs of the species responses determined from J are preserved.

The Lotka-Volterra description is more primary than the community matrix because many B-and-N^* pairs can satisfy Eq. B.15 or B.19 to give the same J or A. Recall that the way to distinguish between primary and derivative properties is: two things cannot differ with respect to their derivative properties without also differing with respect to their primary properties. Each A represents a family of B-and-N pairs, therefore systems can differ in their Lotka-Volterra descriptions yet have the same community matrix description, however two systems with different community matrix descriptions cannot have the same Lotka-Volterra description.

C Tutorials

The following tutorials are available from the Github repository as Jupyter notebooks: https://github.com/nadiahpk/qualitative-modelling/tree/master/tutorials

C.1 Implementing the small example from Fig. 2

The purpose of this tutorial is to show how the weighted-predictions matrix approach, Monte Carlo simulations, and the Boolean approach, would be implemented for the small example from Fig. 2. SI A also shows the theoretical links between loop-analysis, the weighted-predictions matrix, and the Boolean approach.

The weighted-predictions matrix was found using SageMath in Jupyter, and the Notebook is archived as: Tutorial-1.1-Weighted_predictions_matrix_approach.ipynb

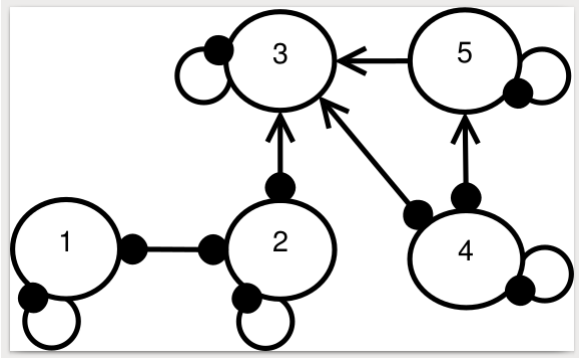
The Monte Carlo simulations and Boolean analysis were performed in Python3 in Jupyter, and the Notebook is archived as:
Tutorial-1.2-Probabilistic_and_Boolean_approaches.ipynb

C.1.1 Finding the weighted-predictions matrix

Weighted-predictions matrix

The weighted-predictions matrix, W, introduced by Dambacher et al. (2002), summarises the positive and negative effects of a press-perturbation on species in an interaction network. In this tutorial, we obtain W for the species interaction network below (`fivevariable2.png`), and use it to predict the effects of a negative press-perturbation of species 3 on species 3, 4, and 5 in the network below. This example is also used in Fig. 2 in the main test of the paper and Supplement A.

```
In [1]: from IPython.display import IFrame
        from IPython.display import Image
        import os

        # display the interaction network used in the example
        #IFrame("fivevariable2.pdf", width=500, height=400)
```



`fivevariable2.png`

Obtain the weighted-predictions matrix

Step 1: Create the qualitative community matrix
It is convenient to start by encoding the interaction network in a flexible way. Below, we specify:

- `spp_list`, a list of species names;
- `positive_edges_dict`, a dictionary that specifies the positive interactions between species; and
- `negative_edges_dict`, a dictionary that specifies the negative interactions between species.

```
In [2]: #
        spp_list = ['s1','s2','s3','s4','s5']

        # key is recipient of a positive effect
        positive_edges_dict = {
        's5': ['s4'],
        's3': ['s2', 's4', 's5'],
        }

        # key is recipient of a negative effect
        negative_edges_dict = {
        's5': ['s5'],
        's4': ['s4', 's5', 's3'],
        's3': ['s3'],
        's2': ['s2', 's1', 's3'],
        's1': ['s1', 's2'],
        }
```

Because we will be working with matrices, the first step is to assign each species name to an index in the matrix.

```
In [3]: sz = len(spp_list) # the number of species

        # a dictionary that maps from a species' name to its index in the
        # matrix
        spp2idx = { spp_name: idx
                    for idx, spp_name in enumerate(spp_list) }
        spp2idx

Out[3]: {'s1': 0, 's2': 1, 's3': 2, 's4': 3, 's5': 4}
```

The qualitative community matrix, denoted $^{\circ}A$ in Dambacher et al. (2002), has a -1 for negative species interactions, +1 for positive interactions, and 0 otherwise. So we use the edges dictionaries together with the spp2idx dictionary to populate the qualitative community matrix, Aq, with zeros, ones, and negative ones accordingly.

```
In [4]: Aq = matrix(QQ, sz, sz)


        for recipient, giverList in positive_edges_dict.items():
            for giver in giverList:
                Aq[ spp2idx[recipient], spp2idx[giver]] = 1

        for recipient, giverList in negative_edges_dict.items():
            for giver in giverList:
                Aq[ spp2idx[recipient], spp2idx[giver]] = -1

        Aq

Out[4]: [-1 -1  0  0  0]
        [-1 -1 -1  0  0]
        [ 0  1 -1  1  1]
        [ 0  0 -1 -1 -1]
        [ 0  0  0  1 -1]
```

Step 2: Find the adjoint of the qualitative community matrix

The classical adjoint or adjugate of the qualitative community matrix, adj($-^{\circ}A$), provides an analogue of the sensitivity matrix

$$-^{\circ}A^{-1} = \frac{\text{adj}(-^{\circ}A)}{\det(-^{\circ}A)}$$

where det() is the determinant, and adj() is the classical adjoint or adjugate. It equally weights each feedback loop between species in the system (magnitude = 1 for all), and counts the sum of those positive (+1) and negative (-1) feedbacks.

```
In [5]: adj = (-Aq).adjoint()
        adj
```

```
Out[5]: [ 6 -4  2  2  0]
        [-4  4 -2 -2  0]
        [-2  2  0  0  0]
        [ 1 -1  0  1 -1]
        [ 1 -1  0  1  1]
```

Because the elements in adj($-^{\circ}A$) are the sum of positive and negative feedbacks, their signs are indicative of the response of each row-species to a negative press-perturbation of each column-species. For example (taken from Dambacher et al. (2002)), if there are 4 positive feedback loops between two species, then this would result in an element with value +4. However, an element with value +4 may also result from 44 positive and 40 negative loops, or 6 positive and 2 negative loops, etc. Because the elements are a simple sum, more information is needed to interpret the result probabilistically; additionally, we need to know what the total number of feedback loops is.

Step 3: Find the absolute feedback matrix from the binary community matrix

To obtain the total number of feedback loops, we first create the binary community matrix. The binary community matrix, $^{\bullet}A$, has a +1 for any interaction between species, regardless of its sign, and a 0 otherwise.

```
In [6]: # create binary community matrix
        Ab = matrix(QQ, Aq.nrows(), Aq.ncols())
        for position in Aq.nonzero_positions():
            Ab[position] = 1
        Ab
```

```
Out[6]: [1 1 0 0 0]
        [1 1 1 0 0]
        [0 1 1 1 1]
        [0 0 1 1 1]
        [0 0 0 1 1]
```

The total count of feedback loops is then obtained from the absolute feedback matrix, T. T is obtained from $^{\bullet}A$

$$T_{ji} = \text{per}(\min {}^{\bullet}A_{ij}),$$

where 'per' is the matrix permanent, and where $\min {}^{\bullet}A_{ij}$ is the submatrix found by removing row i and column j of $^{\bullet}A$. The matrix permanent is calculated in a similar way to the determinant but with all terms in the sum having a positive sign.

```
In [7]: # absolute feedback matrix
        T = matrix(QQ,sz,sz)
        for row in range(sz):
            for col in range(sz):
                keeprows = range(sz)
                keeprows.remove(row)
```

```
            keepcols = range(sz)
            keepcols.remove(col)
            submatrix = Ab.matrix_from_rows_and_columns(keeprows, keepcols)
            T[col,row] = submatrix.permanent()

    T
```

```
Out[7]: [6 4 2 2 2]
        [4 4 2 2 2]
        [2 2 4 4 4]
        [1 1 2 3 5]
        [1 1 2 3 5]
```

Step 4: Use the adjoint of the qualitative community matrix and the absolute feedback matrix to find the weighted-predictions matrix

The weighted predictions matrix W is calculated by dividing the absolute value of each element of the adjoint by the corresponding element of T

$$W = \text{abs}(\text{adj}(-^{\circ}A))./T$$

where ./ indicates element-wise division. By convention, W_{ij} is set to 1 when $T_{ij} = 0$.

```
In [8]: # absolute value of adjugate matrix
        abs_adj = adj.apply_map(abs)

        # element-wise fraction of the adjugate to the
        # absolute feedback matrix
        fnc = lambda x: 0 if x == 0 else 1/x
        W = abs_adj.elementwise_product(T.apply_map(fnc))

        # Though for independent species,
        # we set the weighted predictions matrix value to 1
        fnc = lambda x: x == 0
        for position in (T.find(fnc, indices=True)).iterkeys():
            W[position] = 1
        W
```

```
Out[8]: [  1   1   1   1   0]
        [  1   1   1   1   0]
        [  1   1   0   0   0]
        [  1   1   0 1/3 1/5]
        [  1   1   0 1/3 1/5]
```

Interpret the weighted predictions matrix

From the analysis above, we have

$$\text{adj}(-^{\circ}A) = \begin{bmatrix} +6 & -4 & +2 & +2 & 0 \\ -4 & +4 & -2 & -2 & 0 \\ -2 & +2 & 0 & 0 & 0 \\ +1 & -1 & 0 & +1 & -1 \\ +1 & -1 & 0 & +1 & +1 \end{bmatrix}, T = \begin{bmatrix} 6 & 4 & 2 & 2 & 2 \\ 4 & 4 & 2 & 2 & 2 \\ 2 & 2 & 4 & 4 & 4 \\ 1 & 1 & 2 & 3 & 5 \\ 1 & 1 & 2 & 3 & 5 \end{bmatrix}, W = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1/3 & 1/5 \\ 1 & 1 & 0 & 1/3 & 1/5 \end{bmatrix}$$

As explained in Dambacher et al. (2002), the elements in W, ranging between 0 and 1, indicate the determinacy of the response sign prediction obtained from $\text{adj}(-^{\circ}A)$. A value of 1 indicates that all of the feedbacks between a row and column species are of the same sign. This means that

the species response to a press perturbation, regardless of the specific interaction-strength values, is guaranteed to match the sign in adj$(-^{\circ}A)$. Values less than one indicate indeterminacy: there is a combination of positive and negative feedback loops between the species in the system, and so the species response may be positive or negative, depending upon the specific interaction-strength values. A value of 0 indicates either that there are no feedbacks between the row and column species (see convention above), or that the number of positive and negative feedbacks are equal.

Dambacher et al. (2002) further interpreted the magnitude of $W_{ij} < 1$ values as indicative of the degree of indeterminacy in the species response. Based on Monte Carlo simulations, they recommended a threshold of $W_{ij} > 0.5$ as a general guideline for high ($\approx$ 95% or more) sign determinacy (Dambacher et al. 2001). $W_{ij} > 0.5$ corresponds to > 75% of feedback loops having the same sign. Hosack et al. (2008) also used simulations, where interaction-strength values were sampled from various distributions, to obtain a relationship W_{ij} and sign-determinacy.

In the example in the paper, we are interested in predicting the effects of a negative press-perturbation of species 3 on species 3, 4, and 5. This corresponds to column 3, rows 3, 4, and 5 of the matrices above. In the weighted-predictions matrix, $W_{3,3} = W_{4,3} = W_{5,3} = 0$. Given that $T_{ij} \neq 0$ for the corresponding elements, this means that the response signs are maximally indeterminate: there are an equal number of positive and negative feedbacks between the perturbed species 3 and the other species.

```
In [9]: # the proportion of feedback loops that are positive
        fnc = lambda x: 0 if x == 0 else 1/x
        propn_pos = (1/2)*((adj+T).elementwise_product(T.apply_map(fnc)))
        propn_pos
```

```
Out[9]: [  1   0   1   1 1/2]
        [  0   1   0   0 1/2]
        [  0   1 1/2 1/2 1/2]
        [  1   0 1/2 2/3 2/5]
        [  1   0 1/2 2/3 3/5]
```

If the results of the weighted-predictions matrix analysis were interpreted probabilistically, then giving equal weighting to each feedback, the predictions are equivalent to coin flip between positive or negative response (c.f. Dambacher et al.'s (2002) 'old-field web' results).

C.1.2 Using Monte Carlo simulations and the Boolean approach

Define the species interaction network

We define the species interaction network:

- `positive_edges_dict`, a dictionary that specifies the positive interactions between species; and
- `negative_edges_dict`, a dictionary that specifies the negative interactions between species.

```
In [1]: # spp_list = ['s1','s2','s3','s4','s5']

        # key is recipient of a positive effect
        positive_edges_dict = {
        's5': ['s4'],
        's3': ['s2', 's4', 's5'],
        }

        # key is recipient of a negative effect
        negative_edges_dict = {
        's5': ['s5'],
        's4': ['s4', 's5', 's3'],
        's3': ['s3'],
```

```
        's2': ['s2', 's1', 's3'],
        's1': ['s1', 's2'],
        }
```

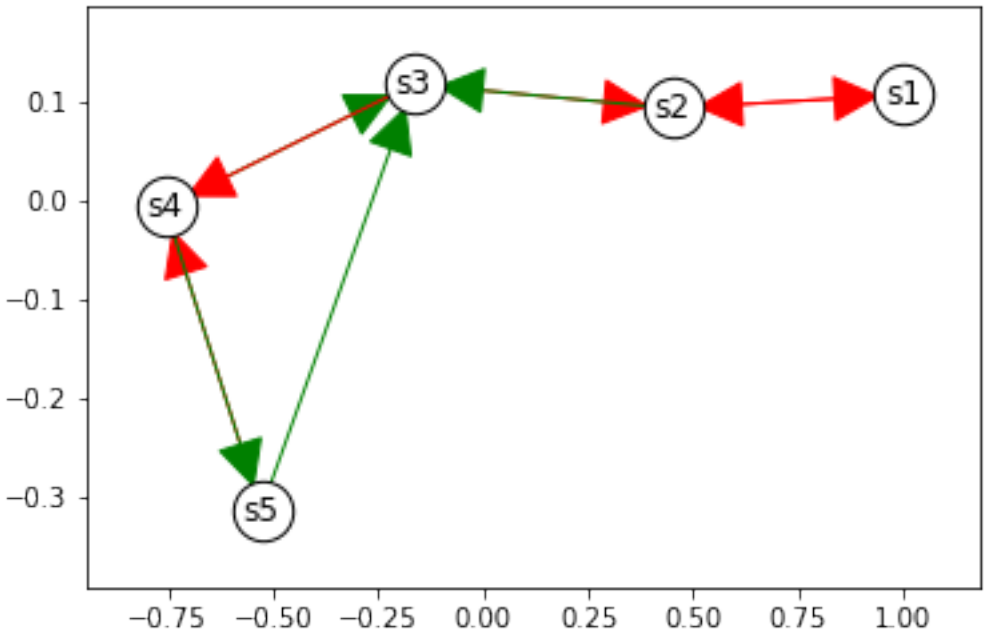
We encode the network structure as a `networkx` digraph. The function `initialise_foodweb` stores the signs of the interactions in the edge data dictionary.

```
In [2]: from qualmod import initialise_foodweb

        web = initialise_foodweb(positive_edges_dict, negative_edges_dict)

        # print edges
        for edge in web.edges(data=True):
            print(edge)
```

```
('s5', 's5', {'color': 'red', 'sign': -1})
('s5', 's4', {'color': 'red', 'sign': -1})
('s5', 's3', {'color': 'green', 'sign': 1})
('s4', 's4', {'color': 'red', 'sign': -1})
('s4', 's5', {'color': 'green', 'sign': 1})
('s4', 's3', {'color': 'green', 'sign': 1})
('s3', 's4', {'color': 'red', 'sign': -1})
('s3', 's3', {'color': 'red', 'sign': -1})
('s3', 's2', {'color': 'red', 'sign': -1})
('s2', 's2', {'color': 'red', 'sign': -1})
('s2', 's1', {'color': 'red', 'sign': -1})
('s2', 's3', {'color': 'green', 'sign': 1})
('s1', 's2', {'color': 'red', 'sign': -1})
('s1', 's1', {'color': 'red', 'sign': -1})
```

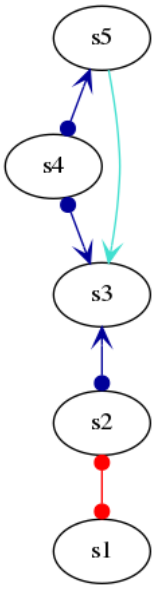
The package `networkx` provides a quick way to plot the interaction network.

```
In [3]: import networkx as nx
        import matplotlib.pyplot as plt

        pos=nx.spring_layout(web)
        nx.draw_networkx_labels(web, pos)
        nx.draw_networkx_nodes(web,
                               pos,
                               node_color = 'white',
                               edgecolors='black',
                               node_size=500)

        edge_colors = [ v[2]['color'] for v in web.edges(data=True) ]
        nx.draw_networkx_edges(web,
                               pos,
                               edge_color=edge_colors,
                               arrows=True,
                               arrowsize=40);
        plt.show()
```



The package `qualmod` also includes a function that uses `graphviz` to draw the interaction network. This is useful if we want to create a `.pdf` of the network to embed in a document, etc.

```
In [4]: from qualmod import draw_foodweb
        from IPython.display import IFrame
        from IPython.display import Image
        import os

        # use draw_foodweb to create a graphviz dot file
        # defining the interaction network
        draw_foodweb(web, f_name = 'web.dot')

        # call graphviz externally to create a pdf of the
        # interaction network
        os.system("dot -Tpdf web.dot > web.pdf")

        # display the pdf of the interaction network here in Jupyter
        # IFrame("web.pdf", width=500, height=600)

        # call graphviz to create a png, display in
        # markdown cell
        os.system("dot -Tpng web.dot > web.png")

Out[4]: 0
```



web.png

The interaction network can also be converted into a qualitative community matrix, Mq below, using the function qualitative_community_matrix.

```
In [5]: from qualmod import qualitative_community_matrix

        (Mq, s2idx) = qualitative_community_matrix(web)
        print(Mq)
```

```
[[-1. -1.  0.  0.  0.]
 [-1. -1. -1.  0.  0.]
 [ 0.  1. -1.  1.  1.]
 [ 0.  0. -1. -1. -1.]
 [ 0.  0.  0.  1. -1.]]
```

The function qualitative_community_matrix also returns s2idx, which is a handy bidirection dictionary that maps the name of the species to their index in the qualitative community matrix.

```
In [6]: print(s2idx)
        print(s2idx['s1'])
        print(s2idx.inv[0])
```

```
bidict({'s1': 0, 's2': 1, 's3': 2, 's4': 3, 's5': 4})
0
s1
```

Randomly sampling perturbation responses / parameter sweep

In the example in Fig. 2 of the paper, we are interested in the response of species s3, s4, and s5 to a negative press-perturbation of species s3.

```
In [7]: # which species is controlled
        # (i.e. perturbed e.g. by pest control program)
        control_list = ['s3']

        # the species we're interested in the responses of
        monitored_list = ['s3','s4','s5']
```

In probabilistic Qualitative Modelling, the monitored species' response to the press-perturbation can be obtained using Monte Carlo simulation. The interaction-strength magnitudes in the community matrix are sampled from $a_{i,j} \sim \mathcal{U}(0,1)$, and the positive and negative responses are counted: `countResponses` below.

In the Boolean approach, random sampling can also be used to do a parameter-value sweep of the model behaviour. The objective is to obtain a set, `observedResponseCombinations`, that lists every species-response combination that is possible, regardless of what the interaction-strength magnitudes are. Here note that, although we again sample from a uniform distribution, the choice is of distribution is somewhat arbitrary. So long as we obtain good-enough coverage of the parameter space that we obtain every possible response combination, then any sampling distribution can be used.

In this example, we apply only one plausibility criterion: the community matrix must be stable.

The method uses the sensitivity matrix approach to obtain species responses (Sq).

```
In [8]: import numpy as np

        # size of the system, can also use sz = web.order()
        sz = Mq.shape[0]

        # initialise collected responses with empty sets
        observedResponseCombinations = set()

        # initialise counts of positive responses with zeros
        countResponses = np.array([0]*len(control_list)*len(monitored_list))

        noSamples = 100
        for n in range(noSamples):

            # find a random community matrix that is stable
            maxEig = 1
            while maxEig > 0:

                M = np.multiply( np.random.random_sample((sz,sz)), Mq )
                maxEig = max(np.real(np.linalg.eigvals(M)))

            # find sensitivity matrix
            Sq = - np.linalg.inv(M)

            # Boolean approach:
            # ---

            #  find the signs of responses of each monitored species
            #  to perturbations in each control species
            response = tuple(['neg' if Sq[s2idx[ms],s2idx[cs]]<0 else 'pos'
                              for cs in control_list
                              for ms in monitored_list ])


            #  add response to our collection of observed responses
            observedResponseCombinations.add(response)

            # Probabilistic approach:
            # ---

            #  add positive responses to count
            posResponses = np.array([ 0 if Sq[s2idx[ms],s2idx[cs]]<0 else 1
```

```
                                    for cs in control_list
                                    for ms in monitored_list ])
        countResponses += posResponses
```

The two types of intermediate results are shown below. For the probabilistic approach, we have a count of positive responses in each species:

```
In [9]: countResponses
```

```
Out[9]: array([77, 23, 23])
```

For the Boolean approach, we have a list of species-response combinations:

```
In [10]: observedResponseCombinations
```

```
Out[10]: {('neg', 'pos', 'pos'), ('pos', 'neg', 'neg')}
```

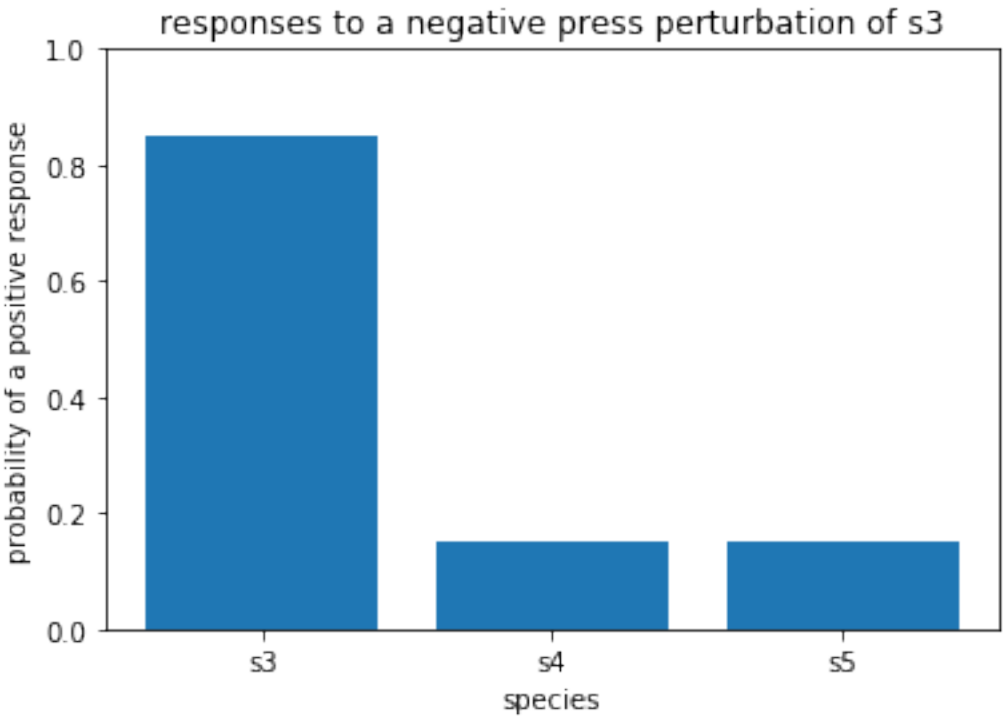
Probabilistic approach

In the probabilistic approach, the proportions of responses giving positive or negative species response are interpreted probabilistically.

In this example, `probabilityResponses` gives stronger support for a positive response in species s3, and gives stronger support for a negative response in species s4 and s5.

```
In [11]: probabilityResponses = countResponses / noSamples
```

```
        plt.title('responses to a negative press perturbation of s3')
        plt.bar(range(len(monitored_list)),
                probabilityResponses,
                tick_label=monitored_list)
        plt.ylim((0,1))
        plt.ylabel('probability of a positive response')
        plt.xlabel('species')
        plt.show()
```



Boolean approach

In the Boolean approach, the species response-combinations that the model predicts are interpreted using Boolean algebra.

Two species-response combinations were observed during the parameter-value sweep (shown as a table below).

```
In [12]: # the responses we'll be counting proportions on
         desiredResponses = [ cs + '_' + ms
                              for cs in control_list
                              for ms in monitored_list]

         # I have hardcoded the answer in here in case, by chance,
         # you miss a response combination in the random sampling above.
         # For a real example, you will need to check that a large no.
         # of samples has been drawn since the last new response
         # combination was found.

         if len(observedResponseCombinations) < 2:
             observedResponseCombinations = {
                 ('neg', 'pos', 'pos'),
                 ('pos', 'neg', 'neg')
             }

         # print the species response combinations that were observed
         # during the parameter sweep above in a table
         header = ' | '.join(desiredResponses)
         print(header)
         print('-'*len(header))
         for responseCombination in observedResponseCombinations:
             print(' ' + '  |  '.join(responseCombination))
```

```
s3_s3 | s3_s4 | s3_s5
---------------------
 neg  |  pos  |  pos
 pos  |  neg  |  neg
```

In order to use Boolean algebra, we need to convert the species responses into Boolean values. The responses have a natural dichotimisation i.e. a positive or a negative species response. Arbitrarily, we assign a positive species response as a `true`, and a negative species response as a `false`.

```
In [13]: str4true = 'pos'; str4false = 'neg'
```

The simulations have returned the species-response combinations that are *possible* in the model, but the Boolean analysis is performed on the *impossible* combinations instead. If we assume that parameter sweep (above) was sufficient to obtain every possible combination, then the set of impossible combinations is simply the complement of the set of observed responses. In other words, the set of impossible combinations is the set of every response-combination that is combinatorially possible minus the set of responses that we observed.

There are a few different ways that we could obtain the complement, but one convenient way is to first observe that every response combination can be interpreted as a binary string, which can be interpreted as an integer. Then the complement corresponds to the list of integers that are missing from the set of integers that we observed.

First, we encode each observed response as a binary string, which is then converted into an integer:

```
In [14]: observedInts = [
             int( ''.join(['1' if i in str4true else '0'
                           for i in responseCombination]), 2 )
             for responseCombination in observedResponseCombinations
         ]
         observedInts

Out[14]: [3, 4]
```

Then, the unobserved species responses correspond to the integers that are missing from observedInts:

```
In [15]: # Start with set of all possible responses as integers ...
         unobservedInts = set(range(2**len(desiredResponses)))
         # ... and loop through the observed response combinations,
         # discarding those that were observed

         for i in observedInts:
             unobservedInts.discard(i)

         unobservedInts

Out[15]: {0, 1, 2, 5, 6, 7}
```

The function getRespvarList2BoolvarList converts the possible species-responses in the model into Boolean variables. It uses the PyEDA package.

```
In [16]: from findpcu import getRespvarList2BoolvarList

         # create our boolean variables and some useful dictionaries
         x, x2s, r2idx = getRespvarList2BoolvarList(desiredResponses,
                                                    str4true,
                                                    str4false)
```

x is a list of the Boolean variables that have been created. For example, s3_s4 is the variable that describes the response of species s4 to a negative perturbation of s3. s3_s4 has the value True if s4 responds positively and False if s4 responds negatively.

```
In [17]: x

Out[17]: [s3_s3, s3_s4, s3_s5]
```

x2s and r2idx are a bidirectional dictionaries that convert between a variable values and a corresponding string or index, respectively.

The next step is to use the function intList2boolexpr to turn the list of integers unobservedInts, corresponding to unobserved species responses, into a Boolean expression.

```
In [18]: from findpcu import intList2boolexpr

         # the boolean expression for our unobserved responses
         unobservedBoolexpr = intList2boolexpr(unobservedInts, x)
```

The Boolean expression is returned in disjunctive normal form, as a sum of products. Here the notation ~ is used to represent Not. For example, ~s3_s4 means 'not a positive response in s4 to a negative press-perturbation in s3', or in other words, 'a negative response in s4'.

```
In [19]: print(unobservedBoolexpr)
```

```
Or(
        And(~s3_s3, ~s3_s4, ~s3_s5),
        And(~s3_s3, ~s3_s4, s3_s5),
        And(~s3_s3, s3_s4, ~s3_s5),
        And(s3_s3, ~s3_s4, s3_s5),
        And(s3_s3, s3_s4, ~s3_s5),
        And(s3_s3, s3_s4, s3_s5)
)
```

The list of unobserved species responses can also be easily converted into a truth table using the function `expr2truthtable` from PyEDA. This is the same table as given in Fig. 2a in the paper.

```
In [20]: from pyeda.inter import expr2truthtable
         expr2truthtable(unobservedBoolexpr)
```

```
Out[20]: s3_s5 s3_s4 s3_s3
             0     0     0 : 1
             0     0     1 : 0
             0     1     0 : 1
             0     1     1 : 1
             1     0     0 : 1
             1     0     1 : 1
             1     1     0 : 0
             1     1     1 : 1
```

In order to summarise the result, we perform a Boolean minimisation using the espresso algorithm.

```
In [21]: from pyeda.inter import espresso_exprs

         # Use espresso to minimise the unobservedBoolexpr
         boolExprMin, = espresso_exprs(unobservedBoolexpr)
```

When we print out the resulting minimised Boolean expression, we see that it is shorter and simpler than the original (compare to `unobservedBoolexpr` above).

```
In [22]: boolExprMin
```

```
Out[22]: Or(And(~s3_s4, s3_s5), And(s3_s3, s3_s4), And(~s3_s3, ~s3_s5))
```

Note however that `boolExprMin` is equivalent to the original expression. We can verify that by printing out its (full) truth table (using `expr2truthtable` again) and comparing it to the one above.

```
In [23]: expr2truthtable(boolExprMin)
```

```
Out[23]: s3_s5 s3_s4 s3_s3
             0     0     0 : 1
             0     0     1 : 0
             0     1     0 : 1
             0     1     1 : 1
             1     0     0 : 1
             1     0     1 : 1
             1     1     0 : 0
             1     1     1 : 1
```

Each of the And terms in the minimised Boolean expression (boolExprMin) is called a PCU (short for the French "projection canonique ultime" meaning "ultimate canonical projection"), and Theuns (1994) discusses some of the theory behind them.

The function boolexpr2RespvalList converts each of the PCUs into a string, which can be useful if we wish to store the results in a file. This function also appends the strings (in our case pos and neg) that we designated above as True and False, to make the output quicker to read.

```
In [24]: from findpcu import boolexpr2RespvalList

         # returns the PCUs of the boolean expression as a list of strings
         PCUList = boolexpr2RespvalList(boolExprMin, x2s)

         PCUList
```

```
Out[24]:
[
    ['poss3_s3', 'poss3_s4'],
    ['negs3_s5', 'negs3_s3'],
    ['poss3_s5', 'negs3_s4']
]
```

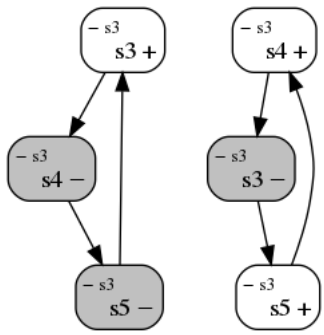
Each PCU can be used to derive a set of logical implications, and these implications can be chained together to create a logical implication network.

All that is needed to ensure that the logical implication network describes the full behaviour of the model is to ensure that at least one implication is derived from each PCU. However, for small networks, it may be preferable to include more than one implication. The function draw_implication_network draws the network in such a way that every implication with 1 antecedent is included.

```
In [25]: from findpcu import draw_implication_network

         draw_implication_network(PCUList)

         # display the pdf of the interaction network here in Jupyter
         # IFrame("implication_network.pdf", width=500, height=500)

         # call graphviz to create a png, display in markdown cell
         os.system("dot -Tpng implication_network.dot > implication_network.png")
```

```
implication_network.pdf has been created
```

```
Out[25]: 0
```



implication_network.png

We can see that each PCU is represented twice in the implication network above. For example, ['poss3_s3', 'poss3_s4'] translates to saying that the following combination of species responses is impossible in the model: positive s3 and positive s4. Therefore a positive response in s3 implies

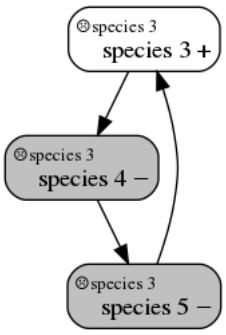
a negative response in s4 (seen in left-hand subnetwork), and a positive response in s4 implies a negative response in s3 (seen in right-hand subnetwork).

Another function is available, `draw_implication_network2`, to draw the implication network with only one implication per PCU. We can specify which species responses we wish to see as the antecedents. The example below also shows some of its other options.

```
In [26]: from findpcu import draw_implication_network2

         # make a positive response of s3 always an antecedent
         alwaysAnteList = ['poss3_s3', 'negs3_s5', 'negs3_s4']

         niceNames = { # names of each species
             's3': 'species 3',
             's4': 'species 4',
             's5': 'species 5',
         }

         fName = 'implication_network_2' # name of the .dot file and .pdf file

         controlSymbol = '&#9785;' # html code for a sad face

         draw_implication_network2(PCUList,
                                   alwaysAnteList,
                                   fName,
                                   niceNames,
                                   controlSymbol)

         os.system("dot -Tpng implication_network_2.dot > implication_network_2.png")
```

```
implication_network_2.pdf has been created
```

```
Out[26]: 0
```



implication_network_2.png

C.2 Implementing the Boolean approach on the Macquarie Island case study

The purpose of this tutorial is to explain how a Boolean analysis of the Macquarie Island case study was performed.

This Jupyter Notebook is archived as: Tutorial-2-Macquarie_Island_case_study.ipynb

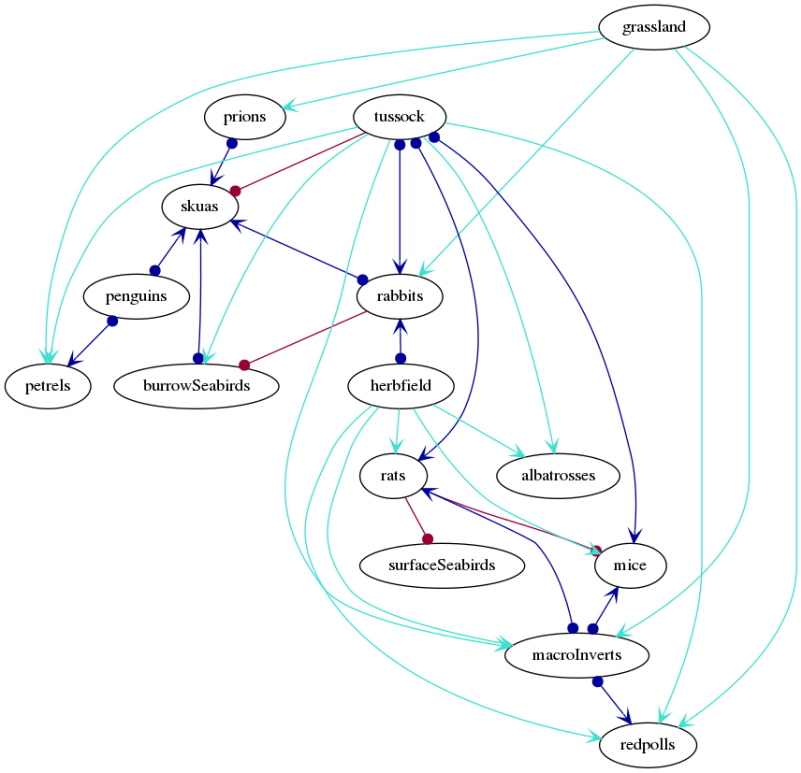
Notebook

We define the Macquarie Island interaction network by the species present, the interactions between them, and the signs of these interactions. The network structure is encoded as a `networkx` digraph. The function `initialise_foodweb` stores the signs of the interactions in the edge data dictionary. The network can be drawn using `draw_foodweb`.

```
In [90]: from qualmod import initialise_foodweb, draw_foodweb
         from qualmod import qualitative_community_matrix, get_conditions_lists
         import numpy as np
         import time

         from IPython.display import Image
         import os

         sppList = [
             'albatrosses',
             'prions',
             'burrowSeabirds',
             'petrels',
             'herbfield',
             'macroInverts',
             'mice',
             'penguins',
             'rabbits',
             'rats',
             'redpolls',
             'skuas',
             'surfaceSeabirds',
             'tussock',
         ]

         # key is recipient of positive effect
         positive_edges_dict = {
             'prions':           ['grassland'],
             'skuas':            ['prions','burrowSeabirds','rabbits','penguins'],
             'petrels':          ['penguins','tussock','grassland'],
             'mice':             ['herbfield','macroInverts','tussock'],
             'rats':             ['macroInverts','herbfield','tussock'],
             'burrowSeabirds':   ['tussock'],
             'rabbits':          ['tussock','herbfield','grassland'],
             'macroInverts':     ['herbfield','grassland','tussock'],
             'albatrosses':      ['tussock','herbfield'],
             'redpolls':         ['macroInverts','tussock','herbfield','grassland'],
         }
         negative_edges_dict = {
             'prions':           ['prions','skuas'],
             'skuas':            ['skuas','tussock'],
             'penguins':         ['penguins','skuas','petrels'],
             'petrels':          ['petrels'],
```

```
    'mice':             ['mice','rats'],
    'rats':             ['rats'],
    'burrowSeabirds':   ['burrowSeabirds','skuas','rabbits'],
    'rabbits':          ['rabbits','skuas'],
    'surfaceSeabirds':  ['surfaceSeabirds','rats'],
    'macroInverts':     ['macroInverts','rats','mice','redpolls'],
    'tussock':          ['tussock','mice','rats','rabbits'],
    'albatrosses':      ['albatrosses'],
    'herbfield':        ['herbfield','rabbits'],
    'grassland':        ['grassland'],
    'redpolls':         ['redpolls'],
}

web = initialise_foodweb(positive_edges_dict, negative_edges_dict)

draw_foodweb(web, f_name = 'macq1.dot')
# call graphviz to create a png, display in markdown cell
os.system("dot -Tpng macq1.dot > macq1.png")
```

Out[90]: 0



Macquarie Island interaction network.
We encode the validation criteria (with respect to a positive perturbation of the rabbits).

In [74]:
```
control_list = ['rabbits']

# Reponse to increase in rabbits
validation = {
    'rabbits': +1,
    'tussock': -1,
}
```

We convert the interaction network into a qualitative community matrix, `Mq` below, using the function `qualitative_community_matrix`.

```
In [78]: sz = web.order()
         (Mq, s2idx) = qualitative_community_matrix(web)
         print(Mq)
```

```
[[-1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  1.]
 [ 0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.  0.  0.  0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  1. -1. -1.  0.  0.  0.  0. -1. -1.  0.  0.  1.]
 [ 0.  0.  0.  1.  1. -1.  0.  0.  0.  0. -1.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  0. -1. -1.  0.  0.  0.  0. -1.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  1. -1.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  1.  0.  0.  0.  0.  0. -1.  0.  0.  0. -1.  0.  0.]
 [ 0.  0.  1.  1.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  1.]
 [ 0.  0.  0.  1.  1.  0.  0.  0.  0.  0. -1.  0.  0.  0.  1.]
 [ 0.  0.  1.  1.  1.  0.  0.  0.  0.  0.  0. -1.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.  0.  1.  0.  1.  1.  0.  0. -1.  0. -1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0.  0. -1. -1.  0.  0.  0. -1.]]
```

The `s2idx` returned is a bidirection dictionary that maps the name of the species to their index in the qualitative community matrix

```
In [79]: s2idx.inv[9]
```

```
Out[79]: 'rabbits'
```

The validation conditions with respect to the indices of the qualitative matrix can be obtained using the function `get_conditions_list`.

```
In [77]: # validation condition
         condn_idxs, condn = get_conditions_lists(s2idx, validation)

         print(list(zip(condn_idxs, condn)))
```

```
[(9, 1), (14, -1)]
```

To save time in this tutorial, I have hard-coded in the number of species-response combinations to be found, which is 36. If you do not know what the number is beforehand, you would use a line similar to the one commented-out above the start of the `while` loop.

The `while` loop collects species response combinations using a parameter-value sweep of the model.

```
In [91]: # Initialise collected responses with empty sets
         # Key is species perturbed and set will contain tuples of responses
         collectedResponses = {spp: set() for spp in control_list}

         t = 1
         t_last_updated = 0
         start_time = time.time()

         # use something like this if you don't
         # know the answer beforehand ~ 10 minutes
         # while t < 1e6:
```

```
    # I know there are 36 responses to find
    while len(collectedResponses['rabbits']) < 36:


        # Find valid stable community matrix
        valid = False

        while not valid:

            # Find a random community matrix that is stable
            maxEig = 1

            while maxEig > 0:
                M = np.multiply(np.random.random_sample((sz,sz)), Mq)
                maxEig = max(np.real(np.linalg.eigvals(M)))

            # Now have a stable matrix

            # Find sensitivity matrix
            Sq = - np.linalg.inv(M)

            # Check validation criteria
            valid = all(np.sign(Sq[condn_idxs,s2idx['rabbits']]) == condn)

        # Now have a valid stable community matrix

        for ps in control_list:

            # NOTE: looking at increase in rabbits not decrease here
            response = tuple('neg' if Sq[s2idx[ms],s2idx[ps]]<0 else 'pos'
                             if Sq[s2idx[ms],s2idx[ps]]>0 else 'zer'
                             for ms in sppList)

            if response not in collectedResponses[ps]:

                # fs[ps].write(','.join(response) + '\n')
                collectedResponses[ps].add(response)
                t_last_updated = t

        t += 1

    end_time = time.time()
    time_elapsed = end_time - start_time
    print('time elapsed = ' + str(time_elapsed) + ' seconds')
    print('number of matrices sampled = ' + str(t-1))
    print('last new combination found at t = ' + str(t_last_updated))
```

```
time elapsed = 0.5220227241516113 seconds
number of matrices sampled = 889
last new combination found at t = 889
```

The possible species-response combinations are now stored in `collectedResponses['rabbits']`. If more than one control-species is involved, then its name would be the key for the `collectedResponses` dictionary.

```
In [95]: print( ' '.join([ ss[:3] for ss in sppList]))
```

```
        for response in collectedResponses['rabbits']:
            print(' '.join(response))
```

```
alb pri bur pet her mac mic pen rab rat red sku sur tus
neg pos neg neg neg pos neg pos pos pos neg neg neg neg
neg pos neg neg neg neg neg pos pos neg neg neg pos neg
neg neg neg neg neg neg neg neg pos neg neg pos pos neg
neg neg neg neg neg neg pos pos pos neg neg pos pos neg
neg pos neg neg neg pos pos pos pos neg pos neg pos neg
neg pos neg neg neg pos neg pos pos pos pos neg neg neg
neg pos neg pos neg pos neg pos pos neg pos neg pos neg
neg pos neg pos neg pos pos pos pos neg neg neg pos neg
neg pos neg pos neg pos pos pos pos pos neg neg neg neg
neg pos neg pos neg neg pos pos pos neg neg neg pos neg
neg pos neg pos neg pos neg pos pos pos neg neg neg neg
neg pos neg pos neg neg neg pos pos neg neg neg pos neg
neg neg neg neg neg neg pos neg pos neg neg pos pos neg
neg pos neg pos neg pos neg pos pos neg neg neg pos neg
neg pos neg pos neg pos pos pos pos neg pos neg pos neg
neg neg neg neg neg neg neg pos pos neg neg pos pos neg
neg pos neg neg neg pos pos pos pos pos neg neg neg neg
neg pos neg neg neg neg pos pos pos neg neg neg pos neg
neg neg neg neg neg pos pos pos pos pos neg pos neg neg
neg neg neg neg neg pos neg neg pos pos neg pos neg neg
neg neg neg neg neg pos neg pos pos pos pos pos neg neg
neg pos neg pos neg pos neg pos pos pos pos neg neg neg
neg neg neg neg neg pos neg pos pos pos neg pos neg neg
neg neg neg neg neg pos neg neg pos pos pos pos neg neg
neg neg neg neg neg pos pos neg pos pos neg pos neg neg
neg neg neg neg neg pos pos pos pos neg neg pos pos neg
neg neg neg neg neg pos neg neg pos neg neg pos pos neg
neg pos neg neg neg pos pos pos pos neg neg neg pos neg
neg pos neg neg neg pos neg pos pos neg pos neg pos neg
neg neg neg neg neg pos pos neg pos neg neg pos pos neg
neg neg neg neg neg pos neg pos pos neg neg pos pos neg
neg neg neg neg neg pos pos pos pos neg pos pos pos neg
neg neg neg neg neg pos neg neg pos neg pos pos pos neg
neg neg neg neg neg pos neg pos pos neg pos pos pos neg
neg neg neg neg neg pos pos neg pos neg pos pos pos neg
neg pos neg neg neg pos neg pos pos neg neg neg pos neg
```

At this point in the process, it is a good idea to write the model responses. Here we write them to a csv file.

```
In [96]: # write output files

        fs = {ps: open('uniques_web1_' + ps + '.csv', 'w')
              for ps in control_list}

        for ps in control_list:

            header = [ps + '_' + ms for ms in sppList]
            fs[ps].write(','.join(header) + '\n')

            for response in collectedResponses[ps]:
```

```
                fs[ps].write(','.join(response) + '\n')

            fs[ps].close()

        print(header)
```

```
['rabbits_albatrosses', 'rabbits_prions', 'rabbits_burrowSeabirds',
        'rabbits_petrels', 'rabbits_herbfield', 'rabbits_macroInverts',
        'rabbits_mice', 'rabbits_penguins', 'rabbits_rabbits', 'rabbits_rats',
        'rabbits_redpolls', 'rabbits_skuas', 'rabbits_surfaceSeabirds',
        'rabbits_tussock']
```

From here onwards, we will be performing the Boolean analysis on the species responses that were found in the model.

We read in the responses from the csv file that was written previously.

```
In [92]: import csv
         from pyeda.inter import espresso_exprs
         from findpcu import getUnobservedInts, getRespvarList2BoolvarList
         from findpcu import intList2boolexpr, boolexpr2RespvalList

         fInName = ('uniques_web1_rabbits.csv')
         str4true = 'pos'
         str4false = 'neg'


         fIn = open(fInName)
         csv_f = csv.reader(fIn)
         allResponses = next(csv_f) # get the header
         fIn.close()

         allResponses
```

```
Out[92]: ['rabbits_albatrosses',
          'rabbits_prions',
          'rabbits_burrowSeabirds',
          'rabbits_petrels',
          'rabbits_herbfield',
          'rabbits_macroInverts',
          'rabbits_mice',
          'rabbits_penguins',
          'rabbits_rabbits',
          'rabbits_rats',
          'rabbits_redpolls',
          'rabbits_skuas',
          'rabbits_surfaceSeabirds',
          'rabbits_tussock']
```

We already know that the response of tussock to rabbits will be negative, so we write a list of desired responses, in the same order as `allResponses`, that omits tussock. The `desiredResponses` list is converted into a Boolean mask, and passed to the function `unobservedInts`, which uses the list of observed responses and desired responses to return a list of unobserved responses as a list of integers.

```
In [97]: # skip tussock, because its response was a validation criterion
         desiredResponses = [
```

```
            'rabbits_albatrosses',
            'rabbits_prions',
            'rabbits_burrowSeabirds',
            'rabbits_petrels',
            'rabbits_herbfield',
            'rabbits_macroInverts',
            'rabbits_mice',
            'rabbits_penguins',
            'rabbits_rats',
            'rabbits_redpolls',
            'rabbits_skuas',
            'rabbits_surfaceSeabirds']

        boolLen = len(desiredResponses)

        # A mask to take out only those entries in our desiredResponses
        desiredResponsesMask = [True if aR in desiredResponses
                                else False for aR in allResponses]

        unobservedInts = getUnobservedInts(fInName,
                                           desiredResponsesMask,
                                           boolLen, str4true)
```

```
In [85]: len(unobservedInts)
```

```
Out[85]: 4060
```

The function `getRespvarList2BoolvarList` is used to convert the possible species-responses in the model into Boolean variables. It uses the PyEDA package (https://pyeda.readthedocs.io/en/latest/).

The function `intList2boolexpr` turns the list of integers `unobservedInts`, corresponding to unobserved species responses, into a Boolean expression.

```
In [98]: # Create our boolean variables and some useful dictionaries
         x, x2s, r2idx = getRespvarList2BoolvarList(desiredResponses,
                                                    str4true, str4false)

         # Turn each integer representing unobserved into a
         # into boolean expression
         unobservedBoolexpr = intList2boolexpr(unobservedInts, x)
```

The function `espresso_exprs`, imported from PyEDA, is used to perform the Boolean minimisation.

```
In [87]: start_time = time.time()

         boolExprMin, = espresso_exprs(unobservedBoolexpr)

         end_time = time.time()
         time_elapsed = end_time - start_time # ~ 1 second
         print(str(time_elapsed))
```

```
1.2536895275115967
```

This results in a Boolean expression that summarises the species responses that were not observed in the parameter-sweep of the model, and are assumed to be impossible.

```
In [88]: boolExprMin
```

```
Out[88]: Or(rabbits_albatrosses, rabbits_burrowSeabirds, rabbits_herbfield,
    And(rabbits_mice, rabbits_redpolls, ~rabbits_surfaceSeabirds),
    And(~rabbits_rats, ~rabbits_surfaceSeabirds),
    And(rabbits_petrels, rabbits_skuas),
    And(~rabbits_macroInverts, rabbits_redpolls),
    And(~rabbits_prions, ~rabbits_skuas),
    And(~rabbits_macroInverts, ~rabbits_surfaceSeabirds),
    And(rabbits_prions, rabbits_skuas),
    And(rabbits_rats, rabbits_surfaceSeabirds),
    And(~rabbits_penguins, ~rabbits_skuas))
```

The function `boolexpr2RespvalList` converts each of the PCUs into a list of strings, which can be useful if we wish to store the results in a file, and is easier to read.

```
In [89]: PCUList = boolexpr2RespvalList(boolExprMin, x2s)
         PCUList
```

```
Out[89]: [['posrabbits_burrowSeabirds'],
          ['posrabbits_albatrosses'],
          ['posrabbits_herbfield'],
          ['negrabbits_surfaceSeabirds', 'negrabbits_rats'],
          ['negrabbits_penguins', 'negrabbits_skuas'],
          ['negrabbits_macroInverts', 'posrabbits_redpolls'],
          ['posrabbits_skuas', 'posrabbits_petrels'],
          ['negrabbits_skuas', 'negrabbits_prions'],
          ['posrabbits_prions', 'posrabbits_skuas'],
          ['posrabbits_rats', 'posrabbits_surfaceSeabirds'],
          ['negrabbits_macroInverts', 'negrabbits_surfaceSeabirds'],
          ['posrabbits_redpolls', 'negrabbits_surfaceSeabirds', 'posrabbits_mice']]
```

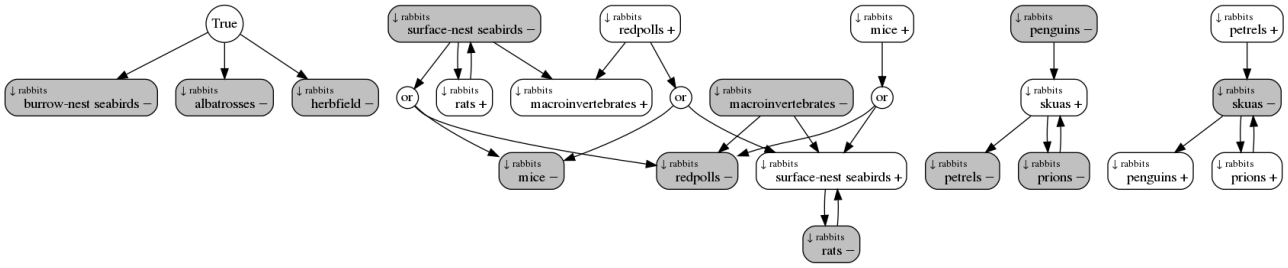
The `PCUList` form is also used as an input to the `draw_implication_network` function, which draws all logical implications resulting from the Boolean minimisation in their single-antecedent form.

```
In [100]: import os
          from findpcu import draw_implication_network

          niceNames = {
                  'rabbits': 'rabbits',
                  'petrels': 'petrels',
                  'mice': 'mice',
                  'burrowSeabirds': 'burrow-nest seabirds',
                  'macroInverts': 'macroinvertebrates',
                  'herbfield': 'herbfield',
                  'redpolls': 'redpolls',
                  'skuas': 'skuas',
                  'rats': 'rats',
                  'surfaceSeabirds': 'surface-nest seabirds',
                  'penguins': 'penguins',
                  'prions': 'prions',
                  'albatrosses': 'albatrosses',
                  'tussock': 'tussock'
                  }

          controlSymbol = '&darr; '
```

```
        draw_implication_network(PCUList,
                                 'uniques_web1_rabbits_pcus',
                                 niceNames, controlSymbol)

        # call graphviz to create a png, display in markdown cell
        cmd = "dot -Tpng uniques_web1_rabbits_pcus.dot > uniques_web1_rabbits_pcus.png"
        os.system( cmd )
```

```
uniques_web1_rabbits_pcus.pdf has been created
```

Out[100]: 0



Implication network.

C.3 Dealing with complicated implication networks

Large implication networks can be difficult to interpret, even when the individual subnetworks are small. In such cases, it may be necessary to separate the subnetworks and rearrange them for a useful interpretation. In this tutorial, we give examples of how decisions about the presentation of the network may be made, and how that may influence the form of the final network presented to conservation decision-makers. We use the Macquarie Island model again as the example, but this time consider the outcomes of rat control.

This Jupyter Notebook is archived as:
Tutorial-3-Dealing_with_complicated_implication_networks.ipynb

Notebook

In this example, we'll use the list of PCU's below (see Tutorial 2 for how to obtain PCUs) that were obtained from simulating rat control on Macquarie Island.

```
In [1]: PCUList = [
          ['posrats_surfaceSeabirds'],
          ['negrats_rabbits', 'negrats_herbfield'],
          ['negrats_prions', 'negrats_skuas'],
          ['posrats_herbfield', 'posrats_rabbits'],
          ['posrats_prions', 'posrats_skuas'],
          ['posrats_tussock', 'posrats_mice',
           'posrats_rabbits'],
          ['negrats_rabbits', 'negrats_burrowSeabirds',
           'posrats_tussock'],
          ['negrats_penguins', 'negrats_skuas',
           'negrats_tussock'],
          ['negrats_rabbits', 'posrats_tussock',
           'negrats_skuas'],
          ['posrats_skuas', 'posrats_petrels',
           'negrats_tussock'],
          ['posrats_redpolls', 'posrats_mice',
           'posrats_rabbits'],
          ['posrats_tussock', 'negrats_petrels',
           'negrats_skuas'],
          ['negrats_tussock', 'posrats_burrowSeabirds',
           'posrats_rabbits'],
          ['negrats_macroInverts', 'posrats_mice',
           'posrats_rabbits'],
          ['posrats_albatrosses', 'negrats_tussock',
           'posrats_rabbits'],
          ['posrats_skuas', 'negrats_tussock',
           'posrats_rabbits'],
          ['posrats_tussock', 'posrats_skuas',
           'posrats_penguins'],
          ['negrats_rabbits', 'posrats_tussock',
           'negrats_albatrosses'],
          ['negrats_rabbits', 'posrats_tussock',
           'negrats_redpolls', 'posrats_macroInverts'],
          ['negrats_macroInverts', 'posrats_redpolls',
           'negrats_tussock', 'posrats_rabbits'],
          ['posrats_petrels', 'posrats_mice', 'posrats_macroInverts',
           'posrats_redpolls', 'negrats_albatrosses'],
          ['posrats_petrels', 'posrats_mice',
           'posrats_burrowSeabirds', 'posrats_redpolls',
```

```
 'negrats_albatrosses'],
['posrats_mice', 'posrats_burrowSeabirds',
 'posrats_macroInverts', 'posrats_redpolls',
 'negrats_albatrosses', 'negrats_skuas']
]
```

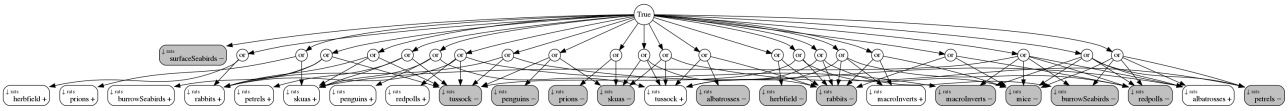
Our objective is to draw an implication network that can be interpreted for decision-makers planning a rat control programme.

In the first attempt, the implication network that results is very complicated and difficult to interpret.

```
In [3]: from findpcu import draw_implication_network, draw_implication_network2
        import os

        draw_implication_network2(PCUList,
                                  [],
                                  'attempt_1',
                                  niceNames = None,
                                  controlSymbol = 'downarrow')

        # call graphviz to create a png, display in markdown cell
        os.system("dot -Tpng attempt_1.dot > attempt_1.png")
```

```
attempt_1.pdf has been created
```

```
Out[3]: 0
```



PCUList

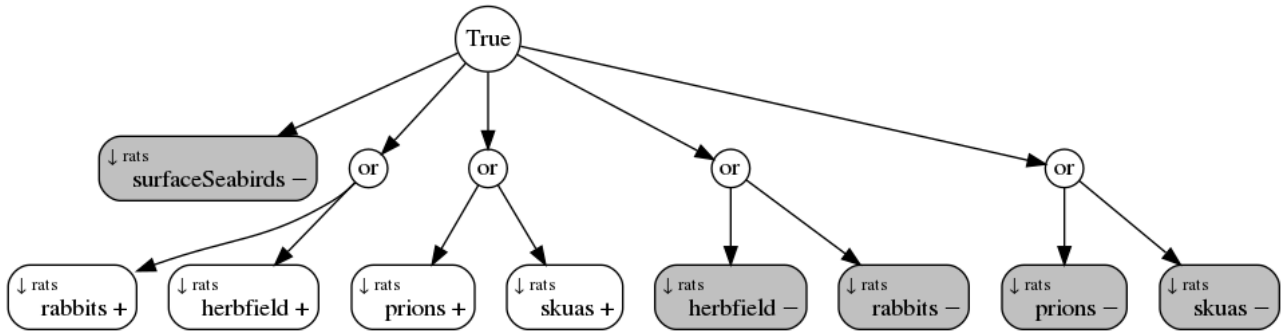
To simplify a network, a good first step is to split the PCUs up by length and present them in separate networks. Provided that every PCU is represented in the networks, then no information will be lost.

It is also a good idea to put highly-connected responses, and other pests, at the top as antecedents. We can spot highly-connected nodes by the number of arrows that are feeding into them, and placing other pest species at the top may be useful to decision-makers who are considering either flow-on effects on other pests, or a multi-species control programme.

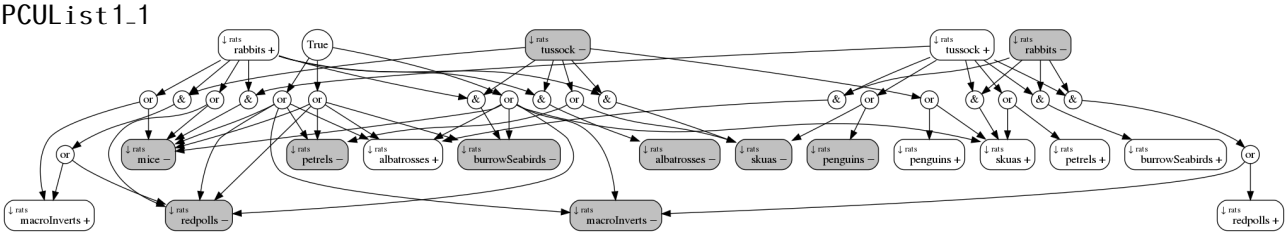
```
In [11]: PCUList1 = [ PCU for PCU in PCUList if len(PCU) <= 2 ]
         PCUList2 = [ PCU for PCU in PCUList if len(PCU) > 2 ]

         draw_implication_network2(PCUList1,
             [],
             'PCUList1_1', niceNames = None, controlSymbol = 'downarrow')
         draw_implication_network2(PCUList2,
             ['negrats_rabbits', 'posrats_rabbits', 'posrats_tussock', 'negrats_tussock'],
             'PCUList2_1', niceNames = None, controlSymbol = 'downarrow')

         os.system("dot -Tpng PCUList1_1.dot > PCUList1_1.png")
         os.system("dot -Tpng PCUList2_1.dot > PCUList2_1.png")
```

```
PCUList1_1.pdf has been created
PCUList2_1.pdf has been created
```

```
Out[11]: 0
```



PCUList1_1



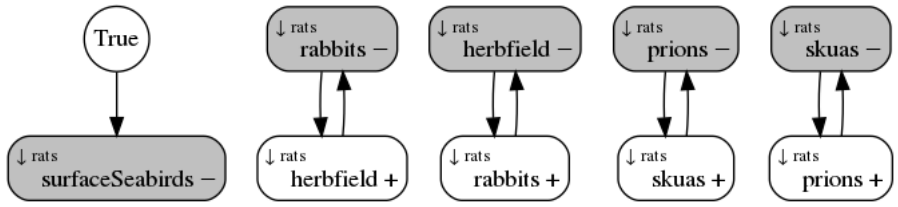
PCUList2_1

In the top network (PCUList1), we recognise the same bidirectional implication relationship that appeared previously, in the network representing rabbit control. Rabbits and herbfield are also in a bidirectional relationship. For short implications, the function draw_implication_network can be more useful. It will repeat the information contained in some of the PCUs (e.g. showing both bidirectional outcomes), but when the implications are short, this doesn't increase the complexity of the network much compared to the increase in clarity of the implications' meaning.

```
In [5]: draw_implication_network(PCUList1,
                                 'PCUList1_2',
                                 niceNames = None,
                                 controlSymbol = 'downarrow')
        os.system("dot -Tpng PCUList1_2.dot > PCUList1_2.png")
```

```
PCUList1_2.pdf has been created
```

```
Out[5]: 0
```



PCUList1_2

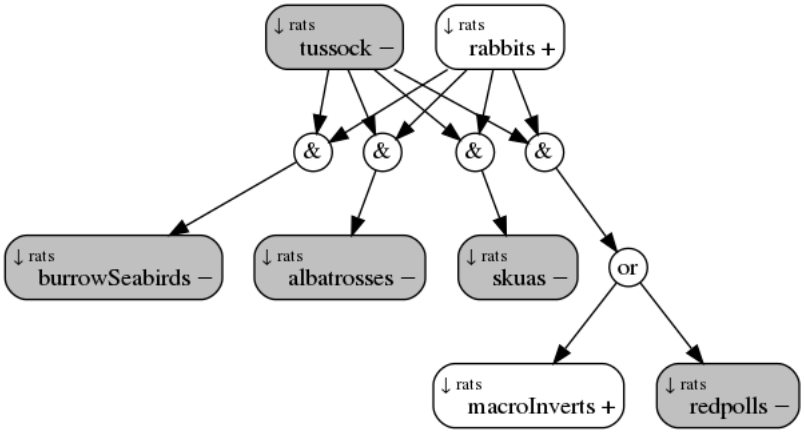
In the second network (PCUList2), we see that a symmetry in the effect of the combined response in rabbits and tussock. We can split those out into two separate networks and place the remainder of the PCUs in PCUList5 for further analysis.

```
In [6]: PCUList3 = list() # rat and tussock relationships
        PCUList4 = list()
        PCUList5 = list() # other
        for PCU in PCUList2:

            if 'posrats_rabbits' in PCU and 'negrats_tussock' in PCU:
                PCUList3.append(PCU)
            elif 'negrats_rabbits' in PCU and 'posrats_tussock' in PCU:
                PCUList4.append(PCU)
            else:
```

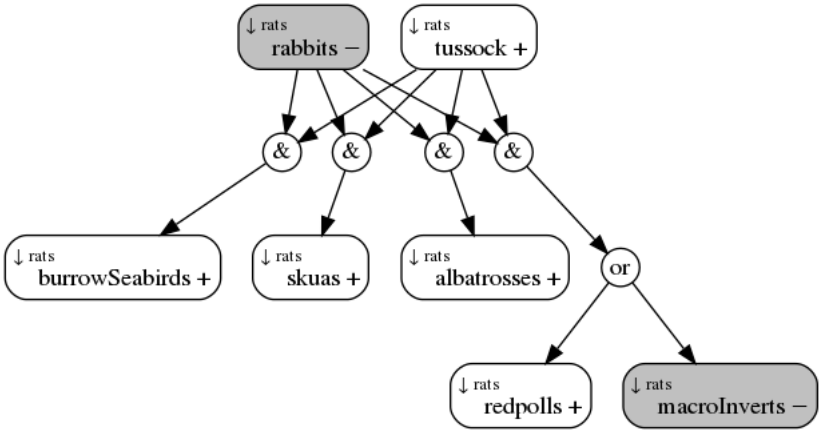
```
                PCUList5.append(PCU)

        draw_implication_network2(PCUList3,
                    ['posrats_rabbits', 'negrats_tussock'],
                    'PCUList3_1', niceNames = None, controlSymbol = 'downarrow')
        draw_implication_network2(PCUList4,
                    ['negrats_rabbits', 'posrats_tussock'],
                    'PCUList4_1', niceNames = None, controlSymbol = 'downarrow')

        os.system("dot -Tpng PCUList3_1.dot > PCUList3_1.png")
        os.system("dot -Tpng PCUList4_1.dot > PCUList4_1.png")
```

```
PCUList3_1.pdf has been created
PCUList4_1.pdf has been created
```

Out[6]: 0



PCUList3_1



PCUList4_1

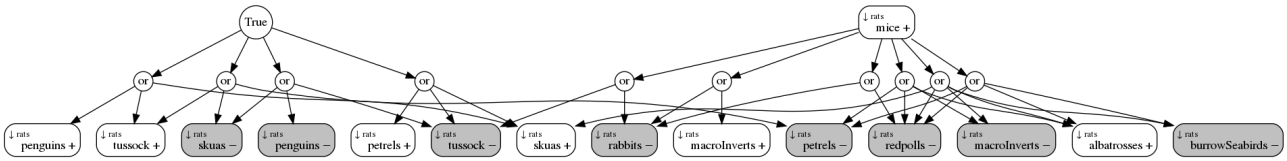
The symmetry between the two scenarios above may be useful for understanding the whole-system behaviour. For example, the response of burrow-nesting seabirds and albatrosses echoes what was found for rabbit control.

We now take a closer look at the PCUs that remain.

```
In [7]: draw_implication_network2(PCUList5,
                    ['posrats_mice'],
                    'PCUList5_1', niceNames = None, controlSymbol = 'downarrow')
        os.system("dot -Tpng PCUList5_1.dot > PCUList5_1.png")
```

```
PCUList5_1.pdf has been created
```

Out[7]: 0



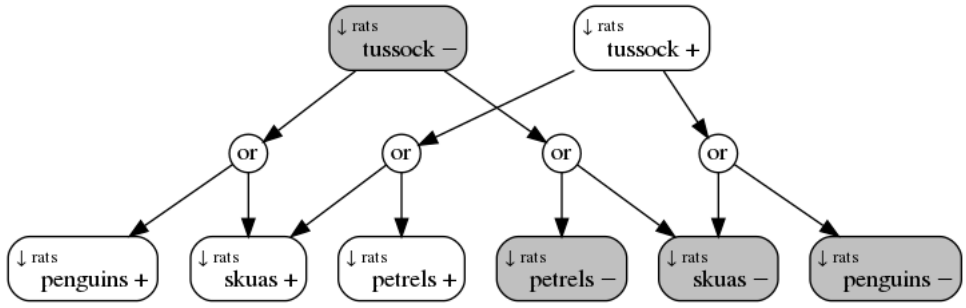
PCUList5_1

There is an obvious split between relationships involving mice and not. The relationships not involving mice have a symmetry, so we should choose an arrangement that emphasises that. Below, we choose to place the effect of rat control on tussock as the antecedent.

```
In [8]: PCUList6 = list() # for rules without mice
        PCUList7 = list() # for rules with mice

        for PCU in PCUList5:
            if 'posrats_mice' in PCU:
                PCUList7.append(PCU)
            else:
                PCUList6.append(PCU)

        draw_implication_network2(PCUList6,
                    ['posrats_tussock', 'negrats_tussock'],
                    'PCUList6_1', niceNames = None, controlSymbol = 'downarrow')
        os.system("dot -Tpng PCUList6_1.dot > PCUList6_1.png")
```

```
PCUList6_1.pdf has been created
```

Out[8]: 0



PCUList6_1

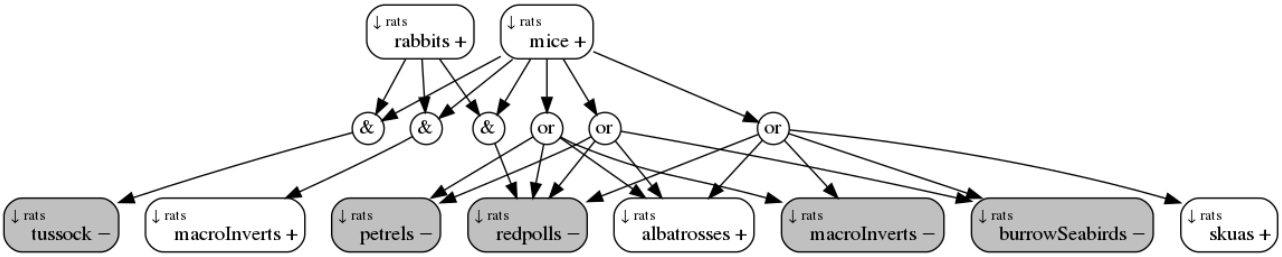
Now we consider the right-hand side, the relationships involving mice.

The importance of tussock suggests that the interaction between rabbits, mice, and tussock may be interesting in its own right. The simultaneous effect on rabbits is also involved in outcomes for macroinvertebrates and redpolls, and rabbits are another pest species, so we move the rabbit response to the antecedent.

```
In [9]: draw_implication_network2(PCUList7,
                    ['posrats_rabbits', 'posrats_mice'],
                    'PCUList7_1', niceNames = None, controlSymbol = 'downarrow')
        os.system("dot -Tpng PCUList7_1.dot > PCUList7_1.png")
```

```
PCUList7_1.pdf has been created
```

Out[9]: 0



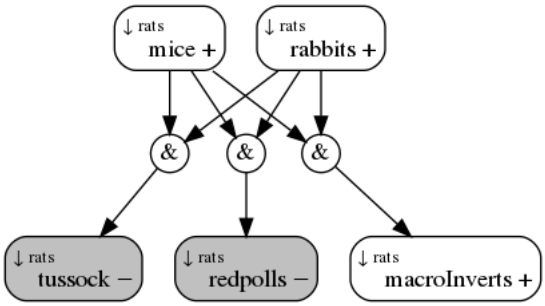
PCUList7_1

One could end the splitting of the remaining PCUs here. Alternatively, one could choose to emphasise the contingency upon the response of redpolls, by further splitting the network.

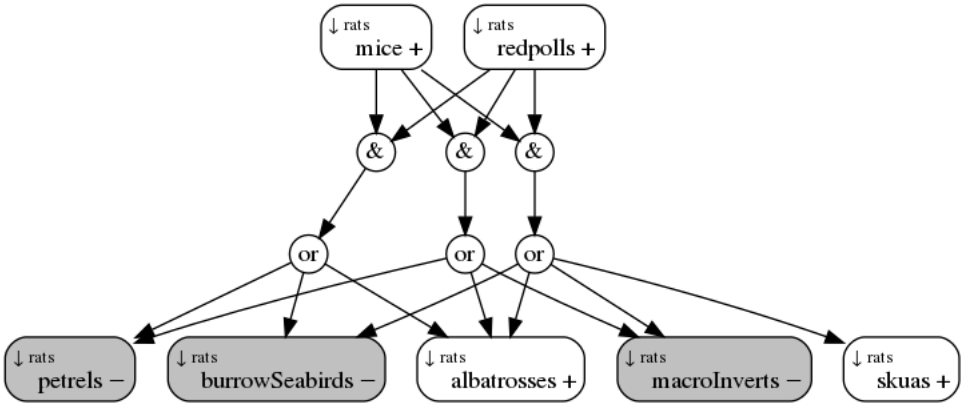
```
In [10]: PCUList8 = list() # for rabbit and mice relationships
         PCUList9 = list() # the remainder, which all involve redpolls

         for PCU in PCUList7:
             if 'posrats_rabbits' in PCU:
                 PCUList8.append(PCU)
             else:
                 PCUList9.append(PCU)

         draw_implication_network2(PCUList8,
                     ['posrats_rabbits', 'posrats_mice'],
                     'PCUList8_1', niceNames = None, controlSymbol = 'downarrow')
         os.system("dot -Tpng PCUList8_1.dot > PCUList8_1.png")

         draw_implication_network2(PCUList9,
                     ['posrats_redpolls', 'posrats_mice'],
                     'PCUList9_1', niceNames = None, controlSymbol = 'downarrow')
         os.system("dot -Tpng PCUList9_1.dot > PCUList9_1.png")
```

```
PCUList8_1.pdf has been created
PCUList9_1.pdf has been created
```

Out[10]: 0



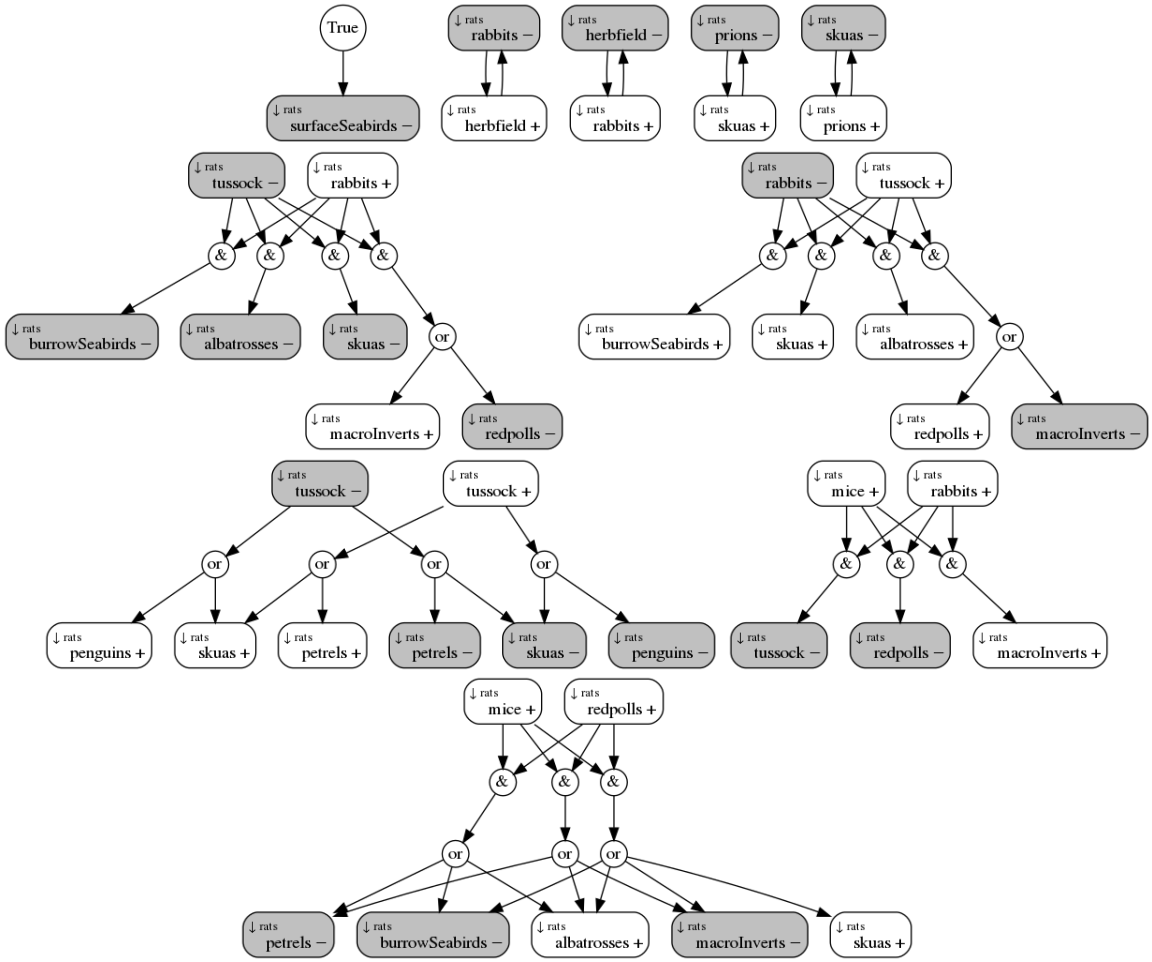
PCUList8_1



PCUList9_1

Potential final presentation

The presentation below combines all of the subnetworks that we split above. The decisions that were made to produce this network were somewhat arbitrary, and the best choice depends upon the needs of conservation decision-makers, and may also reflect our causal understanding of the relationships between species.



C.4 Applying the Boolean approach to an arbitrary model

The purpose of this tutorial is to show the Boolean analysis might be applied to an arbitrary model. The (fictitious) model concerns the social and environmental outcomes of river management.

This Jupyter Notebook is archived as:
Tutorial-4-Using_the_Boolean_approach_on_your_model.ipynb

Notebook

The function `your_model` returns all possible response-combinations from social-ecological model.

```
In [51]: from your_model import your_model

         responsesList, niceNames, collectedResponses = your_model()
```

The model concerns a set of management interventions, and their effects on 5 social and ecological response variables.

```
In [52]: # print
         for v in niceNames.values():
             print(v)
```

```
intervention
environmental flow
leisure use of river
water price
satisfaction with water authority
engagement of stakeholders
```

The response combinations have been encoded as follows:

```
In [53]: for k, v in niceNames.items():
             print(k + ': ' + v)
```

```
int: intervention
qua: environmental flow
lei: leisure use of river
pri: water price
sat: satisfaction with water authority
eng: engagement of stakeholders
```

And the response-combinations that were found in the model were returned as follows:

```
In [54]: # print header
         header = [ r[-3:] for r in responsesList ]
         print('  '.join(header))
         print(''.join(['-----']*len(header)))

         # print response combinations found
         for response in collectedResponses:
             print('  '.join(response))
```

```
eng  sat  pri  lei  qua
------------------------
neg  neg  neg  neg  neg
```

```
neg  neg  neg  neg  pos
neg  neg  neg  pos  pos
neg  neg  pos  neg  neg
neg  neg  pos  neg  pos
neg  neg  pos  pos  pos
neg  pos  neg  neg  neg
neg  pos  neg  neg  pos
neg  pos  neg  pos  pos
pos  neg  neg  neg  pos
pos  neg  neg  pos  pos
pos  pos  neg  neg  neg
pos  pos  neg  neg  pos
pos  pos  neg  pos  pos
pos  pos  pos  neg  pos
pos  pos  pos  pos  pos
```

The first step is to assign one of the responses to `True` and the other to `False`.

```
In [55]: str4true = 'pos'; str4false = 'neg'
```

Now we can treat the responses as Boolean variables. We use PyEDA to encode them as Boolean variables.

```
In [56]: from pyeda.inter import espresso_exprs
         from findpcu import getUnobservedInts, getRespvarList2BoolvarList
         from findpcu import intList2boolexpr, boolexpr2RespvalList

         x, x2s, r2idx = getRespvarList2BoolvarList(responsesList, str4true, str4false)

         # print the Boolean variable names
         x

Out[56]: [int_eng, int_sat, int_pri, int_lei, int_qua]

In [57]: type(x[0])

Out[57]: pyeda.boolalg.expr.Variable
```

We turn the list of observed responses into a list of unobserved responses (i.e. impossible response combinations), encoded as integers.

```
In [58]: observedInts = [ int(''.join(['1' if i in str4true else '0'
                                       for i in responseCombination]), 2)
                                       for responseCombination in collectedResponses ]
         observedInts
         unobservedInts = set(range(2**len(responsesList)))
         # ... and loop through the observed response combinations,
         # discarding those that were observed

         for i in observedInts:
             unobservedInts.discard(i)

         unobservedInts

Out[58]: {2, 6, 10, 12, 13, 14, 15, 16, 18, 20, 21, 22, 23, 26, 28, 30}
```

Using the Boolean variables above, we can create a Boolean expression from the unobserved responses.

```
In [59]: unobservedBoolexpr = intList2boolexpr(unobservedInts, x)

         # print the Boolean expression of unobserved responses
         unobservedBoolexpr
```

```
Out[59]: Or(And(~int_eng, ~int_sat, ~int_pri, int_lei, ~int_qua),
    And(~int_eng, ~int_sat, int_pri, int_lei, ~int_qua),
    And(~int_eng, int_sat, ~int_pri, int_lei, ~int_qua),
    And(~int_eng, int_sat, int_pri, ~int_lei, ~int_qua),
    And(~int_eng, int_sat, int_pri, ~int_lei, int_qua),
    And(~int_eng, int_sat, int_pri, int_lei, ~int_qua),
    And(~int_eng, int_sat, int_pri, int_lei, int_qua),
    And(int_eng, ~int_sat, ~int_pri, ~int_lei, ~int_qua),
    And(int_eng, ~int_sat, ~int_pri, int_lei, ~int_qua),
    And(int_eng, ~int_sat, int_pri, ~int_lei, ~int_qua),
    And(int_eng, ~int_sat, int_pri, ~int_lei, int_qua),
    And(int_eng, ~int_sat, int_pri, int_lei, ~int_qua),
    And(int_eng, ~int_sat, int_pri, int_lei, int_qua),
    And(int_eng, int_sat, ~int_pri, int_lei, ~int_qua),
    And(int_eng, int_sat, int_pri, ~int_lei, ~int_qua),
    And(int_eng, int_sat, int_pri, int_lei, ~int_qua))
```

The complexity of this expression can be reduced using Boolean minimisation, using the espresso algorithm from PyEDA

```
In [60]: boolExprMin, = espresso_exprs(unobservedBoolexpr)

         # print minimised Boolean expression
         boolExprMin
```

```
Out[60]: Or(And(~int_eng, int_sat, int_pri),
    And(int_sat, int_pri, ~int_qua),
    And(int_lei, ~int_qua),
    And(int_eng, ~int_sat, int_pri),
    And(int_eng, ~int_sat, ~int_qua))
```

A more human-readable form of the minimised Boolean expression can be obtained using boolexpr2RespvalList

```
In [61]: PCUList = boolexpr2RespvalList(boolExprMin, x2s)
         PCUList
```

```
Out[61]: [['negint_qua', 'posint_lei'],
          ['posint_eng', 'negint_qua', 'negint_sat'],
          ['posint_sat', 'posint_pri', 'negint_qua'],
          ['posint_sat', 'posint_pri', 'negint_eng'],
          ['posint_eng', 'posint_pri', 'negint_sat']]
```

The function draw_implication_network2 can be used to create an implication network. Here, we have specified that the effects of management interventions on community engagemend, environmental flow, and water price, should be antecedents in the network.

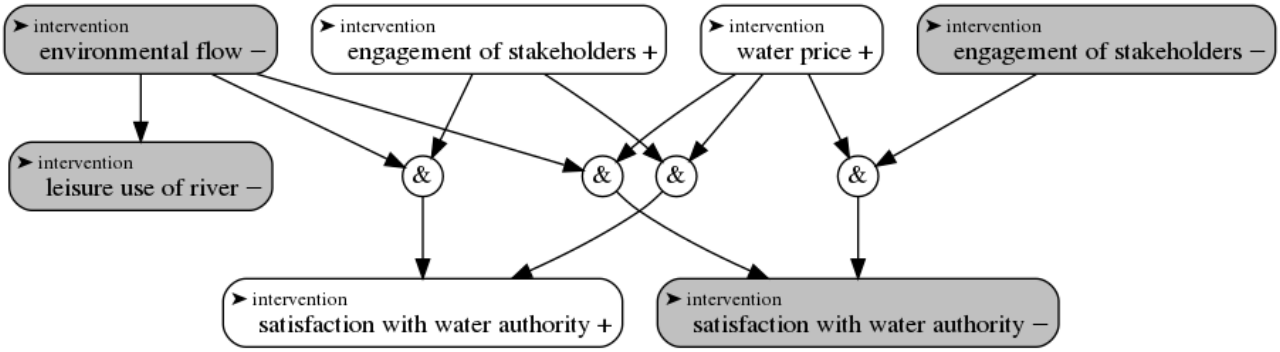
```
In [62]: draw_implication_network2(PCUList,
['posint_eng', 'negint_eng', 'negint_qua', 'posint_qua', 'posint_pri', 'negint_pri'],
'your_model', niceNames = niceNames, controlSymbol = '&#10148; ')
```

```
your_model.pdf has been created
```

A pdf of the implication network has been created. However we can also use graphviz to create figures in other formats.

```
In [63]: from IPython.display import Image
         import os

         os.system("dot -Tpng your_model.dot > your_model.png")
```

```
Out[63]: 0
```



An implication network for `your_model`.

D Interpreting the results of the Boolean approach

D.1 Boolean variables and logical implications

A Boolean variable is a variable having one of only two values, typically the truth values, True or False. For example, a Boolean variable 'raining' can may take the value: True, if it is raining; or False, if it is not.

Boolean variables can be linked by implications to create conditional statements. For example:

Statement 1: 'if it is raining then the streets are wet'

links Boolean variables 'raining' and 'wet streets' by an implication ('then'). This can be written as a logical implication statement:

$$\text{raining} \rightarrow \text{wet streets} \tag{D.20a}$$

where the right-arrow plays the role of the word 'then'.

It is worth noting that the converse of a true statement is not necessarily true, i.e. the direction of an implication cannot be 'reversed'. For example, it does not follow from Statement 1 that 'if the streets are wet then it is raining'; the streets might be wet for some other reason, e.g. because the street cleaner has come past.

The contrapositive of a true statement is also true. For example Statement D.20a can be rearranged to say:

Statement 1b: 'if the streets are not wet then it is not raining'

which can be written

$$\text{not wet streets} \rightarrow \text{not raining} \tag{D.20b}$$

where 'not' is an operator that negates the value of a Boolean variable, i.e. switches from False to True, and vice versa.

The example in Eq. D.20b also highlights a key difference between the implicit meaning of an "if … then …" in an English sentence and a logical implication: implications are statements about truth, not causality. Rain causes the streets to be wet. The absence of wet streets does not cause the absence of rain. But it is true that, if the streets are not wet, then it is not raining.

The operators 'and' and 'or', written as $\wedge$ and $\vee$ respectively, can be used to connect Boolean variables to create more complex conditional statements. For example:

Statement 2: 'if it is raining and I do not have my umbrella then I will have wet hair'

may be written

$$\text{raining} \wedge \text{not have umbrella} \rightarrow \text{wet hair} \tag{D.21a}$$

The rules of Boolean algebra can be used to rearrange the statement into other equivalent forms. For example, the logical implication in Eq. D.21a can be rearranged to

$$\text{not wet hair} \rightarrow \text{not raining} \vee \text{have umbrella} \tag{D.21b}$$

$$\text{not have umbrella} \wedge \text{not wet hair} \rightarrow \text{not raining} \tag{D.21c}$$

and so on. In general, an implication with k terms can be rearranged into $2^k - 2$ different implications.

A convenient way to summarise a set of equivalent implication statements is to write the combination of Boolean variables that is impossible by that statement. For example, Eqs. D.20a and D.20b can be rearranged

$$\text{raining} \wedge \text{not wet streets} \rightarrow \text{False} \tag{D.22}$$

and Eqs. D.21a to D.21c can be rearranged

$$\text{raining} \wedge \text{not have umbrella} \wedge \text{not wet hair} \rightarrow \text{False} \tag{D.23}$$

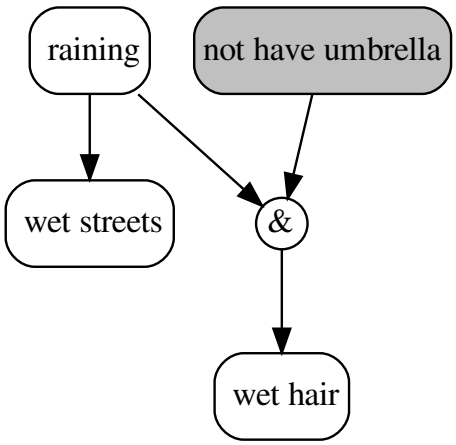


Figure D.6: A logical implication network that can be formed by combining the logical implications in Eq. D.20a and D.21a.

The combinations of Boolean variables on the left-hand sides of Eqs. D.22 and D.23 are impossible combinations. These impossible combinations are analogous to the impossible species-response combinations, discussed in the body of the text, that form the PCUs.

In the body of the text, we have also discussed the logical implication network, which chains logical implications together by their shared terms. Fig. D.6 shows one possible logical implication network that results from combining the examples above.

D.2 Species responses as Boolean variables and model predictions as logical implications

In the case study in the body of the text, a particular species A's response to a pest control program can take one of two values: positive, which we denote by $A+$; or negative, which we denote by $A-$. Because the species responses are two-valued, we can treat them as a Boolean variable, by mapping the positive response to True, and mapping the negative response to False. By doing so, we are then able to harness the machinery of Boolean algebra, in order to analyse and summarise the species responses in the ecological model as a whole.

The goal of the Boolean approach is to find conditional statements that link the model's predictions about species responses together. For example, if through model simulations we find that:

'if A responds positively then B responds positively'

then that can be written as a logical implication statement

$$A+ \rightarrow B+ \tag{D.24}$$

In contrast to the probabilistic QM approach, implication rules like Eq. D.24 are deterministic. If we say that Eq. D.24 is true in the model, then we are saying that the model always predicts that a positive response in species A will be accompanied by a positive response in species B. In the case study, the structure of the network of interactions between species is given, however the interaction strength values are unknown. Therefore Eq. D.24 can be more specifically interpreted as:

'in the model, if A responds positively then B responds positively, regardless of the interaction-strength values'

D.3 Logical implication networks from ecological models

In the following examples, we will assume that we have simulated models of pest suppression, similar the the Macquarie Island case study, and used the Boolean approach to analyse the results. We

will assume that we have modelled the suppression of a pest species, P, and that we are interested in the responses of the other species in the network: A, B, C, etc.

The simplest possible outcome is that a single-species response is impossible in the model, regardless of what the other responses are. For example, if the model never predicts a negative response in A

$$A- \rightarrow \text{False} \tag{D.25}$$

then this can be represented by the implication network in Fig. D.7a. Because $A-$ is a Boolean variable, the logical implication can also be rearranged

$$\text{True} \rightarrow A+ \tag{D.26}$$

resulting in the implication network in Fig. D.7b. Fig. D.7b would be the more usual way of presenting the result, because it gives the result in terms of what the model predicts will happen to a species.

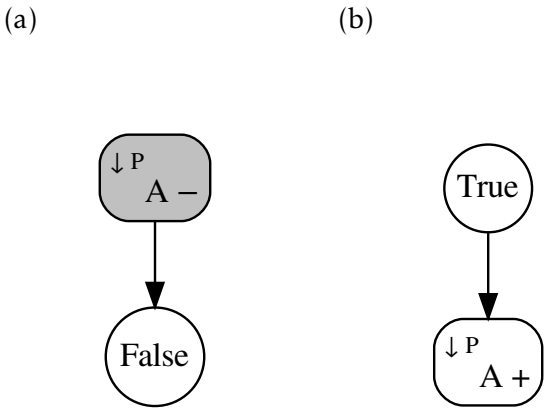


Figure D.7: Two logical implication networks corresponding to the model prediction that species A will never respond negatively to the suppression of pest-species P. A negative response in A is abbreviated 'A–', a positive response in A is abbreviated 'A+', and the '↓P' in the top left-hand corner of each node reminds us that the context is species responses to the suppression of P.

The kind of single-species response scenario that is shown in Fig. D.7 is easily revealed by existing QM methods. A weighted-predictions matrix analysis would result in $W_{AP} = 1$. A Monte Carlo simulation will return a 100% probability that A responds positively. The advantage of the Boolean approach is that it can analyse more complex scenarios, where a species response is contingent upon the responses in other species.

The simplest contingent case involves two species connected by a unidirectional implication. For example, the logical implication network in Fig. D.8a contains one logical implication

$$A+ \rightarrow B- \tag{D.27a}$$

which reads as:

'if A responds positively then B responds negatively'

The logical implication in Eq. D.27a can also be rearranged to obtain its contrapositive:

$$B+ \rightarrow A- \tag{D.27b}$$

which reads as:

'if B responds positively then A responds negatively'

and is shown in Fig. D.8b.

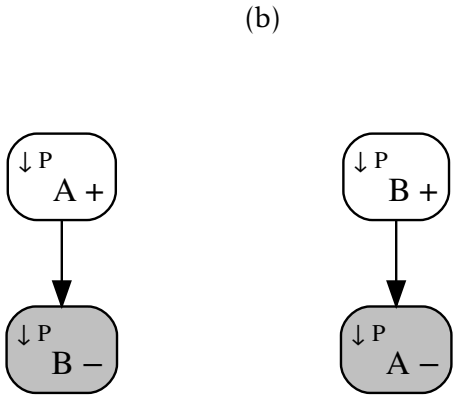


Figure D.8: Logical implication networks corresponding to the logical implication in Eq. D.27a and Eq. D.27b.

Eq. D.27a and Eq. D.27b are equivalent, and are derived from a single PCU:

$$A+ \wedge B+ \rightarrow \text{False} \tag{D.27c}$$

In other words, the results from our model simulations never predict a simultaneously positive response in A and a positive response in B.

It is important to note that the direction of an implication cannot simply be reversed. For example, from Fig. D.8a, we cannot say *'if B responds negatively then A responds positively'*. In the full truth table corresponding to the PCU in Eq. D.27c (Table D.7), when B responds negatively then either a positive or a negative response in A is possible. The only valid way to reverse the order of the variables in the implication is to rearrange it using Boolean algebra; in this case, to take the contrapositive, shown in Fig. D.8b.

Table D.7: The truth table corresponding to Eq. D.27c, with arrows highlighting the rows corresponding to the situation in which B responds negatively. When B responds negatively, either a positive or a negative response in species A is possible.

Suppression of P causes positive response in:		Species-response combination impossible in the model?	
A	B		
False	False	False	←
False	True	False	
True	False	False	←
True	True	True	

Rearranging the implications can be useful for obtaining statements that are more practical from a management perspective. For example, the implication shown in Fig. D.9a,

$$A+ \rightarrow B- \vee C- \tag{D.28}$$

can be rearranged into 5 other equivalent logical implications, with corresponding implication networks Fig. D.9b-f. The choice of which network to present to stakeholders depends upon practical considerations. For example, if species A and B are of particular conservation concern, then Fig. D.9a could be rearranged to produce Fig. D.9c, which frames the result in terms of impacts on A and B. As another example, if monitoring species B and C is particularly easy, then Fig. D.9e may be useful to managers in the design of a post-intervention monitoring programme, to provide early warning about potential deleterious effects on the more obscure species A.

It is important to note that the 'or' operator in an implication network is inclusive. Inclusive

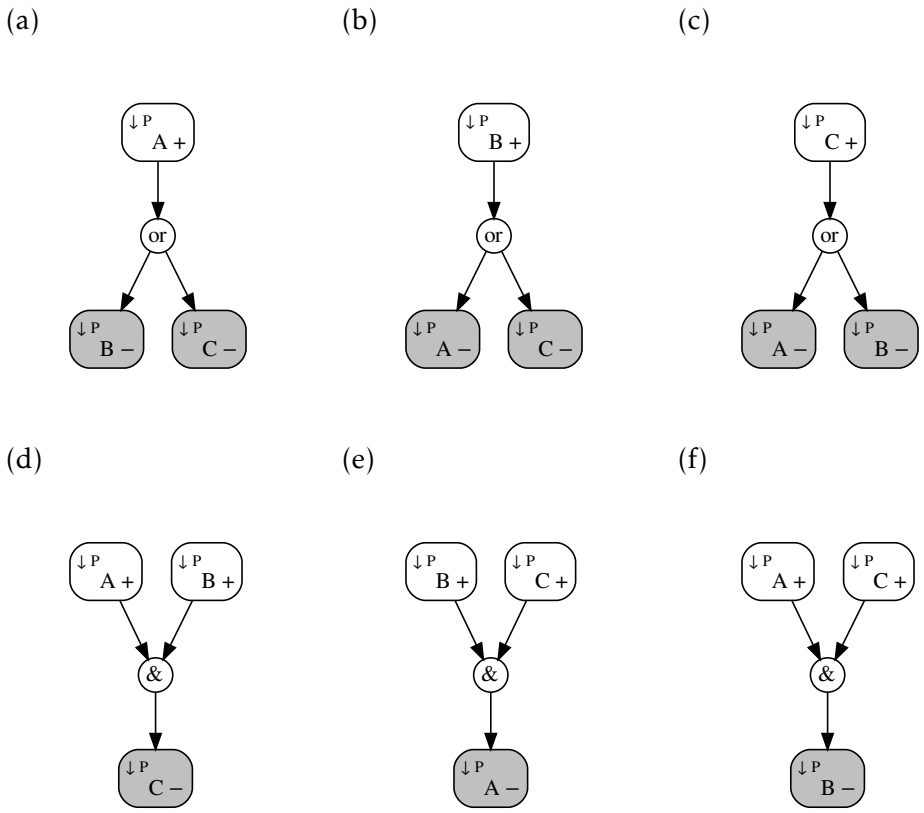


Figure D.9: The logical implication network corresponding to Eq. D.28 (a), and five equivalent networks that can be derived from it (b-f).

means that only one of the consequents has to occur, but more than one may also occur. For example, in Fig. D.9a, a positive response in A predicts a negative response in B or a negative response in C. Therefore the following three outcomes could occur: (1) $B-$ and $C+$, (2) $B+$ and $C-$, or (2) $B-$ and $C-$.

An 'and' operator requires all antecedents to be satisfied in order to *guarantee* the consequent. For example, in Fig. D.9d, the model only guarantees a negative response in species C if there is a positive response in both A and B. It is *possible* that species C might respond negatively if only A or only B is positive (see Fig. D.9a-b), however it is not guaranteed.

The absence of a species from the implication network indicates that any response in that species is possible, and that which response occurs cannot be predicted from the other species responses. For example, if the result shown in Fig. D.9 was obtained for a four-species ecosystem, with species A, …, D, then any response in species D is possible.

For more complicated implication networks, the basic interpretation depends upon interpreting each of the implications that appear in the subnetworks within it (as above). Subnetworks can be identified by finding species responses that are joined by a direct implication, or joined to the same 'and' or 'or' node and by implications. For example, in the implication network in Fig. D.10, each of the subnetworks (circled in a different colour) corresponds to a PCU as follows:

$$A+ \longrightarrow \text{False} \tag{D.29a}$$

$$B- \longrightarrow \text{False} \tag{D.29b}$$

$$C+ \wedge D- \longrightarrow \text{False} \tag{D.29c}$$

$$C+ \wedge E+ \wedge F- \rightarrow \text{False} \tag{D.29d}$$

$$C+ \wedge G- \wedge H- \rightarrow \text{False} \tag{D.29e}$$

$$G- \wedge I+ \wedge J- \wedge K+ \rightarrow \text{False} \tag{D.29f}$$

For large implication networks, it may take some work to separate and identify useful subnetworks. The choices one makes depend heavily upon the conservation decision-making context, and the tutorial in SI C.3 provides an example of how these decisions may be made.

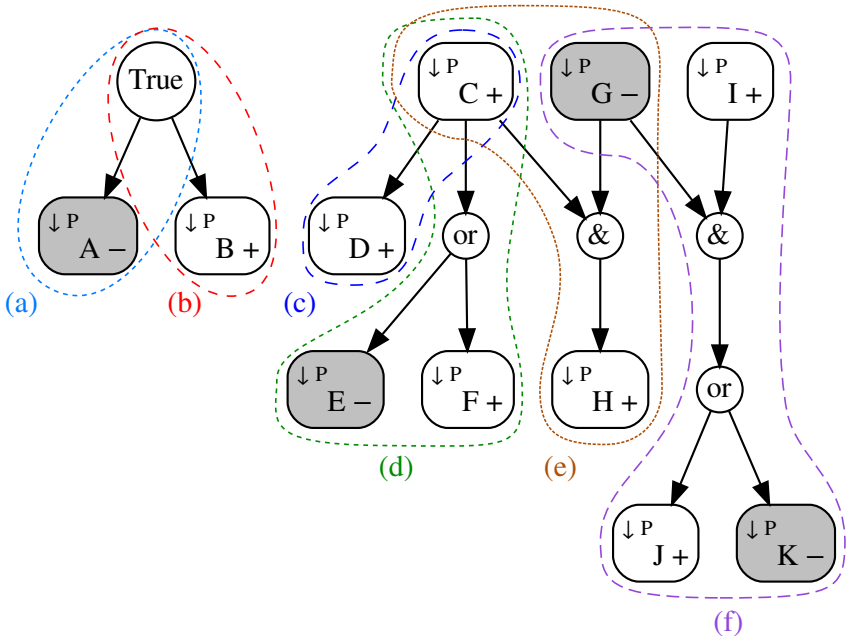


Figure D.10: A logical implication network, highlighting the subnetworks corresponding to each implication derived from the PCUs in Eqs. D.29a to D.29f.

In more complicated implication networks, information can also be gleaned from the relationships between subnetworks. Disconnected subnetworks indicate that the species within those subnetworks respond together in some way, and can often be related to the appearance of tightly connected subnetworks in the original species interaction network. However, the implication network can also identify relationships between species responses that are not obvious from analysing original interaction network itself; this reflects the complex-systems nature of ecosystem behaviour, which results from direct and indirect feedbacks between species.

Species responses that appear in multiple subnetworks may indicate key species, or determinants of the system's behaviour (e.g. $C+$ in Fig. D.10). This may be interesting from a theoretical perspective, to identify important species in the system, or from a practical perspective, if those species are candidates for a post-control monitoring programme. However it is important to remember that implication relationships are statements about truth, not causality. Again using the example in Fig. D.10, one cannot say that a positive response in C 'causes' a positive response in D; the cause of their relationship involves all of the direct and indirect feedbacks between those two species and other species in the system.

A particularly interesting case, from a pest-control perspective, is when another pest's response is highly connected in the logical implication network. For example, in Fig D.11, the mesopredator M's response to the control programme has a key determining effect upon outcomes on other species. This situation suggests that a multi-species control programme, where both the pest P and the mesopredator M are suppressed, could have beneficial outcomes. A modeller should then per-

form further simulations, to determine what the model predicts that the outcomes of a multi-species control programme will be.

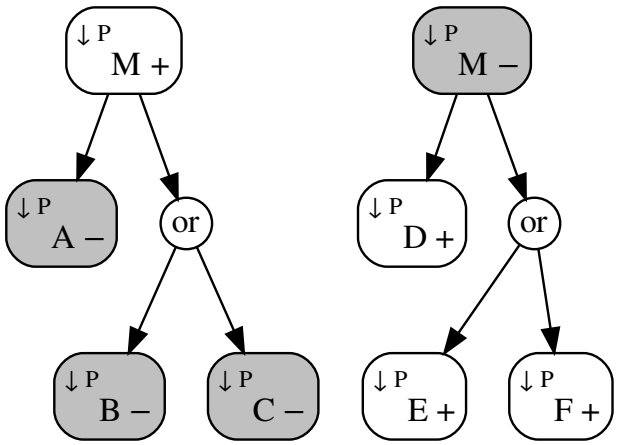


Figure D.11: A logical implication network where a mesopredator species M has a key determining effect upon species outcomes.

The difficulty of interpreting an implication network increases with the number of species involved in each subnetwork. For example, in Fig. D.12, a positive response in A does not give us much information about what will happen to the other species. Because the 'or' node is inclusive, a positive response in A may be accompanied by the response shown for: B only; B and C only; B, C, and D only; etc. While it is difficult to interpret such a result in terms of predictions, it has some use in that it indicates that the species responses are highly (though not completely) indeterminate.

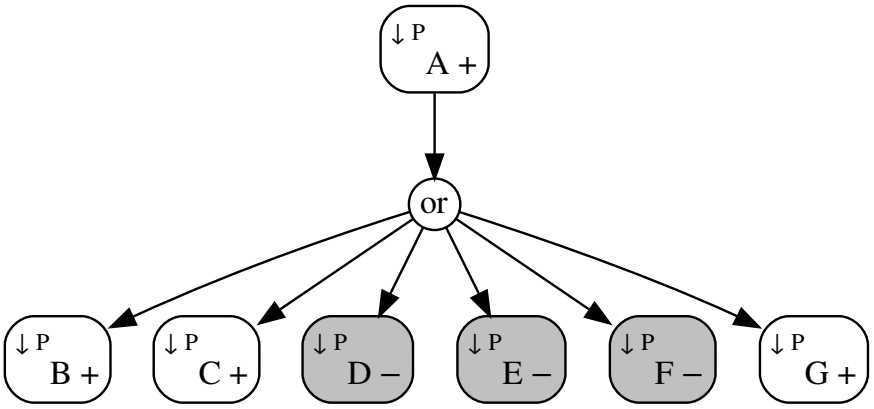


Figure D.12: A logical implication subnetwork containing many species responses reflects indeterminacy in those species responses.

E Additional results

The sampling method influenced how easy it was to obtain a community matrix that passed the plausibility constraints (Fig. E.13). Increasing the strength of the stability constraint increased the difficulty obtaining a plausible community matrix. When the sampling was performed on the community matrix parameters (Raymond baseline), approximately 0.5 matrices were rejected before a stable matrix was found. In contrast, when sampling on the Lotka-Volterra parameters (test 1) an average of approximately 950,000 matrices were rejected before the plausibility constraints were met (cf. Christianou and Kokkoris, 2008).

Additional species-response predictions, that were not included in the body of the text, are found in Fig. E.14 and E.15.

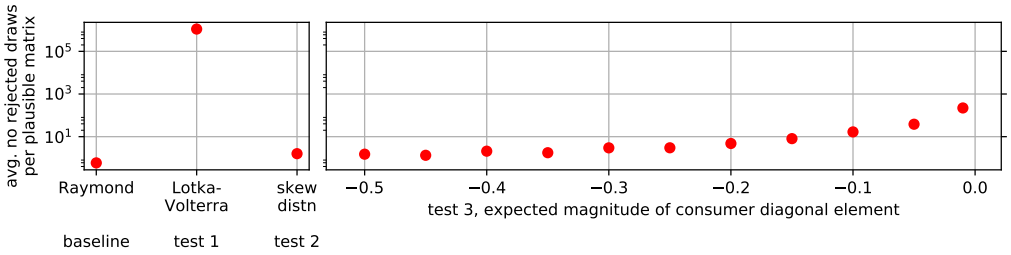


Figure E.13: Additional results from the different Monte Carlo simulation methods that were applied to the Macquarie Island case study.

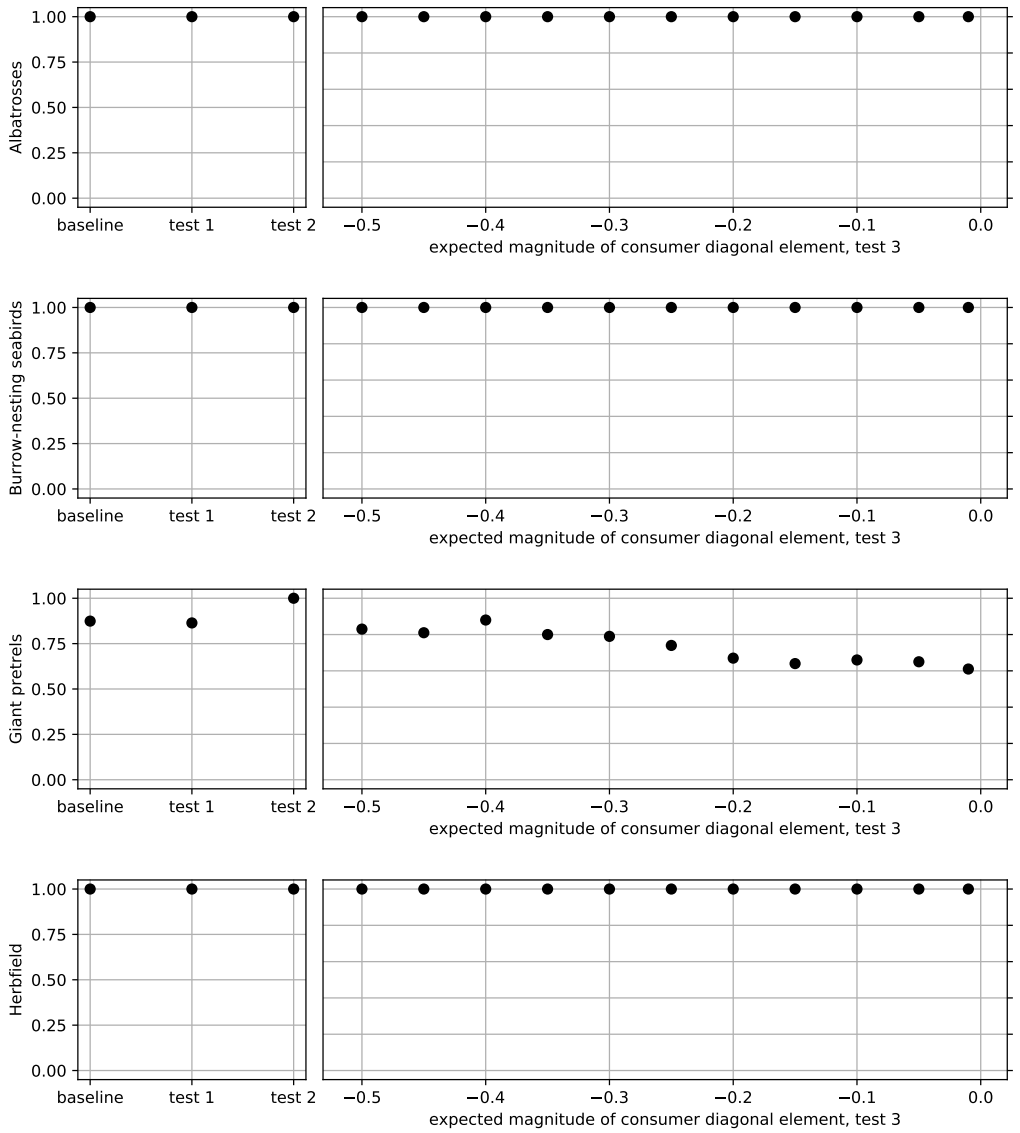


Figure E.14: Additional results from the different Monte Carlo simulation methods that were applied to the Macquarie Island case study.

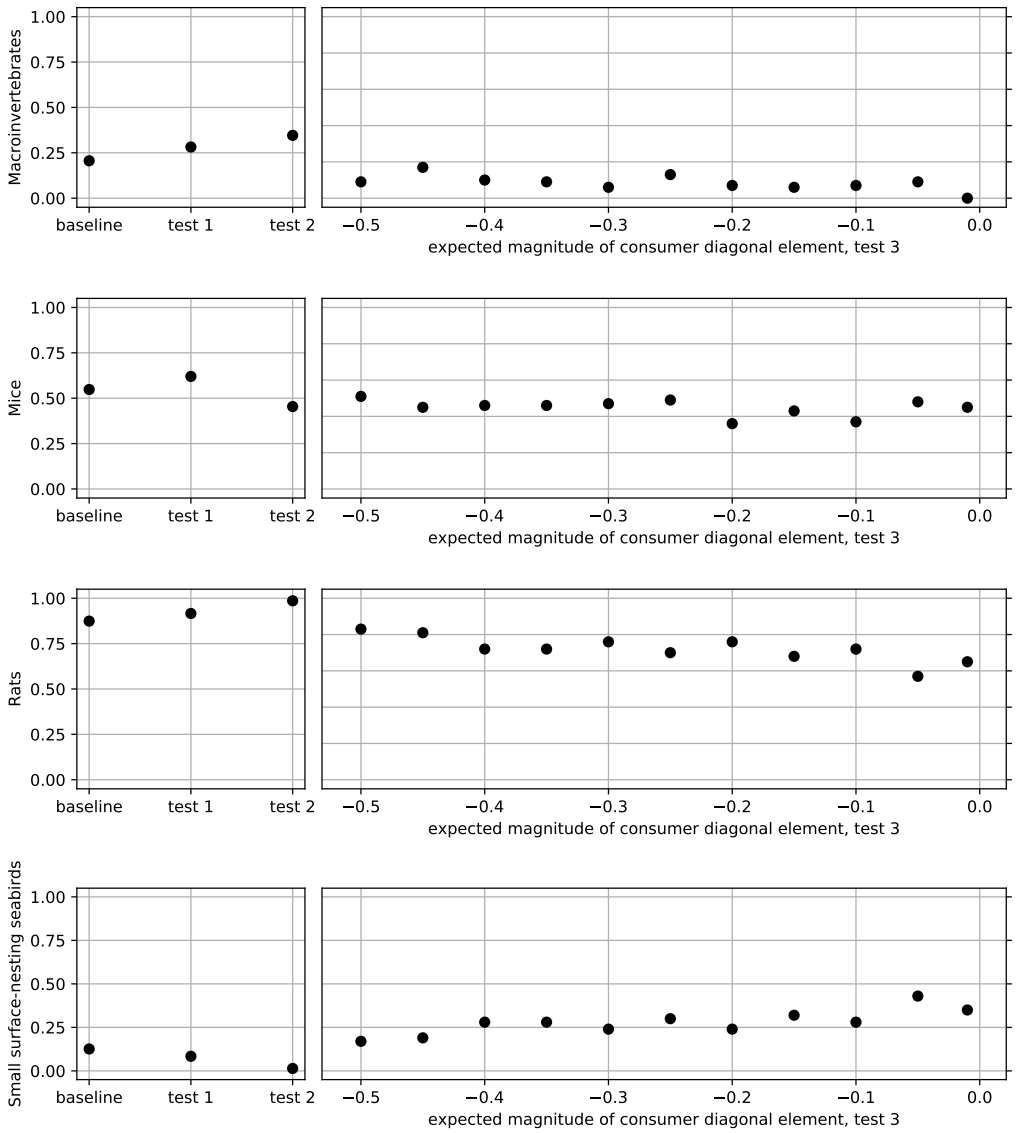


Figure E.15: Additional results from the different Monte Carlo simulation methods that were applied to the Macquarie Island case study.

F Multi-species press-perturbation

The response of a species to a multi-species pest-control was modelled in Raymond et al. (2011) by summing the corresponding elements in the sensitivity matrix (see also Ramos-Jiliberto et al., 2012; Harvey et al., 2016). In this appendix, we show that this implicitly assumes that the magnitude of the perturbation on each pest is equal.

In the case of a single-species perturbation, the effect on species i of a press-perturbation of species j is found by rearranging Eq. 12 of Nakajima (1992):

$$dx_i = s_{i,j} dz_j \tag{F.30}$$

where dx_i is the change in the steady-state population size of species i or the focal species' response, dz_j is the rate in inflow or removal of species j, and $s_{i,j}$ is the corresponding element of the sensitivity matrix. In this case, the sign of the species response dx_i in Eq. F.30 can be determined without knowing the magnitude of the perturbation (assumed small).

In the case of a multi-species perturbation, the effect on species i of simultaneous press-perturbation of a set of species $j \in P$ is found similar to Eq. 15 in Nakajima (1992):

$$dx_i = \sum_{j \in P} s_{i,j} dz_j. \tag{F.31}$$

In this case, if there are a mix of signs of elements $s_{i,j}$ in the sum in Eq. F.31, then the sign of the species response dx_i cannot be determined without knowing the relative magnitudes of the perturbations dz_j.

When Raymond et al. (2011) summed the elements of the sensitivity matrix to obtain the sign of the species' responses to a multi-species press-perturbation, they were effectively assuming that all dz_j in Eq. F.31 were equal in magnitude. Alternatively, they may have been invoking an implicit assumption about the probability distribution of the relative magnitudes of dz_j.