Supplemental Materials to

Parametric Order Constraints in Multinomial Processing Tree Models: An Extension of Knapp and Batchelder (2004)

Henrik Singmann

Karl Christoph Klauer

David Kellen

Albert-Ludwigs-Universität Freiburg

Author Note

Henrik Singmann, Karl Christoph Klauer, and David Kellen, Institut für Psychologie, Albert-Ludwigs-Universität Freiburg, Freiburg, Germany.

For questions regarding this heuristic contact singmann@gmail.com

A Heuristic for Creating MPT Models of Order Constraints

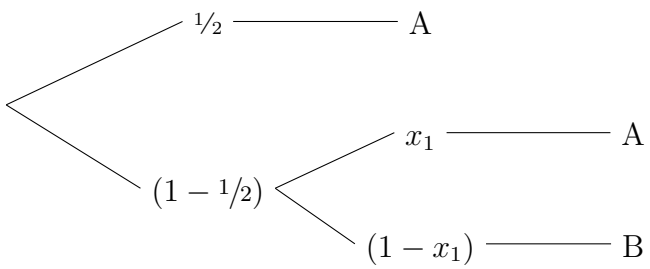
Introduction

This document provides a heuristic for constructing binary MPT models for linear order constraints such as those presented in Figure 4 of the main document. The heuristic is based on implementing the vertices of the convex polytope in a different way than described in the main document to represent more complex order constraints than those represented in Figure 4. However, whereas it can be shown that this heuristic preserves the order constraints, we cannot prove for all patterns and their extensions that it also allows to represent any set of probabilities, although we believe this to be the case. To circumvent that problem we additionally performed some numerical tests on the parameterizations presented here and in Table 2 of the main document which are presented at the end of this document.

Purely Linear Order Constraints

Linear orders of the form $A \geq B \geq C$ for MPT models (e.g., Figure 4, pattern I) can be created by two different methods, either *downward* (i.e., starting from the largest element, in this case A) or *upward* (i.e., starting with the smallest element, in this case C). Both methods are recursive algorithms which need to be successively applied to the elements in the order and direction given. Furthermore, both methods require to consider certain vertices of the polytopes, to determine the minimum or maximum probability each element can have. In the case of purely linear order constraints the main vertices we need to consider are minima of the largest and and maxima of the smallest element which are identical and given by equal probabilities for all elements. For the example with three elements this vertex is ⅓ for each element and in the case of only two elements, $A \geq B$, ½. Note that these vertices are indeed the minimum for the largest element (i.e., the former inequality cannot be satisfied if $A < 1/3$ or the latter not if $A < 1/2$) and the maximum for the smallest element (i.e., the former inequality cannot be satisfied if $C > 1/3$ and the latter not if $B > 1/2$).

Downward Method. The downward method consists of a set of steps which need to be applied successively from the largest to smallest element: (1) assign the minimum amount required to fulfill the (current) inequality to the current element, (2) distribute a freely estimated amount of the remainder in equal proportions to the current and all larger elements, (3) distribute the remainder of step (2) to the next element by recursively applying steps (1) and (2) or if the last element is reached, assign the remainder to the last element. For the case of the simplest inequality with only two elements $A \geq B$ this leads to the following tree and probabilities for A and B (note that for referencing purposes all trees in this document are consecutively numbered):



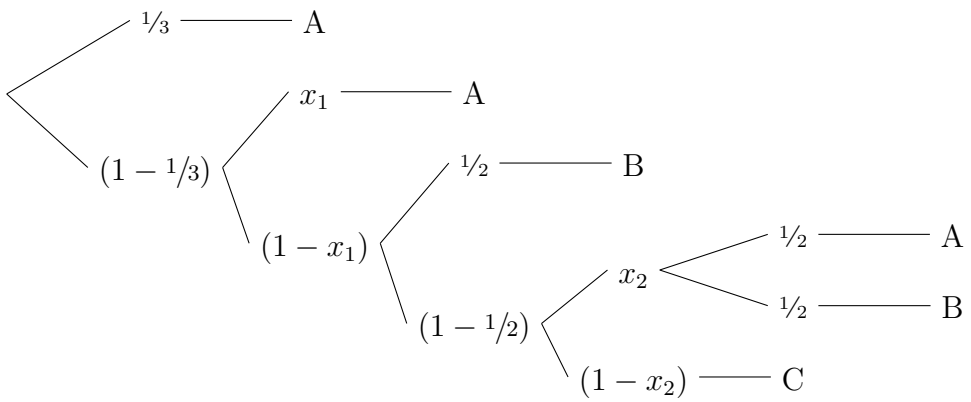
(1)

$$P(A) = ½ + (1 - ½)x_1$$

$$P(B) = (1 - ½)(1 - x_1)$$

For the case of three elements $A \geq B \geq C$ the obtained tree is somewhat more complicated given the restrictions in step (2) to distribute the remainder equally among the current and all larger elements. Furthermore, we need to make sure to correctly adjust the constant used to distribute the minimum amount in step (1) (i.e., ⅓ to A, but ½ to B given that in step (1) for B the remaining inequality is only $B \geq C$ and the

relevant vertex consequently ½):



(2)

$$P(A) = ⅓ + (1 - ⅓)x_1 + (1 - ⅓)(1 - x_1)(1 - ½)x_2½$$

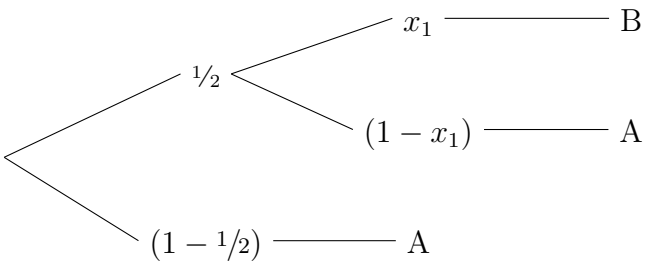
$$P(B) = (1 - ⅓)(1 - x_1)½ + (1 - ⅓)(1 - x_1)(1 - ½)x2(1 - ½)$$

$$P(C) = (1 - ⅓)(1 - x_1)(1 - ½)(1 - x2)$$

It can be shown that the downward method is equivalent to the vertex mixture method described in the main manuscript and hence we can be sure that it both imposes the order constraint and allows all ordered set of probabilities to be represented. For illustration purposes consider that step (1) makes sure that none of the elements can receive less probability than the boundary conditions defined above. For the second example $P(A) \geq ⅓$. Furthermore, step (2) allows that each element can be of a freely estimated amount larger than the minimum (e.g., $P(A)$ can be 1) while still maintaining that the inequality holds (e.g., if $x_1 = 0$ and $x_2 = 1$, $P(A) = P(B) = ½$. In other words, step (2) allows that any ordered set can be represented, for example if $x_1 > 0$, $A > B$. Note that this method can easily extended for order constraints with more than 3 elements by employing the correct constant in step (1) which corresponds to the equal probability for all remaining elements and extending the model accordingly.

Upward Method. The *upward* method consists of a number of steps which needs to be successively applied to all elements starting with the smallest one: (1) take the maximum amount which can be applied to the current element and (2) assign a freely estimated amount of this proportion to the current element, (3) distribute the

remainder from (1) equally among all superordinate elements and assign each element the same freely estimated amount of (2), (4) apply steps (1), (2), and (3) successively to the next elements and add the resulting tree to the remainders of steps (2) and (3) or if the last element is reached assign the remainder of steps (2) and (3) to the last element. In other words, one takes the maximum amount allowed for the current element, and assigns a freely estimated amount of this proportion to this element and the same amount to the superordinate elements (thereby fulfilling the order constraint) then recursively applies this approach to the remainders leading to the fact that both remainders contain the same subtrees. For the case of the simplest inequality with two elements $A \geq B$ this leads to:



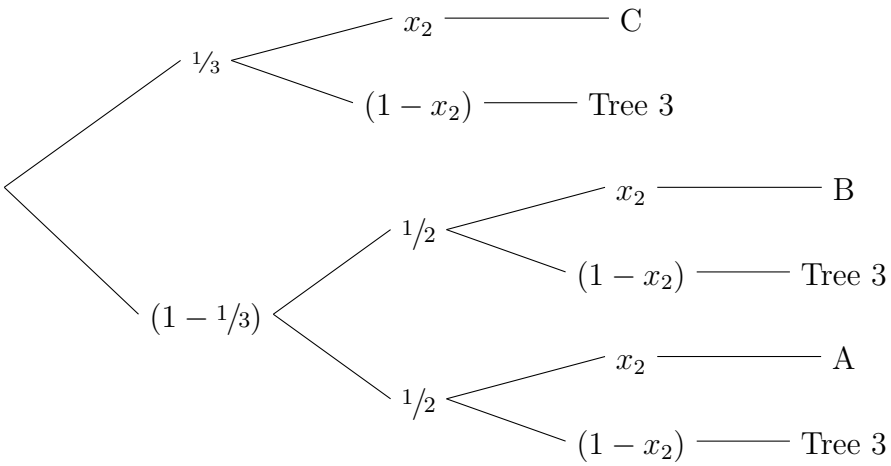
(3)

$$P(A) = ½(1 - x_1) + (1 - ½)$$

$$P(B) = ½\, x_1$$

For the case of three elements $A \geq B \geq C$ the obtained tree is considerably larger given that the complete subtree for $A \geq B$ (corresponding to tree (3) above) exists thrice, once for the remainder of the first step (2) and twice for the remainder of the first step (3). Furthermore, we again need to make sure to correctly adjust the constant

used to distribute the maximum amount in step (1) (i.e., ⅓ to C, but ½ to B):



(4)

$$
\begin{aligned}
P(A) = {} & ⅓(1 - x_2)½(1 - x_1) + ⅓(1 - x_2)(1 - ½) + \\
& (1 - ⅓)½\, x_2 + \\
& (1 - ⅓)½(1 - x_2)½(1 - x_1) + (1 - ⅓)½(1 - x_2)(1 - ½) + \\
& (1 - ⅓)½(1 - x_2)½(1 - x_1) + (1 - ⅓)½(1 - x_2)(1 - ½) \\
P(B) = {} & ⅓(1 - x_2)½\, x_1 + \\
& (1 - ⅓)½\, x_2 + \\
& (1 - ⅓)½(1 - x_2)½\, x_1 + \\
& (1 - ⅓)½(1 - x_2)½\, x_1 \\
P(C) = {} & ⅓\, x_2
\end{aligned}
$$

As was the case for the downward method, the upward method is equivalent to the vertex mixture method and consequently allows one to both impose the order constraint and to represent all ordered sets of probabilities. Steps (1) to (3) are crucial in controlling that the order constraint is not violated. At most, each element receives the maximum proportion it is allowed to have, for example, $P(C) \leq 1/3$, $P(B) \leq ⅓ \times ½ + ⅔ \times ½ \leq ½$ (controlled by step (1)). To make sure that no

superordinate element is smaller than a subordinate, step (3) assigns all superordinate elements the proportions assigned to subordinate elements. To see that all possible sets of probabilities are represented one needs to jointly consider steps (2) and (4). Step (2) allows that of the maximum amount for each element, the amount the element takes is completely free (i.e., can take any allowed proportion). Furthermore, step (4) makes sure that in case an element receives no probability, the remaining elements distribute the complete remaining probability mass according to the constraint (e.g., if $x_2 = 0$ and $x_1 = 1$, $P(A) = P(B) = 1/2$). Again, this method can easily be extended for order constraint with more elements by calculating the correct constant. However, this entails the cost that for each element added, the size of the tree rapidly grows as the existing tree is copied multiple times.

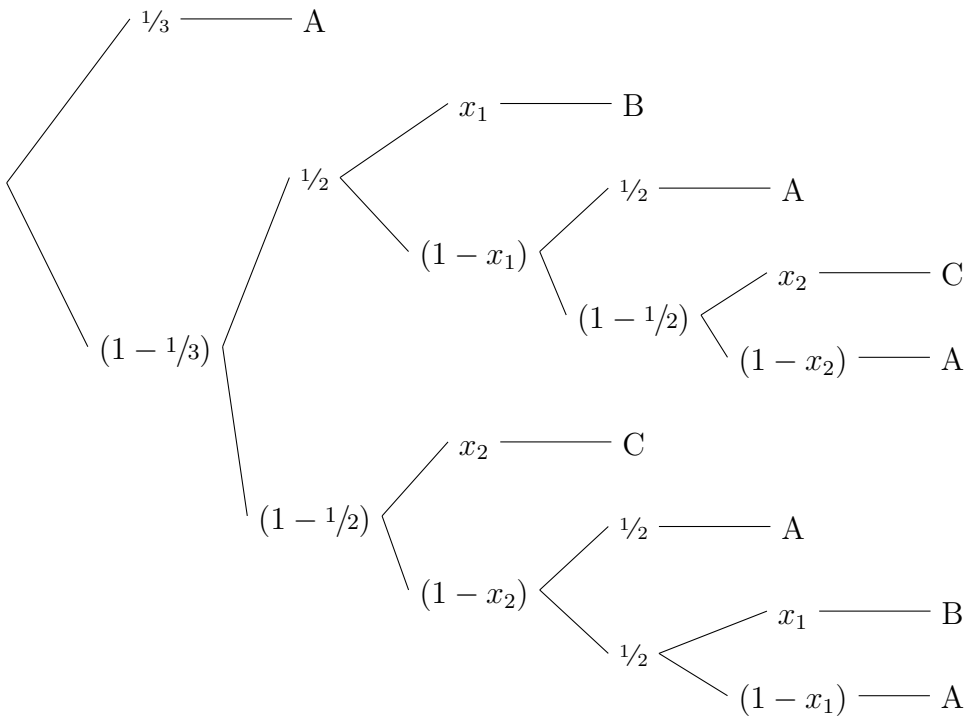
Representing more Complex Constraints. We have seen that both methods lead to equivalent but different parametrization of linear order constraints which begs the question why we introduced both methods. The reason for this is that both methods form the building blocks for different types of order constraints. As discussed in the main document, for some of the patterns of constraints presented in Figure 4, the vertex mixture representation is overparameterized, that is there are more vertices than independent data points. The heuristic described here can help in finding a binary MPT representation for order constraints that is not overparameterized. More specifically, the downward method can be used for cases in which the order constraint forms a fork-like structure (e.g., $A \geq B$ and $A \geq C$ as in patterns V, VI, IX, and X; i.e., those patterns for which the vertex mixture representation is overparameterized). In cases in which the linear order constraint forms an inverted fork-like structure (e.g., $A \geq C$ and $B \geq C$ as in patterns II, III, IV, and V) the upward method should be used. Note that in line with the discussion of the vertex mixture method, no parameterization for the cross-over pattern VII can be construed given the heuristic sketched in this document. Additionally, we couldn't find a parameterization using the heuristic for pattern VIII.

Fork-Like Order Constraints

Consider a binary fork-like constraint with $A \geq B$ and $A \geq C$ with no constraint between B and C (similar to Figure 4, pattern X). For simplicity we call A head and B and C tails. For this case it seems that one needs to employ a variant of the downward method. Again, an important step is to identify the minima and maxima vertices (which differ from each other for more complex constraints as those discussed from now on). As with purely linear order constraints, a lower bound for the minimum value of the largest element is given by an equal distribution, $P(A) = P(B) = P(C) = 1/3$. However, the maximum amount of the smaller elements cannot easily be given as the equal distribution $P(A) = P(B) = P(C) = 1/3$ as well as $P(A) = P(B) = 1/2$ or $P(A) = P(C) = 1/2$ are possible values.

To build a model for a fork-like structure, the following steps need to be applied: (1) the head receives the minimum to fulfill the inequality; (2) distribute the remainder in equal parts according to the number of remaining tails; (3) for each element of step (2), one tail receives a freely estimated amount; (4) for the remainder from each step (3), recursively apply steps (1) to (3) but this time by only considering the order structure without the current and superordinate tails, reusing the parameters from superordinate step (3) for each element, or if only the head remains, assign the remainder to the head. For the case of the example above with three elements, this corresponds to the following

model (we omit the equations from now on for sake of brevity):

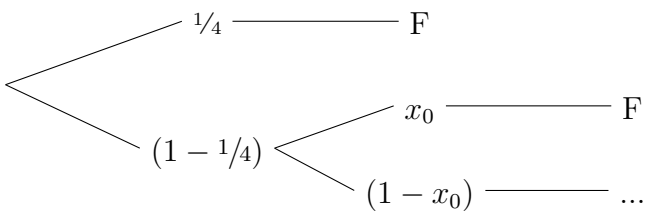


(5)

Again the question arises whether or not this parametrization imposes the order constraint and if all possible ordered sets of probabilities can be represented. As for the cases discussed before, the parametrization is equivalent to the one described in the appendix of the main document and hence both conditions are met. The order constraint is satisfied because the probability for each tail cannot exceed the probability of the head, given that step (1) is applied recursively and consequently each amount going into a tail also goes into the head. In other words, for each remainder from a step (2) the following step (1) controls that the inequality cannot be violated. Furthermore, it is easy to see that this method allows to represent the critical minima and maxima vertices. Furthermore, the proportion of the probability mass that is not bound by order constraint (i.e., by steps (1)) free parameters freely distribute the remainder which we believe allows to represent all possible sets of probabilities that follow the constraint. For example, in case there is no C (i.e., $x_2 = 0$), x_1 can completely control what proportion B can have with the extreme that in case of $x_1 = 1$,

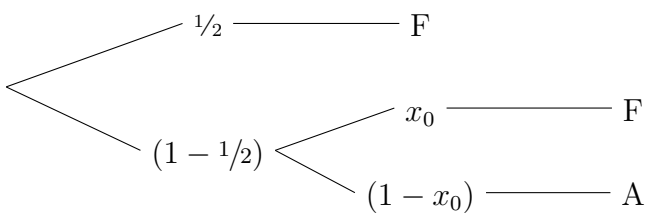
$P(A) = P(B) = 1/2$ (note that if $x_1 = x_2 = 1$, $P(A) = P(B) = P(C) = 1/3$).

Two further important notes apply to the method presented here for fork-like structures. First, this method can easily be used to model order constraints with more than two tails, however, in this case, the size of the model again grows rather rapidly. We implemented this method for a model with three tails in Appendix A (this constraint corresponds to pattern X). Second, this method can be extended for example in case we combine purely linear order constraints with the fork-like structure such that $F \geq A \geq B$ and $F \geq A \geq C$ (i.e., pattern VI). One way would be to preface the model (Tree 5) with the following structure using the initial downward method (this is necessary to allow $P(F) = 1$ and $P(A) = P(B) = P(C) = 0$):



(6)

Furthermore, in case of applying step (1) recursively (i.e., except for the first application) the two instances of A in Tree 5 would need to be replaced by a fork distributing the minimum amount for both F and A equally among F and A (i.e. in contrast to $1/2$ to A, $2/3 \times 1/2$ to each F and A). In addition, after considering F (Tree 6) and the two tails, in other words if only the heads remain (we consider both A and F to be head in this case), a simple order restriction enforcing $F \geq A$ (Tree 1) would need to replace the remaining two instances of A alone (the full model is given in Appendix B):



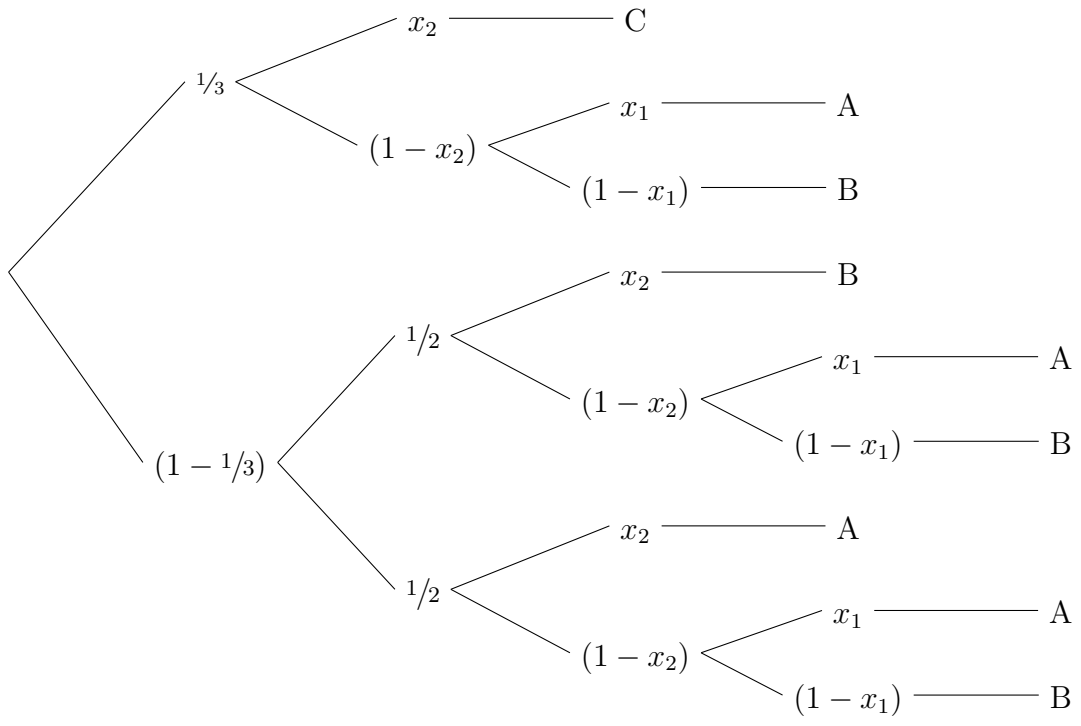
(7)

Note that again, this model not only satisfies the order restriction (i.e., A cannot be larger than F) it seems to allows for all possible sets of probabilities as exemplified by only considering F and A (i.e., $x_1 = x_2 = 0$ and consequently $P(B) = P(C) = 0$). As already said above, in the case of $x_0 = 1$, $P(F) = 1$ and $P(A) = 0$. Values of x_0 smaller than 1 allow for all possible combinations of F and A while satisfying the order constraint. For example, if $x_0 = 0$, $P(F) = P(A) = 1/2$; if $x_0 = 1/2$, $P(F) = 37/48$ and $P(A) = 11/48$.

Inverted Fork-Like Order Constraints

Consider a binary inverted fork-like (or joining) order constraint with $A \geq C$ and $B \geq C$ with no constraint between A and B (similar to Figure 4, pattern II). For simplicity we again call the upper elements A and B heads and the lower element C tail. To build such an order constraint using the heuristic discussed here we need to employ the upward method; in other words we need to consider the tail first. As before the relevant vertex is the maximum amount the tail (the smallest element) is allowed to obtain which corresponds to $P(A) = P(B) = P(C) = 1/3$. Furthermore, an important observation is again that to obey the order restriction, the elements A and B need to receive at least the same amount as C. However, as there is no ordering between A and B the steps simplify to the following: (1) take the maximum amount which can be applied to the smallest element and (2) assign a freely estimated amount of this proportion to the smallest element, (3) distribute the remainder from (1) equally among all superordinate elements and assign each element the same freely estimated amount of (2), (4) distribute the remainder of (3) freely among all superordinate elements. This simplification of the upward method leads to the following simplification of Tree 4 (i.e.,

the three element upward tree) in which Tree 3 is replaced by a simple binary branching:

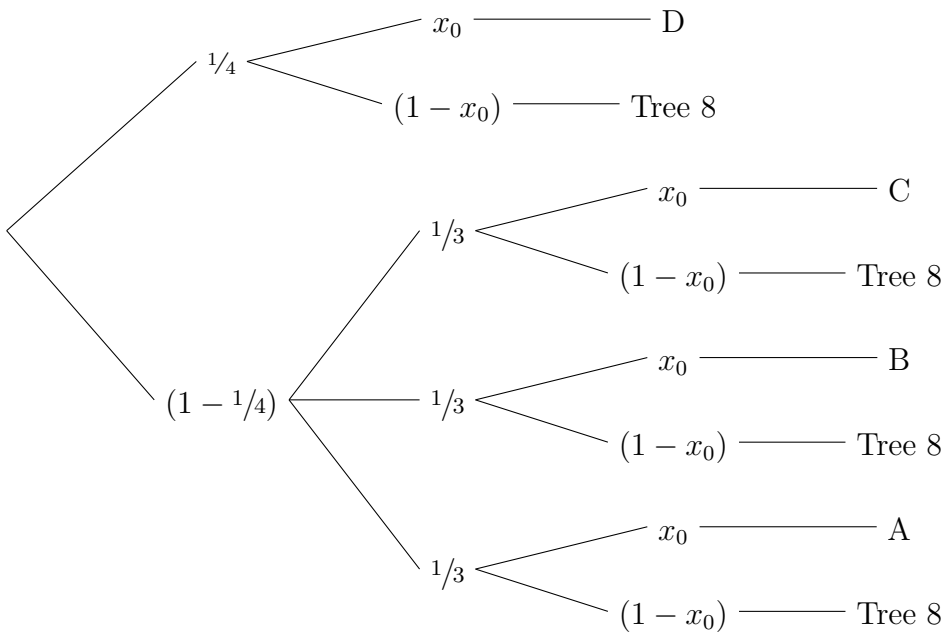


(8)

Similarly to the case of the purely linear order constraint this parameterization obeys the constraint as the smallest element cannot receive more than ⅓ and the amount alloted to the smallest element is also alloted to the superordinate elements. Furthermore, this parameterization is equivalent to the vertex parameterization discussed in the main document (Section "Extensions"). This method can be easily extended to inverted fork-like structures with more than two heads. In this case only the branching in step (3) and the constant in step (1) need to be adapted. We exemplified this for pattern II of Figure 4 in Appendix C.

The present approach can also be extended by adding further linear order constraints. For example, to parameterize pattern III of Figure 4, which adds a further element below C so that the final ordering is $A \geq C \geq D$ and $B \geq C \geq D$, one can simply start by applying the upward method to D and then paste Tree 8 to the remainders. Naturally we need to use the appropriate constant for C in this case and

observe again a dramatic growth in tree size:

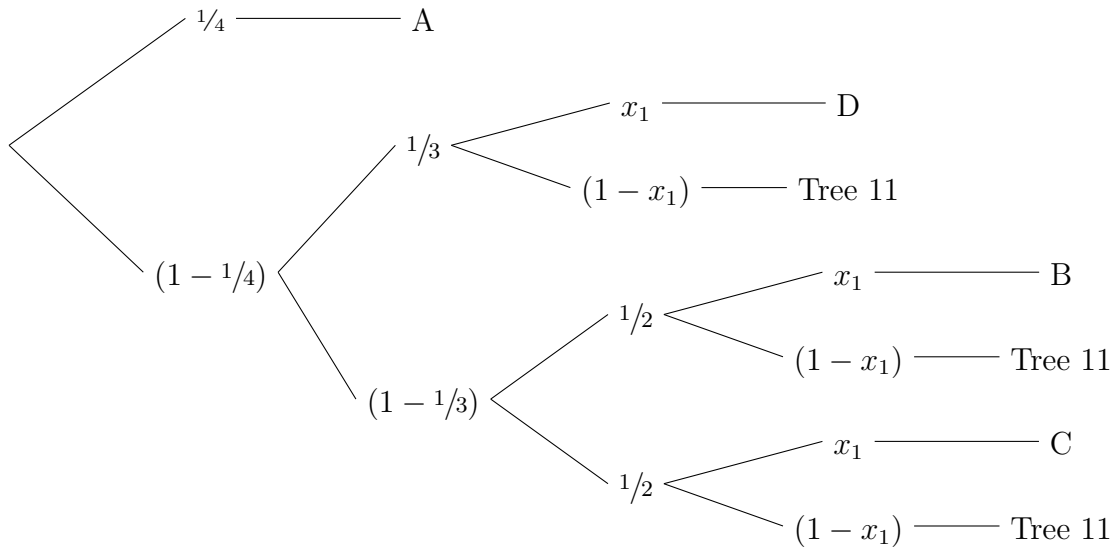


(9)

Combining Inverted and Normal Fork

As a final example, let us consider pattern V in which a fork is combined with an inverted fork for the following constrain, $A \geq B \geq D$ and $A \geq C \geq D$. Again it is important to consider the relevant vertices. At a minimum A needs to receive ¼ and the maximum amount for D is also ¼. For building this model we first need to apply the downward method on A and then, before implementing structures for B and C, apply the upward method to D. This approach leads to the following partial tree which

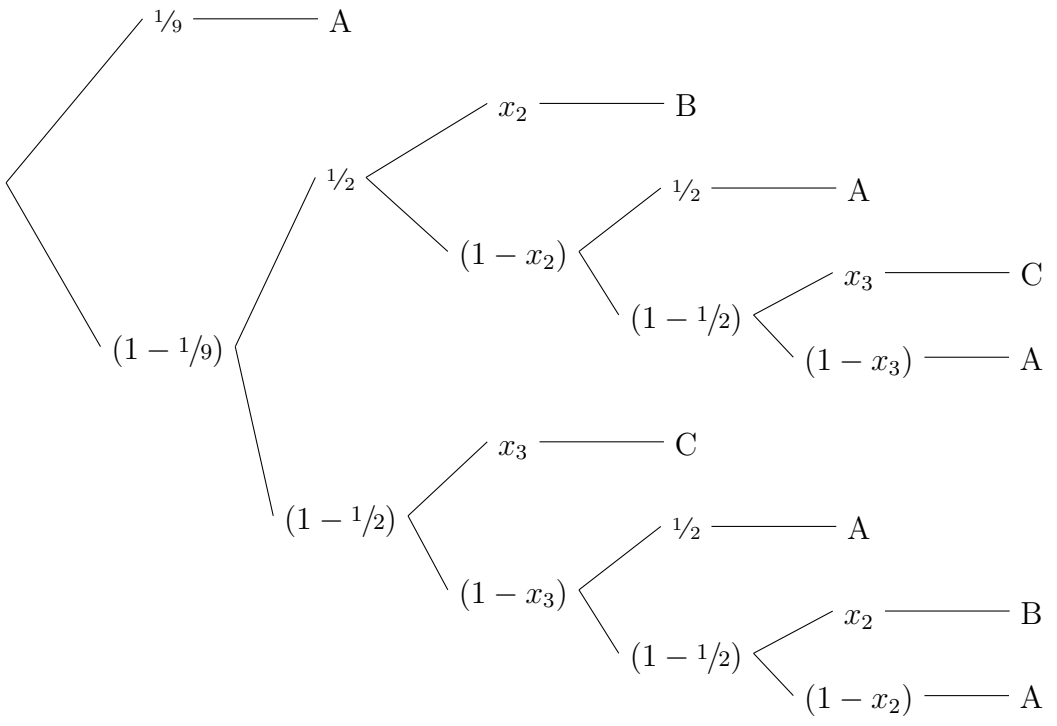
we will then combine with a subtree described in more details below:



(10)

This partial tree implements the two major constraints just discussed. A receives at least ¼ and D receives at maximum ¼. Before implementing the subtree which will be joined with the partial tree, we now need to consider some vertices again. When only considering the upper (inverted fork) part of the constraint, we see that now the minimum for A is ⅓ instead of ¼. To ensure this, A needs to receive, in addition to the already alloted ¼, $⅓ - ¼ = 1/12 = ¾ \times ⅑$ of the remainder. As only ¾ or $(1 - ¼)$ enter this part of the tree, the constant needs to be set to ⅑. With the exception of this new constant, the subtree is essentially identical to the regular tree for a fork-like

structure (Tree 5):



(11)

As before, we are confident that this model enforces the constraint and allows one to represent all possible sets of probabilities to be represented. To exemplify this consider the following cases representing some of the relevant vertices. If $x_1 = x_2 = x_3 = 1$, $P(A) = P(B) = P(C) = P(D) = 1/4$. If $x_1 = 0$ and $x_2 = x_3 = 1$, $P(A) = P(B) = P(C) = 1/3$ and $P(D) = 0$. If $x_1 = x_2 = 0$ and $x_3 = 1$, $P(A) = P(C) = 1/2$ and $P(B) = P(D) = 0$. Furthermore, if for example, $x_1 = 1/2$, $x_2 = 0$, and $x_3 = 1$, $P(A) = P(C) = 9/24$ and $P(B) = P(D) = 3/24$. This shows that relevant vertices and values in between can be successfully captured with this model.

Numerical Tests

In order to ensure the appropriateness of the parameterizations presented here and in Table 2 of the main document we performed some numerical test on them. First, we tested the parameterizations of the vertex method presented in Table 2. Those parameterizations surely constrain the predicted multinomial distributions to follow the

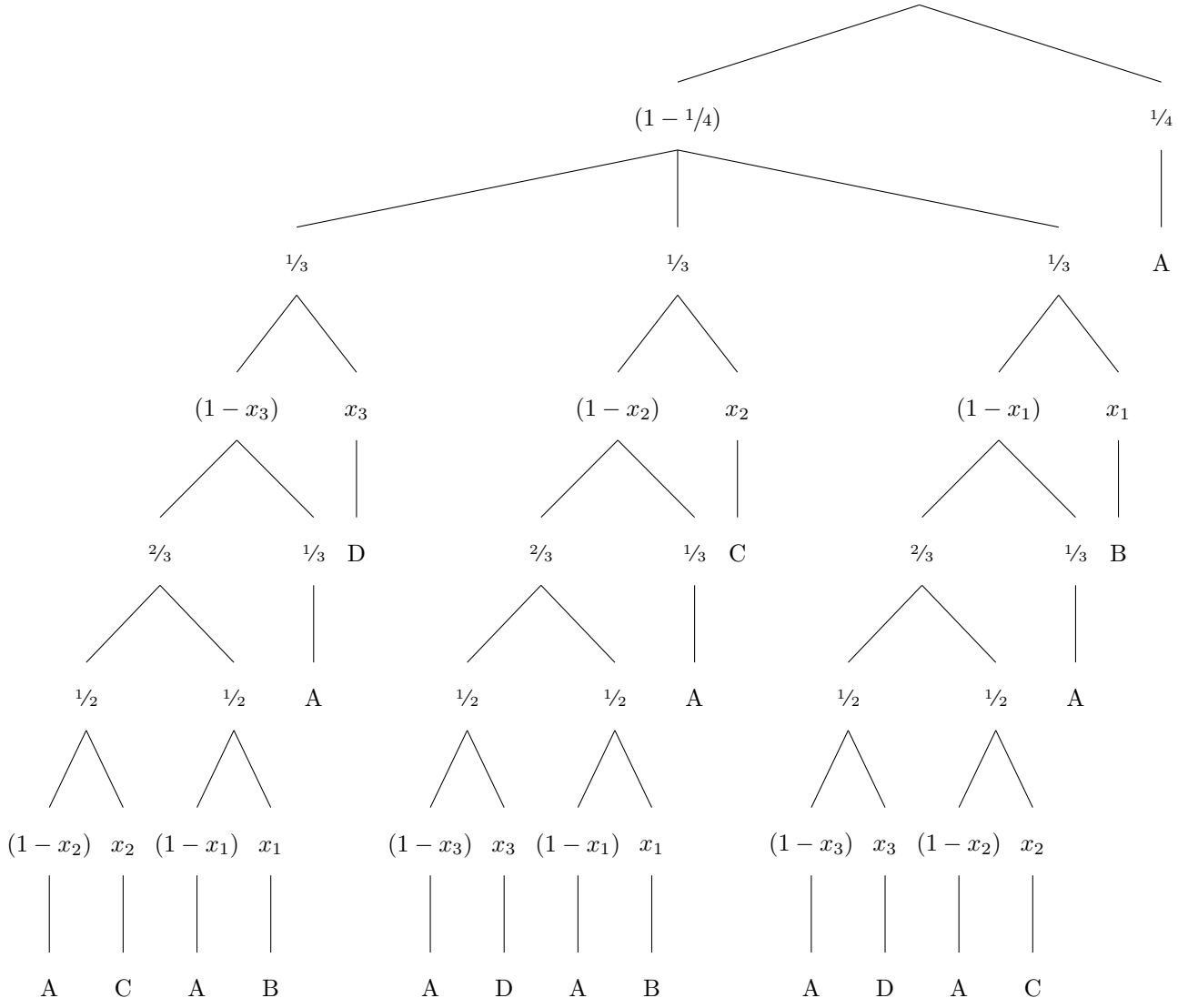
order constraint as they simply represent a mixture of the vertices. However, we did not prove for all parameterizations whether or not they allow all ordered set of probabilities to be represented (but see Appendix of main document). We fitted all possible ordered sets of probabilities for each of the parameterizations presented in Table 2 using a .005 grid (i.e., in ½ percentage steps). To minimize the impact of the numerical optimization routine used, we multiplied each set with 1000 before fitting the data (i.e., a set of $P(A)P(B) = P(C) = P(D) = .25$ was transformed into a value of 250 each). Furthermore, we performed 20 optimization runs for each set of probabilities with different random starting values. The results showed that the maximum deviation between observed and predicted values across all sets was .001 or $1/5000$ of the step size used. This shows that, at least for the chosen grid size of .005, the presented parameterization is able to represent all possible ordered sets of probabilities. The analysis scripts for those tests can be found in file `test_orderings.R`.

Next, we tested some of the parameterizations presented in this manuscript. Specifically we tested those parameterizations for which we did not show their equivalence to the parameterization presented in the main text, namely binary inverted fork (Tree 8), ternary inverted fork (Trees 9 and 8), combination of forks (Trees 10 and 11), and binary fork plus linear order above (Appendix B). For these parameterizations we not only tested all possible ordered sets of probabilities to show that the parameterizations are able to represent these, but also all sets of ordered probabilities that are not consistent with the order constraint to show that the parameterizations are not able to represent those. As these dramatically increased the sample size we reduced the grid (or step) size to .01. Again we multiplied all probabilities with 1000 before fitting them and performed 20 fitting runs with different random starting values for each set of probabilities. Fitting the possible sets of probabilities showed that the maximum deviation of observed and predicted values was .0006 or $1/16667$ of the step size, again showing that the parameterizations are able to represent all possible sets of probabilities. Fitting the sets of probabilities not consistent with the order constraint showed that the minimum summed absolute deviations of observed and predicted

responses (i.e., summed across all probabilities in one set) was 10 which corresponded to the step size or to the minimum deviation from the allowed set of ordered probabilities. This latter result showed that the parameterizations presented here enforce the order constraint. The analysis scripts for those tests can be found in file `test_orderings_supplemental.R`.

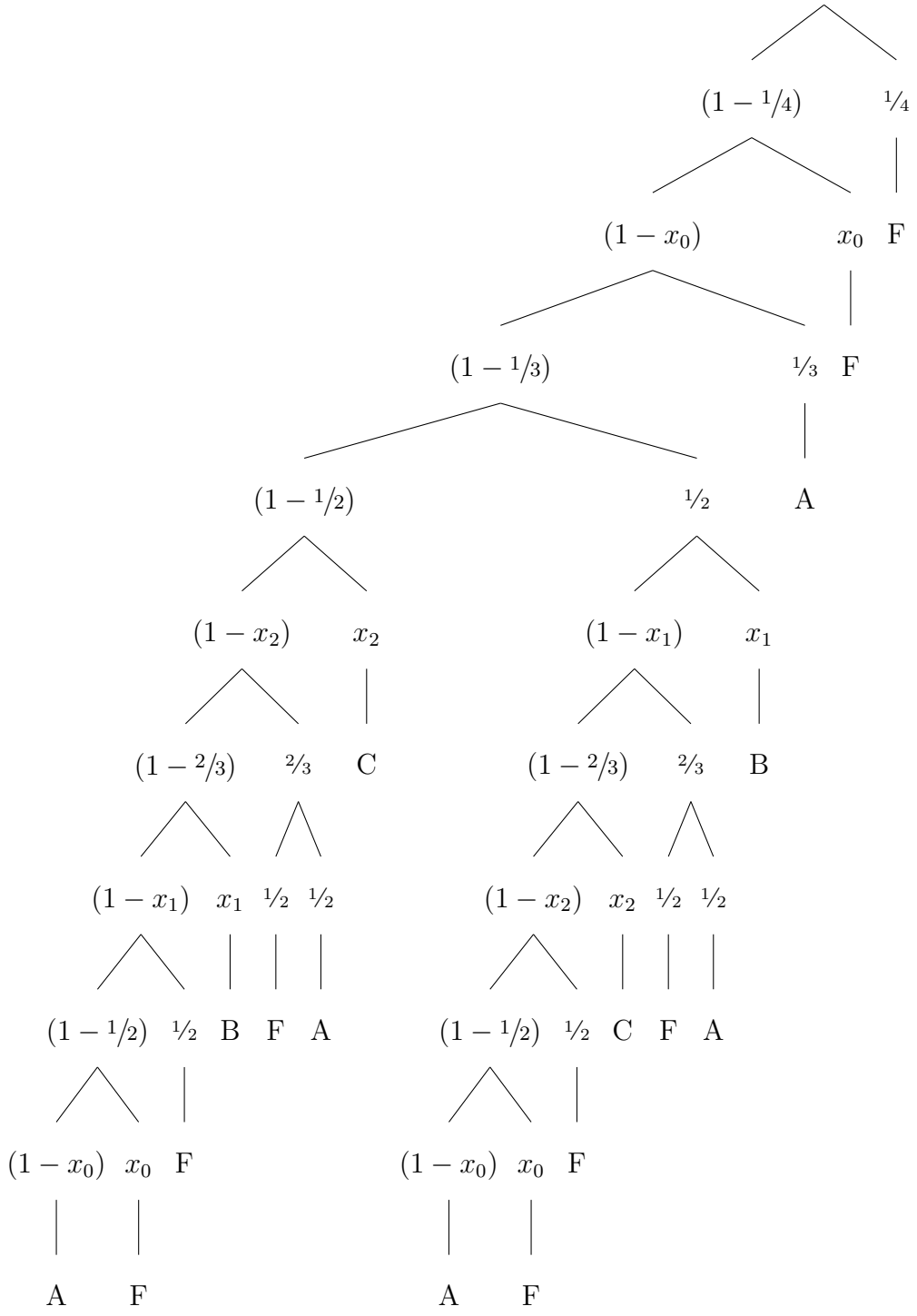
Appendix A

Model for ternary fork-like structure with $A \geq B$, $A \geq C$, $A \geq D$, and no further constraints (i.e., pattern X, Figure 4, main document).



Appendix B

Model for a combination of a purely linear order constraint with a fork-like order constraint: $F \geq A \geq B$ and $F \geq A \geq C$ (i.e., pattern VI, Figure 4, main document).



Appendix C

Model for ternary inverted fork-like structure with $A \geq D$, $B \geq D$, $C \geq D$, and no further constraints (i.e., pattern II, Figure 4, main document).