Combining System Dynamics and Decision Analysis for Rapid Strategy Selection

Nathaniel Osgood
Department of Computer Science, University of Saskatchewan[1]

Abstract

An important class of decision problems involve the selection of a policy in the presence of both uncertainty regarding future eventualities and a system exhibiting complex policy response. Within this paper, we examine the performance benefits of performing strategy selection for such problems using a hybrid modeling approach that combines decision analysis and system dynamics. Within this approach, a modeler uses a decision tree to encapsulate choices, uncertainties, and consequences (the last computed by a system dynamics model). While this hybrid technique offers many additional advantages in terms of expressiveness and the encouragement of systematic investigation of policy space, this paper focuses on the performance gains it provides. In particular, the use of decision trees to represent such decision problems permits the use of dynamic programming, which can dramatically decrease the costs of identifying a preferred policy beyond what is possible by evaluating possible policies in turn. The paper quantifies these performance advantages by means of recurrence relations for arbitrary trees, and derives inductively proven closed-form expressions of performance gain for complete trees. The results suggest that the hybrid method yields speedups exponential in the depth of the tree for both complete and randomly generated trees.

1 Introduction

Consider a situation characterized by unfolding uncertainty in which we wish to use a system dynamics model to make a set of policy decisions[2]. Under such circumstances, our decision-making at a certain point in time will frequently depend on the sequence of chance events that has occurred up to that point.

Employing the terminology of decision analysis, we use the term "decision rule" to refer to a function that specifies our sequence of policies given any possible history of events. In many problems, we seek to identify the best decision rule, recognizing that in most cases the best decision rule will need to be *adaptive* – that is, will vary in its choice of a decision based on the event history experienced until this point.

The purpose of this paper is to communicate the computational advantages of a novel hybrid technique for identifying the optimal decision rules in such situations. To do this, we contrast the computational costs of two possible techniques for identifying an optimal decision rule. The first technique for identifying a preferred decision rule requires evaluating each decision rule in turn within the system dynamics model and selecting the

[1] Address correspondence and reprint requests to Nathaniel Osgood, Department of Computer Science, 280.6 Thorvaldson Building, 110 Science Place, University of Saskatchewan, Saskatoon, SK S7N 5C9 Canada. Email: osgood@cs.usask.ca

[2] We make no assumptions here as to the etiology of this uncertainty. Some of it may lie with respect to occurrence of external events, while in other cases it may reflect degree of confidence concerning internal factors or the effectiveness of our decisions.

rule that scores the best. The alternative approach (described comprehensively in a forthcoming paper [Osgood & Kaufman, 2005]) involves constructing a decision tree to represent the decisions and uncertainties and performing a dynamic-programming based backwards induction algorithm on that decision tree in order to arrive at a preferred decision rule. This new[3] technique is general, scalable and can reduce the cost of strategy identification by a factor exponential in the depth of the tree.

The advantages of this new technique will be particularly pronounced for complex decision problems involving many levels of decisions and events, heavyweight system dynamics models and cases in which the model includes endogenous uncertainty. An important priority for realizing the full advantages of the technique is the provision of software support for the hybrid models.

The next section introduces basic terminology and a simple motivating example to which we will refer throughout the paper. Section 2.6 discusses the traditional technique for identifying a preferred decision rule and illustrates how this technique would be applied to the example. Section 4 briefly characterizes the hybrid technique for decision rule selection and illustrates its application on the example. Section 5 provides recursive expressions for the computational costs of both techniques and demonstrates how the costs scale empirically with full and randomly generated decision trees. Section 0 summarizes paper results and discusses the prospects for realistic application of the hybrid algorithm for strategy identification.

2 A Simple Example

2.1 The Problem

This section introduces a very simple system dynamics policy choice problem in order to serve as a concrete point of reference throughout the paper. In this stylized example, we wish to design a capacity expansion plan for our electrical infrastructure over the timeframe from 0 to t_E. Because the electrical demand anticipated will vary widely depending on the level of economic growth experienced over that same timeframe, we wish to consider adaptive plans that offer flexibility to react to changing economic conditions. Suppose further that we have at hand a descriptively complex system dynamics model capable of simulating the dynamic economic and environmental consequences of an electrical infrastructure build out sequence in the context of a specified pattern of ongoing economic growth (and thus demand) as well as a single-attributed preference function[4] that allows us to express our level of preference regarding the consequences of different scenarios. This preference function might, for example,

[3] While simple, single-decision decision trees have previously been employed to guide exploration of system dynamics model, the author is not aware of previous explicit use of multiple-decision decision trees together with system dynamics. There are many cases in which exploration of scenarios or decision rules using a system dynamics model have followed the structure of an implicit decision tree, and there are likely many cases in which a very simple, single-level form of backwards induction was implicitly used to derive an optimal decision in the presence of uncertainty (for example, determining the preferred choice under different possible eventualities). The author is not aware of any cases in which this process has been formalized and extended to multi-level decision trees.

[4] For scoring each consequence we specify the selection of a preference function rather than some more general metric in order to allow for proper handling of uncertainty (risk) when evaluating decision rules.

consist of some discounted utility measure based on the price of electricity, and level of electrical brownouts or blackouts experienced, etc.
It should be emphasized that this example is as simple as possible for the sake of clarity in order to provide a concrete sense of some of the principles and tradeoffs. The example does not demonstrate important aspects that extend from the full generality of the framework.[5]

2.2 Modeling Approximations

Suppose for simplicity that we wish to consider only two possible stages of capacity expansion and divide our planning horizon of [0, t_E) into two smaller timeframes: [0 to t_M) and [t_M, t_E). Suppose further that we choose to dichotomize the level of economic growth for each of these time horizons into one of two cases (high economic growth vs. low economic growth), and that we specify the subjective likelihood that high economic growth will occur within each of these timeframes with probability p_E and p_L. For expository simplicity, we assume that the probability of high economic growth in both periods is independent of the event and decision sequence up until that point. Suppose that we wish to consider only plans involving one of two possible decisions for each of the time windows of interest: The choice of either expanding or not expanding our generation capacity (where these expansions or non-expansions are assumed to begin at either 0 or t_M).

2.3 The Corresponding Decision Tree

Figure 1 depicts a decision tree laying out the set of uncertainties, decisions and consequences associated with this example.

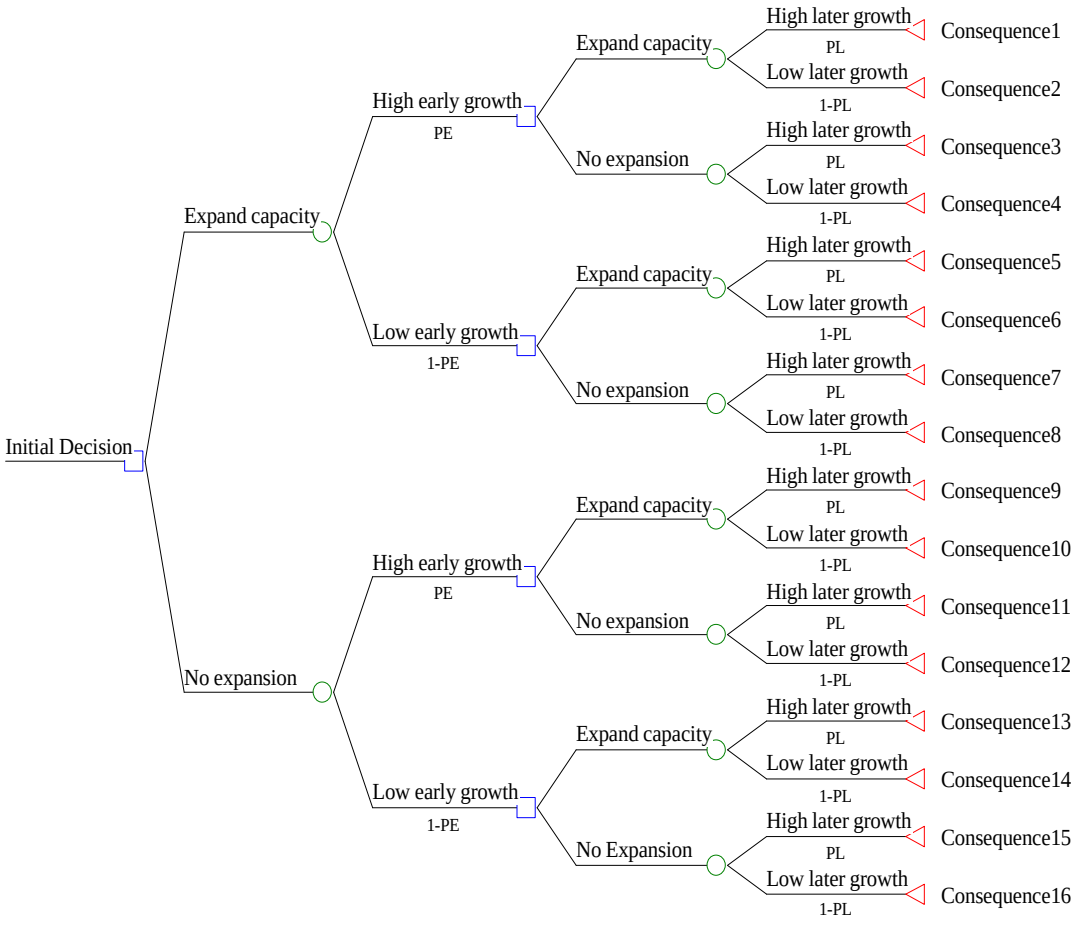


Figure 1: Decision Tree for the Simple Capacity Decision Example

The decision tree is oriented so that time flows from left to right. The tree includes square *decision* nodes that represent choices to be made at a particular point in time (e.g. choice of add vs. do not add capacity), circular *event* nodes representing uncertainties over time (e.g. occurrence of high vs. low economic growth), and triangular terminal nodes. Each terminal node is associated with the *consequences* of a particular *scenario* – the unique sequence of specific events and particular decisions that lie on the path from the root node to that terminal node.

2.4 Decision Rules

Recall that a decision rule describes the decisions that will be made in response to any sequence of events. Table 1 lists the set of possible decision rules for the problem at hand. Because the decision at time 0 is not preceded by any events, it is not conditional on any event occurrence. By contrast, the decision at time t_M may be conditional on the growth of the economy for the time period $[0, t_M)$. It bears noting that there are quite a number of decision rules despite the very simple structure the decision tree (just two decisions and two events). As we will see, the number of decision rules will in general rise rapidly with the depth of the tree.

Table 1: Enumeration of possible decision rules for the example shown in Figure 1. Each row corresponds to a different decision rule.

Rule Id	Decision rule (expressed informally)		Terminal Nodes
	Decision at 0	Decision at t_M	
1	Do not expand	Do not expand	11,12,15,16
2	Do not expand	If e=Low, expand else do not expand	11,12,13,14
3	Do not expand	If e=High, expand else do not expand	9,10,15,16
4	Do not expand	Expand	9,10,13,14
5	Expand	Do not expand	3,4,7,8
6	Expand	If e=Low, expand else do not expand	3,4,5,6
7	Expand	If e=High, expand else do not expand	1,2,7,8
8	Expand	Expand	1,2,5,6

For this example, we must choose between 8 decision rules, reflecting the fact that each decision rule must choose at two points in time between two possible choices, and the fact that the second decision can be made in response to two possible eventualities (low economic growth vs. high economic growth for [0, t_M)).

2.5 Relationship between Decision Rules and Terminal Nodes

Understanding the relationship between decision rules and terminal nodes is central to understanding the tradeoffs and analysis presented in this paper. We therefore pause to make two remarks on this relationship.

In all but the most degenerate trees, a particular decision rule will be associated with many terminal nodes. (See Table 1) Each of these terminal nodes represents a scenario – a particular sequence of events and decisions – that could occur if the decision rule were in place. The path to this terminal node captures both the sequence of events and the decision rule's particular responses to those events that define the scenario. For example, under decision rule 3, different patterns of events could lead to scenarios associated with any of the terminal nodes labeled 9, 10, 15, and 16. This is depicted graphically in Figure 2. The fact that there are four such terminal nodes reflects the fact that in this simple example there are 2 possible events, each with 2 possible outcomes. To understand the effectiveness of a decision rule, it is necessary to calculate the consequences of the various scenarios with which it can be associated. Conversely, once equipped with a preference function for judging consequences we can use the preference function to express our preference for decision rules as well. We do this by considering the set of possible scenarios with which each decision rule is associated and the likelihood with which each of those scenarios can occur under that decision rule.

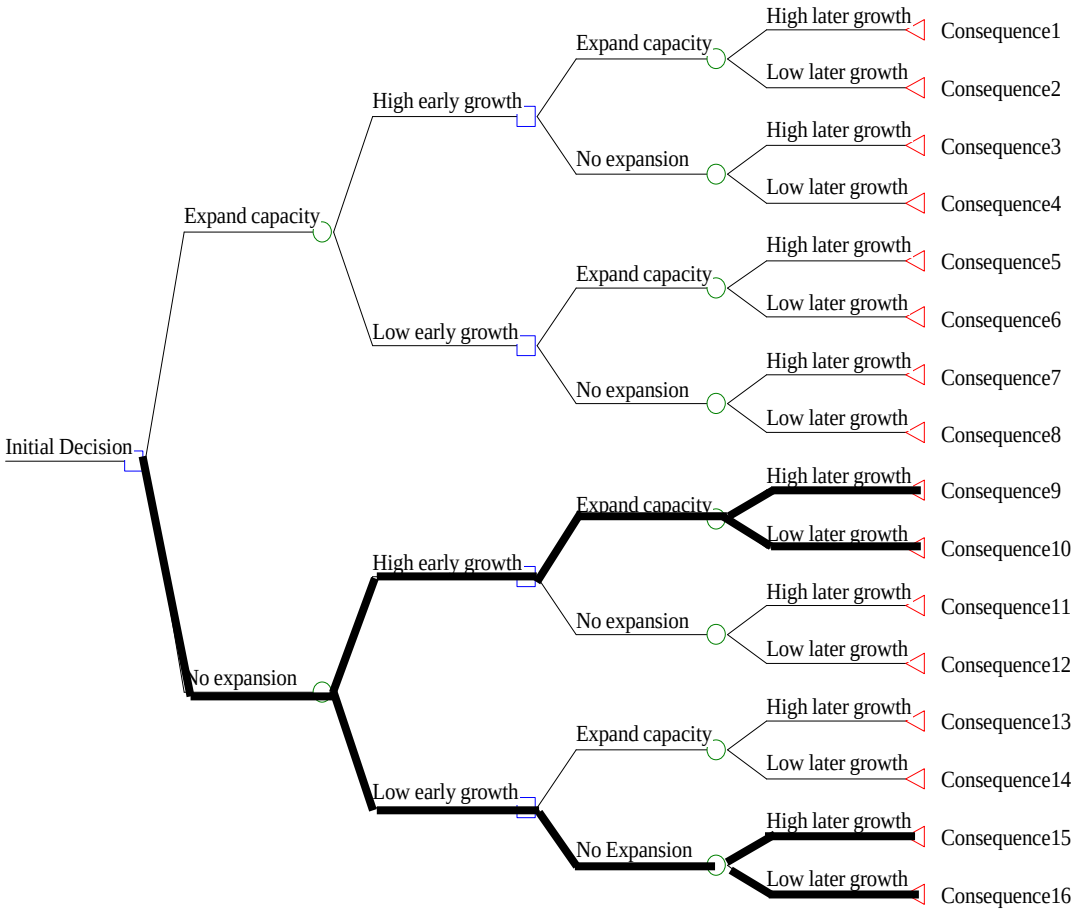


Figure 2: Representation of a specific decision rule in the decision tree. Depending on which event sequence takes place (which emboldened path is taken), this decision rule could lead to any of the scenarios represented by terminal nodes 9, 10, 15 and 16.

We noted above that each decision rule will typically be associated with several terminal nodes. Conversely, a particular terminal node *t* within a decision tree will in general be associated with a set of decision rules. The decision rules in that set happen to be identical in their response to the particular sequence of events represented on the path to terminal node *t*, even though they may differ dramatically in terms of the decisions they would make for other sequences of events. Evaluating the scenario associated with the terminal node tells us something about the behavior of all of the decision rules. For example, terminal node 15 in Figure 1 could result from either decision rule 3 or decision rule 1. This reflects the fact that both decision rule 3 and decision rule 1 both forego any expansion in response to a scenario in which low early growth is followed by high later growth.

2.6 Identifying a Preferred Decision Rule

Consider now the problem of identifying the best decision rule given our problem structure. Using the terminology introduced above, we seek to find an optimal decision

rule for expanding our electricity generation capacity over the timeframe from 0 to t_E in the light of uncertainty regarding economic growth from time 0 to time t_E.
The next two sections discuss two general approaches for this type of planning problem. The first of these approaches involves enumeration of the decision rules and evaluation of each decision rule in turn; the other involves identifying the optimal decision rule through roll-back on a decision tree. For each case, we will discuss the application of the general technique to the specific example shown in Figure 1.

3 Identifying Preferred Decision Rule Endogenously

The first approach for identifying an optimal decision rule involves iterating through each of the possible decision rules in turn. Each such rule is evaluated by executing the system dynamics model on each of the possible scenarios that can occur[6]. The decision rule is then scored using the weighted average of the preference function applied to each of these event-history-specific scenarios where the weighting reflects the likelihood of occurrence of that scenario. Following Raiffa, we term this procedure a *normal form analysis*.
Consider applying this technique to the example whose decision tree is shown in Figure 1. We begin by considering the set of decision rules that must be compared, as shown in Table 1. For each possible decision rule, we encode the decision rule as one or more formulas in the system dynamics modeling package. These formulas determine which decisions to make at each point in time, possibly taking into account the event and decision history experienced to this point. The decisions and event structure of this example is simple enough that the rules shown in Table 1 could likely be implemented in just one or two formulas in the system dynamics model.
For example, consider the evaluation of rule 3 in Table 1, which foregoes a capacity expansion at time 0 and then performs one starting at time t_M if and only if the economy experienced high growth from [0, t_M). Recall that for expository simplicity we are assuming that the occurrence of economic growth is exogenous to the model, being specified for example by a time series or as a parameter specifying growth rate. A capacity decision in the system dynamics model might be represented as a formula governing the flow into an appropriate stock (e.g. "generation capacity under construction"). The formula governing this flow could be designed so that for a window of time starting at t_M a flow would be generated into the stock if the parameter specifying the rate of economic growth during [0, t_M) indicated a high level of growth.
In order to evaluate a decision rule, it is necessary to run the simulation on each possible scenario (possible sequence of events and decisions)[7] that can occur under that decision

[6] In a deterministic model, each of these scenarios would typically be executed in a distinct system dynamics run. For a stochastic model, several scenarios may be executed together in a single Monte Carlo run.
[7] If event occurrence were endogenous (as might be the case if events occurrence were dictated by a stochastic process and by endogenous dynamics) then multiple sequences of events might be examined within a single Monte-Carlo run. For example, if electrical generating capacity itself were to probabilistically influence the economic growth experienced, we would not seek to exogenously impose an assumed economic growth rate, but allow it to be generated by the system dynamics model itself. This issue is discussed in much greater depth in a companion paper [Osgood & Kaufman, 2004]. While both approaches examined in this paper scale gracefully to handle endogenous events, for simplicity we confine

rule. Each simulation run computes the consequences of a particular scenario. For each such run we record the likelihood of scenario occurrence and the result of applying the preference function to the scenario consequences (typically performed endogenously to the system dynamics model).

For the example, once we have specified the formulas encoding decision rule 3 in the system dynamics model, we would have to run the model separately for each possible scenario that can result during the application of this decision rule. These scenarios (associated with terminal nodes 9, 10, 15, 16 in Figure 1) correspond to different possible combinations of events – i.e. combinations of high and low early and late economic growth (LL, LH, HL, HH). After running the model on each of these four scenarios in turn, we record the preference-adjusted outcome (as specified by the appropriate variable in the system dynamics model) and the probability of the occurrence of each ($(1-p_E)(1-p_L)$, $(1-p_E)p_L$, $p_E(1-p_L)$, $p_E p_E$).

When the consequences for all possible scenarios are explored for a given decision rule (e.g. 9, 10, 15 and 16 for rule 3), we can complete the evaluation of the decision node. We do this by determining the expected value of the set of scores that resulted from applying the preference function to each of the terminal nodes executed. For decision rule 3, for example, the decision rule's score would be computed by the formula:

$$p_E p_L c_9 + p_E (1-p_L) c_{10} + (1-p_E) p_L c_{15} + (1-p_E)(1-p_L) c_{16}.$$

Where we use the notation c_i to indicate the result of applying the preference function to the consequence for scenario (terminal node) i.

By performing this procedure for each possible decision rule in turn, we can arrive at a tableaux of the expected preferences for all decision rules. We then select as optimal the decision rule with the highest expected preference.

Given that each evaluation of the system dynamics model will in general be expensive (particularly if stochastic effects require running a large set of particular realizations), one statistic of considerable interest is the total number of particular terminal nodes (specific scenarios, each corresponding to a unique event and decision history) that require evaluation in order to identify the optimal strategy. As suggested by Table 1, for the particular example at hand we need to evaluate 4 terminal nodes in the tree in order to evaluate each decision rule. Given that we evaluate each decision rule in turn, the total number of particular terminal node executions required in order to identify the best possible decision rule is thus 8*4=32.

4 Identifying an Preferred Decision Rule Using a Decision Tree

The section above described a procedure for evaluating decision rules endogenously within the system dynamics package. In this section, we introduce an alternative hybrid approach that supplements the system dynamics model with a decision tree model specific to the problem. This method relies on the system dynamics model for simulation of each scenario (terminal node) but eliminates the need to enumerate all possible decision rules, instead using dynamic programming (via backwards induction) to rapidly identify a preferred decision rule from the set of preference-adjusted scenario outcomes.

---

our analysis here to the case where we must exogenously specify event occurrence, with each event history requiring a single execution of the system dynamics model.

For a case where all events are exogenous[8], the hybrid technique begins with the construction of an explicit representation of the decision tree corresponding to the situation of interest (for the example, see Figure 1). This initial tree should include a specification of the branch probabilities but should for the time being leave unspecified the consequences associated with each terminal node.

Following the construction of the tree, the modeler proceeds with a system dynamics evaluation of the terminal nodes of the decision tree, each of which represents a particular scenario of interest characterized by a unique history of chance events and decisions. For each such terminal node, the system dynamics model is parameterized to represent the scenario (decision and event history) of interest and the model is run. During or (less frequently) following the run, a preference function is applied to the model-computed consequences – typically trajectories of particular variables in the system dynamics models. We then associate the (scalar) value of the preference function with the terminal node in the decision tree.

For the example we have been using, consider executing terminal node 1 in Figure 1. For this terminal node, the system dynamics model would first be parameterized to represent a scenario involving an initial expansion of generation capacity, followed by a high level of growth over the interval $[0,t_M)$, followed by a further expansion of production capacity starting at time t_M and finally culminating in another period of high economic growth from $[t_M, t_E)$. Following parameterization, the system dynamics model would be run to produce the consequences of interest. Because the system dynamics model in this case includes its own internal preference function, the model would itself apply this function to the results of this evaluation and report the result as part of its final output. The final output value for this variable would be set as the consequence of terminal node 1 in the decision tree model, for use during the next (rollback) stage.

Two points about this process bear emphasizing.

- The results of executing a given terminal node is wholly determined by the model itself and the specific history of events and decisions that play out during its execution[9]. The results of executing a given terminal node are *not* determined by any other feature of the broader decision rule with which this terminal node may be associated. As noted in Section 2.5, a particular terminal node will in general be associated with (can result from) many different decision rules. Although they may differ entirely in their treatment of other sequences of events, the decision rules will share identical responses to the particular sequence of events that gives rise to the terminal node. As a result, the pattern of causation leading up to the

[8] This section discusses the application of this approach to the simpler case in which no endogenous uncertainty is present. The handling of endogenous uncertainty requires a more general approach in which the tree is constructed dynamically as Monte Carlo ensembles are run on for meta-scenarios consisting of successively different combinations of possible decisions and any exogenous uncertainties. Contiguous events are aggregated and approximated so as to allow for meaningful decision rule induction. Within this approach, each meta-scenario output consists of a risk profile of preference-adjusted outcomes, from which (because of the properties of preference functions) we extract the mean value for rollback. A companion paper provides a comprehensive introduction to this combination of system dynamics ands decision analysis [Osgood & Kaufman 2005] for the more general case where endogenous uncertainty is present and discusses many other advantages offered by this technique.

[9] For a model including endogenous stochastics, any random number generator state would also have to be included in this list.

terminal node will be identical for these different decision rules and the system dynamics state generated by and output generated from executions of these different decision rules on this *particular event history* will be identical.

- The system dynamics model evaluations here are being performed once for each possible *scenario* (in this case, the terminal nodes labeled from 1 to 16) and not for each possible *decision rule*. As we will see, in most decision trees there are far more possible decision rules than there are scenarios.

After having determined the preference-adjusted consequences associated with each terminal node (scenario), the modeler can use the dynamic programming algorithm of *backwards induction* ("rollback") to identify the preferred decision rule. This process operates recursively on the decision tree by determining the preferred decision for each decision node and requires no further system dynamics runs. To help clarify its operation and running time, we describe the backwards induction algorithm "BI(·)" below in procedural pseudocode.

- **Terminal Nodes.** The rollback value of a terminal node is simply the value of the preference function applied to the (time-indexed) system dynamics output for that node. As noted above, for simplicity we will typically compute the preference function for the output directly in the system dynamics model.

```
BI(t) :
        return f(SDOutput(t))
```

- **Event Nodes.** For event (chance) nodes, the backwards induction algorithm associates the value of the event node with the expected value of the results of applying the backwards algorithm to its children.

BI(e) :

return $\sum_{c \in child(e)} p(c)BI(c)$

- **Decision Nodes.** For decision nodes, the backwards induction algorithm determines the optimal choice for the node and sets the value of the node to be the value of that choice. The optimal choice for the node is the choice whose expected preference value is greatest.

```
BI(d) :
        vMax = -1
        cMax = nil
        for c  in Child(d)
                v = BI(c)
                if v > vMax
                        vMax = v
                        cMax = c

        OptimalDecision(d) = cMax
        Return vMax
```

In contrast to the normal form analysis presented in the previous section, this technique requires one run of the system dynamics model for every scenario (terminal node of the decision tree); the particular example at hand requires 16 execution of the system

dynamics model (one for each of the terminal nodes labeled 1-16) rather than the 32 required for the traditional algorithm. We note that this cost is far below that required by the normal form analysis. As we will see in the next section, this is typical of evaluation of decision rules on non-trivial trees.

5 Performance Comparison

This section examines the performance burden associated with each of the two techniques for identifying a preferred decision rule.

Before starting the analysis we note again our assumption that because executing the system dynamics model is expected to be relatively computationally expensive compared to construction of or simple arithmetic operations on decision trees. The latter operations are linear in the count of nodes in the tree, likely with a very small coefficient. As a result, the performance cost for each technique is measured in terms of the number system dynamics model runs required.

We will characterize below these performance costs in recursive fashion. We also provide closed form solutions for these performance costs for the special case of complete trees of alternating decision and event nodes and which have uniform branching c for event nodes and *d* for decision nodes. The correctness of these formulas is established via an inductive proof included within this paper's appendix.

5.1 Performance when Identifying Preferred Decision Rule Endogenously

The first form of analysis we will examine is the normal-form analysis discussed in Section 3. Recall that when we rely upon the system dynamics model to endogenously represent decision rules we enumerate the different decision rules, evaluating each in turn. In order to evaluate each rule, the rule must be encoded using one or more formulas in the systems dynamics model. In order to evaluate a decision rule, we must execute the system dynamics model to compute the consequences resulting from adhering to this decision rule. As we saw in Table 1 for the example, this evaluation process typically involves several particular runs – each of which characterizes the performance of that decision rule under different potential event histories.[10] Because the simple structure of the example, each decision rule is uniformly associated with 4 possible scenarios – one for each of the possible combinations of economic growth in the early and late periods.

As a first step towards quantifying the performance of this approach, we first compute the count of decision rules that must be encoded in this fashion. We will then consider how many particular system dynamics runs are required for evaluating the set of all decision rules for an arbitrary decision tree.

[10] Note that for a model with endogenous stochastics, a particular run of the system dynamics model would typically conceptually correspond to the evaluation of all terminal nodes at serve as leaves for some subtree rooted within the decision tree.

5.1.1 Decision Rule Count

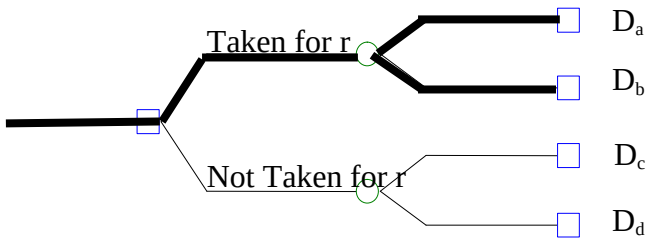


Figure 3: Decision nodes requiring specification within a rule; because the root decision has chosen the upper choice, decisions D_c and D_d do not require specification.

An important insight for quantifying the count of decision rules associated with a decision tree is the realization that while decision rules in general specify particular decisions for decision nodes, they need not specify a specific decision for *every* decision node in a decision tree. Instead, decision rules must specify decisions for every *logically consistent* set of decision nodes in a decision tree.

To understand this distinction and the notion of "consistency" to which it appeals, consider the decision tree depicted in Figure 3 in light of a specific decision rule *r* in which the decision associated with the top branch of the root (decision) node has been specified. A decision rule such as *r* must specify particular decisions to obtain for the root decision node as well as descendant decision nodes D_a and D_b. By contrast, decision rule *r* need not specify a decision for either D_c or D_d because the set of decisions that have been decided upon in *r* (in particular the decision associated with the root node) will make it impossible to reach a situation where these D_c or D_d are required. We refer to this phenomenon informally as "decision node occlusion": Selection of a particular choice (child) for a decision node *d* obviates the need for a decision node to specify decisions within subtrees rooted at other children of *d*. This observation will have substantial implications in limiting the count of decision rules that are possible for a particular decision tree.

A general formula for the number of decision rules can easily be expressed recursively. For a root node *n*, we can express a formula s(*n*) for the number of distinct decision rules associated with the tree rooted at *n*. We define *s(n)* recursively based on the type of node:

- **Event node *e*:** For a subtree rooted at an event node, it is not possible for a decision maker to determine which event will occur. As a result, a given decision rule will have to *independently* specify whatever decisions require specification *for each possible outcome* of this chance event. For the example shown in Figure 1, a decision rule for a tree rooted at a chance node associated with early growth level of the economy would have to specify distinct sets of decisions for the case where the economy experiences high early economic growth and the case where the economy experiences low early economic growth. Because the decision rule can independently specify any decisions required for each event outcome, the set of possible decision rules required under node *e* is the set of possible *combinations* (cartesian product) of the decision rules possible for the subtrees

associated with each possible outcome (child) of the event. The size of that set is just the *product* of the number of possible decision rules required for each child of the event node, i.e.

$$s(e) = \prod_{c \in child(e)} s(c)$$

- **Decision node *d***: For the case of a *decision* node, a specific decision rule need only specify exactly one possible outcome of this decision – but will also have to specify particular decisions required by any decisions beneath this branch (but not under any other branch). The set of decision rules that are possible for the subtree rooted at *d* is simply the union of the set of decision rules that are possible for the subtrees rooted at each particular child of *d*, with one twist: Each decision rule from child *c* is extended with an extra element that specifies that the parent (decision) node has made choice c. The count of particular decision rules associated with this node is thus the same as the cardinality of the union of the set of decision rules associated with the children of the decision node. i.e. $s(d) = \sum_{c \in child(d)} s(c)$.
- **Terminal node *t*:** There is only one (empty) decision rule possible for this node: Thus, *s(t)*=1.

The values of these recurrences will depend on the structure of the particular decision tre on which they are invoked.

Consider a complete tree of alternating layers of decisions and event nodes, where the decision nodes have *d* children and the event (choice) nodes have *c* children. Based on these recurrences, we can arrive at a closed-form expression for the count of strategies at height h of the tree (where the terminal nodes are considered to be at height 0). Slightly abusing the notation introduced above, we can express the number of decision rules required in the tree as a function of the height in this complete tree:

$$s(h) = d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{c-1} \right)}$$

This paper's appendix includes a straightforward inductive proof for the correctness of this formula.

5.1.2 Count of Runs Required to Evaluate a Decision Tree

The previous section provided a recursive characterization the number of distinct decision rules possible for a decision tree of arbitrary structure. For a normal form analysis, we must iterate through the set of decision rules, evaluating each in turn. Each terminal node (scenario) execution required by this process must execute within the system dynamics model and is expected to impose a heavy performance cost. This section builds on the last section by providing a recursive formula for the count of particular scenario executions (runs) *r(n)* required for a decision tree rooted at node *n*. The formula for *r(n)* is based upon the count of decision rules *s(n)* that are required in that tree.

Recall from Section 2.5 that in general, each terminal node can be reached by a common sequence of events under many distinct decision rules. Under normal form analysis a

given terminal node may therefore require execution many times, as part of the evaluation of many different decision rules. For example, in Table 1, each possible terminal node (e.g. 1) must be executed to evaluate each of two distinct decision rules. Not surprisingly, this holds more generally*: The number of times a given terminal node must be evaluated is equal to the number of decision rules that include it*. Thus, for the example under consideration, each terminal node is evaluated twice by the procedure for normal form analysis.

An important insight for quantifying the count of decision rules that could result in evaluation of a given terminal node T is to consider the general characteristics of those decision rules in light of T's location in the decision tree. In particular, consider the *decision nodes* located on the path from the root of the decision tree to T. Because the path terminates in a terminal node (T), we know that each such decision node *d* on the path must have some child on the path to T. This child represents a particular specific choice for decision *d*, and any decision rule that can lead to T must include that choice.

As a result of this reasoning, given a terminal node T we will in general know many components of the decision rules that can lead to the scenario represented by T. These decision rules are not fully fixed – they may vary with respect to the values they associate with other decision nodes which are not on this path but which are consistent with it (See Section 5.1.1). These other decision nodes may vary across all of their logically consistent values.

Casting this insight into a quantitative form, we can construct a recursive expression for the count $r(n)$ of terminal nodes (scenarios) requiring execution for a given tree rooted at n:

- **Terminal node.** For a trivial subtree consisting just of a terminal node, we have just a single run to execute: $r(t)=1$.
- **Event node.** As we saw in Section 5.1.1, for an event node a given decision rule must specify values for decisions associated with all different possible chance eventualities associated with the event. There is a combinatorial effect here in terms of the number of runs required: If a given branch *c* of the event node itself requires *m* distinct runs to evaluate when taken in isolation, each of these runs must be executed for each of the possible *decision rules* that must be specified for all other branches of the event node (note that in computing the number of runs *r(c)* required under node *c*, we have already considered the decision rules lying within the subtree rooted at *c*.) Thus $r(e)=\sum_{c\in child(e)} r(c) \prod_{b\in child(e)\&b\neq c} s(b)$.
- **Decision node.** For a decision node, a particular decision rule must specify one and only one decision to be made. Because these decisions are mutually exclusive, a given run associated with a particular one of the decisions *c* (i.e. lying within the subtree rooted at *c*) needs only to be executed for a subset of decision rules – namely, decision rules that happen to specify that particular decision *c* as the outcome of this decision node. As a result, when we consider the number of runs required by the decision node as a whole, there are no combinatorial effects between the count of runs required by each child of that node. Specifically, the total number of runs required for evaluating this decision node is just the sum of

the number of runs required to evaluate each possible decision of the decision node in turn. Thus $r(d)=\sum_{c\in child(d)} r(c)$.

Just as was the case for the recurrences giving the count of strategies associated with a decision tree, the number of runs required will vary with the stricture of the tree. To give a sense of run count scaling, we consider again the complete decision tree introduced above, in which decision nodes and event nodes form alternating layers, where the parents of the terminal nodes are all decision nodes, and where decision nodes are associated with *d* children and event nodes with *c*. Once again slightly abusing the notation above by overloading *r* to as a function of node height within a complete tree, we can specify the number of runs required to conduct normal form analysis of a tree of height *h* (treating the terminal nodes as lying at height 0) as follows:

$$r_N(h)=c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h,2)}$$

As for the strategy count, we have constructed a straightforward inductive proof of the correctness of this formula. This proof is included within the appendix to this paper.

5.2 Performance for Preferred Decision Rule Identification Using a Decision Tree

Consider now the case in which we identify a preferred decision rule using a decision tree and backwards induction (in short, make use of extensive form analysis). As discussed in a previous section, we assume that the performance is dominated by the system dynamics runs required for scenario evaluation and ignore the cost of rollback. In this case, because rollback proceeds after evaluating each of the terminal nodes, the number of terminal node executions is simply equal to the number of terminal nodes. While we can write this mathematically as $\left|n \in Nodes(t) \mid IsTerminalNode(n)\right|$ (where *t* is the entire tree), for consistency with the above we describe a recursive procedure for calculating it. Given the root node *n*, *r*(*n*) will be the count of terminal node executions required to identify the preferred decision rule using backwards induction.

- **Terminal node.** $r(t)=1$.
- **Decision node.** $r(d)=\sum_{c\in child(d)} r(c)$.
- **Event node.** $r(e)=\sum_{c\in child(e)} r(c)$

In order to provide a concrete sense of performance, consider again the case of a complete tree of height *h* with alternating layers of decisions and event nodes, where each decision and event node has *d* and *c* children respectively, where the terminal nodes are considered to lie at height 0, and where the parents of the terminal nodes are decision nodes. In this case, the number of terminal nodes (runs) requiring evaluation is

$$r_E(h)=c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil}$$

As for the cases above, we have demonstrated the correctness of this formula via inductive proof, although details are omitted here.

5.3 Comparisons

	Runs	Runs per Terminal Node
Normal Form	$c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h,2)}$	$c^{\mathrm{mod}(h-1,2)} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)-\left\lceil \frac{h-1}{2} \right\rceil+\mathrm{mod}(h,2)}$
Extensive Form	$c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil}$	1

Table 2 Closed form expressions for the count of system dynamics executions required to identify a preferred strategy for a complete tree of height h, with alternating rows of event nodes (each with c children) and decision nodes (each with d children), where all the parents of the terminal nodes are decision trees.

Tree Height	# Decision Rules	Extensive form run count $r_E(\cdot)$	Normal form run count $r_N(\cdot)$
5	128	32	512
6	16,384	64	131,072
7	32,768	128	262,144
8	1,073,741,824	256	17,179,869,184
9	2,147,483,648	512	34,359,738,368

Table 3: Summary of decision rule count for complete binary trees of the form noted above, and a comparison of the number of system dynamics model runs required using normal form and extensive form analysis.

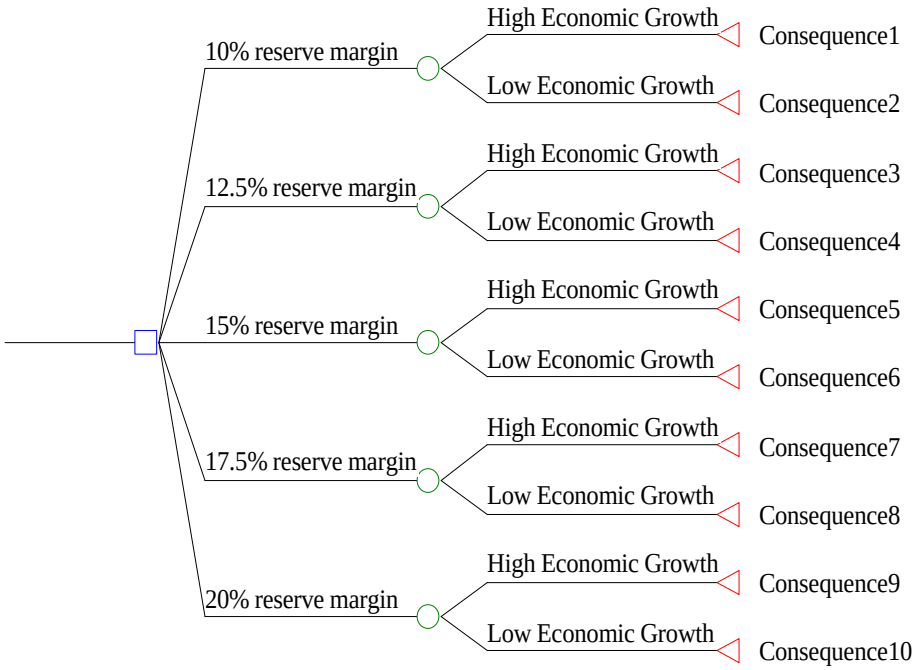
The performance disadvantages of the normal form approach can be readily appreciated from Table 2 and Table 3. As seen from Table 2, for the normal form approach the number of runs required is super-exponential in the height of the tree – the exponent on the d term rises geometric with the height of the tree. The extensive form approach is substantially faster; because the number of terminal nodes in the tree rises geometrically in the height of the tree, the number of terminal nodes whose evaluation is required necessarily remains geometric in the tree height. However, the use of backwards induction eliminates the super-exponential component in the number of runs required to evaluate the tree.

Performance was also examined on random trees. Such trees are associated with asymmetric structure and stochastic depth and branch counts for both event and decision nodes. For such trees, the gap in performance remained substantial but appeared somewhat less dramatic than for full trees. A later paper will describe these results in greater detail.

5.4 Splitting Decision Rules between System Dynamics and Decision Trees

It is worth remarking on a technique for reducing the performance costs of identifying a preferred policy (decision rule) by decision rule order reduction. Some historic models which have explicitly used both decision trees and system dynamics models have avoided

the expenses associated with exploring a broad set of decision rules by factoring the choice of decision rules into two parts, with one component of a decision rule represented in the decision tree and the other component within the system dynamics model.
Using this technique, the modeler first identifies certain particularly competitive subspace of the space of possible decision rules. Frequently this subspace is a thin manifold of decision rules that can be specified using just a few parameters. Such parameterized decision rules (and the decisions that make them up) are then encoded within the system dynamics model. The parameters of those system dynamics decision rules are then set by explicit decisions within the decision tree. In this way, a very large number of particular decisions is reduced to just the few choices of parameter values needed to identify one of member of the subspace of highly competitive decision rules.



For example, consider a decision tree for determining electrical capacity buildout over time, where the timing and technology of buildout may be affected by the observed (uncertain) electrical demand and fuel prices. This tree could be represented using a successive series of choices as to whether to buildout and (if so) which technology to employ. Using the alternate technique described in this section, the modeler can factor the decision rule into two pieces.

- The system dynamics model can be used to represent the moment-to-moment decision rule determining whether to build out given a certain desired *reserve margin.* When a buildout is required, the implicit rule may further dictate that any new expansion uses the cheapest available technology at the time of buildout.
- The decision tree can employ a (root) decision node for the choice of reserve margin. Chance nodes in the decision tree can dictate factors such as the rate of economic growth, changes in fuel prices, and other factors.

By cleverly partitioning decision making between the system dynamics model and the decision tree such techniques can offer substantial value in reducing the size of the space of potential decision rules that must be evaluated. For the example cited here, this reduces the problem from the selection of an optimal sequence of build-or-not-build decisions (and choice of technologies) to the selection of a single initial decision of reserve margin.

While very helpful for certain classes of problems, this approach does not escape from the fundamental tradeoffs analyzed within this paper. In particular, such split-decision-rule approaches exhibit several limitations, including two listed below:

- **Difficulty of identifying the competitive (parameterized) set of decision rules to be considered.** In many complex policy choice problems, the subset of policy options (decision rules) most likely to be competitive is not obvious *a priori*. As a result, it may not be possible to parameterize such rules and encapsulate them within the system dynamics model. In a rich decision tree that represents many types of events and decisions, it may be highly challenging to identify the desirable subset of decision rules to that should be implicitly specified. Even for the sample example discussed here, the most appropriate choice of technology may depend on the reserve margin due to economies of scale and differing length of time required to commission a plant. Confining analysis to a very a small subset of rules (e.g. rules in which the technology which is currently least expensive is employed for build-out) can mss opportunities for identifying such relationships.
- **Hidden re-evaluated substructure.** While restricting the set of policy options being considered can significantly reduce the size of a decision tree, shifting the evaluations of decision rules to the decision tree obscures the common substructure among different scenarios. While the split-decision-rule approach will in some case offer a compelling advantage, the presence of hidden substructure lessens the computational advantages of such approach. For the example discussed above, the any of a wide range of operating margins may lead to identical behavior in response to a variety of particular scenarios – e.g. rates of economic growth. When the choice to expand or not to expand is explicitly represented in the decision tree, this substructure is exposed and can be exploited through extensive form analysis. When the decision as to whether or not to expand is represented only in the system dynamics model (parameterized just the choice of an operating margin within the decision tree), this common substructure remains hidden. The computational costs of hiding this substructure grow considerably more apparent when one must consider other decisions, such as the choice of sub-technology, the lumpiness of the investment, etc. While not shown here, decision trees can be easily extended to handle an arbitrary number of types of choices.

In summary, techniques that divide decision rules between the decision tree and the system dynamics model present an attractive means of lessening the computational burden of decision rule selection in those limited cases in which we can identify a highly attractive subspace of decision rules a priori and when the common substructure among the decision rules within this subspace is sufficiently limited to allow it to be re-evaluated for each decision rule without overriding costs.

5.5 Conclusions

This paper has provided a brief introduction to hybrid modeling techniques that combine system dynamics with decision analysis. The use of decision analytic techniques in conjunction with system dynamics offer a wide variety of advantages, including modeling clarity, rapid sensitivity analysis on event probabilities, access to sophisticated decision analytic tools, nimble interpolation between the results of examined policy choices, and "offline" policy analysis for certain important classes of policies. This paper has focused on the *performance* advantages of such hybrid modeling techniques for policy selection. These advantages arise from the ability of hybrid models to take advantage of the common substructure of decision rules by the use of dynamic programming to select preferred decision rules. The extent of the performance advantages is sufficiently large that hybrid techniques offer the opportunity to perform substantially more detailed policy analysis than would be possible using system dynamics alone. By partitioning decision rules between the decision tree and system dynamics model, additional performance tuning is possible.

While realizing the integration between system dynamics and decision analysis is currently tedious, we believe that a software framework that achieved this integration would confer significant modeling advantages. Forthcoming papers will describe the design and operation of such a framework as well as the generality and additional advantages of the hybrid approach.

5.6 Acknowledgements

The author wishes to acknowledge several people who played an important role in contributing related ideas and in making this research possible. Gordon Kaufman deserves special thanks as an early and enthusiastic proponent of prospects for combining decision analytic and system dynamics approaches, even in the face of pronounced skepticism by other modelers. Gordon's devotion of time and discussion to the topic was seminal in encouraging the development of the methodology, and he was prescient in intuiting many important aspects of the framework that emerged. Andy Ford provided very helpful comments, and was particularly helpful in highlighting out the relevance and value of some previously overlooked past work that directly suggested the value of the class of techniques discussed in Section 5.4. Finally the author is thankful to the audience members of the System Dynamics Winter Conference and especially Itsung Tsai and Charlie Lertpattarapong, who helped provide comments on closely related work.

References

[Osgood & Kaufman, 2005] Osgood, N. and Kaufman, G. *Decision Analysis and System Dynamics*. In preparation for submission to *System Dynamics Review*.

Appendix – Proofs of Closed Form Solutions to Recurrences

This appendix contains a proof of the formulas for the count of decision rules within the tree and the count of runs required to evaluate those decision rules using normal form analysis. In both cases, we assume a complete (full) tree of alternating layers of decisions and event nodes, where the decision nodes (located at odd heights of the tree) have *d* children and the event (choice) nodes (located at even heights of the tree at all levels above height 0) have *c* children. Finally, the terminal nodes are considered to be at height 0 of the complete tree.

The basic relations to be proven are that for the count of decision rules in the tree, expressed as a function of the height *h* in the tree:

$$s(h)= d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor +1}-c^{\mathrm{mod}(h-1,2)}}{e-1}\right)}$$

and that for the count of runs required when the tree is evaluated using normal form analysis. Once again, this is expressed as a function of the height of the node in the tree:

$$r(h)= c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h,2)}$$

Because the formula for the count of required runs depends on the formula for the count of decision rules, we will first prove the strategy formula, and then turn to the formula for the count of runs.

Proof of Formula for Normal Form Strategy Count

Base case: Terminal nodes (h=0)

Recall: A terminal nod is treated as having a single (empty) strategy associated with it.

Proof of formula:

$$s(h)=s(0)= d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor +1}-c^{\mathrm{mod}(h-1,2)}}{c-1}\right)} = d^{\left(\frac{c^{\lfloor 0 \rfloor +1}-c^{1}}{c-1}\right)} = d^{\left(\frac{c^{1}-c^{1}}{c-1}\right)} = d^{0} = 1 \qquad \square$$

Inductive case:

Decision nodes:

Since decision nodes are assumed to lie at *odd* heights h+1 is odd (i.e. mod(h+1,2)=1). Also worth remarking is that $\left\lfloor \frac{h+1}{2} \right\rfloor = \left\lfloor \frac{h}{2} \right\rfloor$.

Consider that formula is true for height ≤ h, prove for h+1.

For the case of a *decision* node, a specific decision rule need only specify exactly one possible outcome of this decision – but will also have to

specify particular decisions required by any decisions beneath this branch (but not under any other branch). The set of decision rules that are possible for the subtree rooted at *d* is simply the union of the set of decision rules that are possible for the subtrees rooted at each particular child of *d*, with one twist: Each decision rule from child *c* is extended with an extra element that specifies that the parent (decision) node has made choice c. The count of particular decision rules associated with this node is thus the same as the cardinality of the union of the set of decision rules associated with the children of the decision node. i.e. $s(d) = \sum_{c \in child(d)} s(c)$

Reasoning inductively, we have

$$\sum_{i \in child(d)} d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{c-1} \right)} = dd^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^1}{c-1} \right)} = d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c}{c-1} \right) + 1} = d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c}{c-1} + \frac{c-1}{c-1} \right)}$$

$$= d^{\left(\frac{c^{\left\lfloor \frac{h+1}{2} \right\rfloor + 1} - 1}{c-1} \right)}$$

But this is just $d^{\left(\frac{c^{\left\lfloor \frac{h+1}{2} \right\rfloor + 1} - c^{\mathrm{mod}((h+1)-1,2)}}{c-1} \right)}$, which proves the inductive case for decision nodes (i.e. the odd levels of the tree).

Event nodes:

Since even t nodes are assumed to lie at *even* heights h+1 is even (i.e. mod(h+1,2)=09). Also worth remarking is – in contrast to the case for decision nodes – $\left\lfloor \frac{h+1}{2} \right\rfloor = \left\lfloor \frac{h}{2} + 1 \right\rfloor = \left\lfloor \frac{h}{2} \right\rfloor + 1$.

For a subtree rooted at an event node, it is not possible for a decision maker to determine which event will occur. As a result, a given decision rule will have to *independently* specify whatever decisions require specification *for each possible outcome* of this chance event. Because the decision rule can independently specify any decisions required for each event outcome, the set of possible decision rules required under node *e* is the set of possible *combinations* (cartesian product) of the decision rules possible for the subtrees associated with each possible outcome (child) of the event. The size of that set is just the *product* of the number of possible decision rules required for each child of the event node, i.e. $\prod_{i \in child(e)} d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{e-1} \right)}$.

Consider that formula is true for height ≤ h, prove for h+1.

We have s(h+1)= $\prod_{i\in child(e)} d^{\left(\frac{c^{\left\lfloor\frac{h}{2}\right\rfloor+1}-c^{\mathrm{mod}(h-1,2)}}{e-1}\right)}$

$$= \prod_{i\in child(e)} d^{\left(\frac{c^{\left\lfloor\frac{h}{2}\right\rfloor+1}-c^{\mathrm{mod}(h-1,2)}}{c-1}\right)} = \left(d^{\left(\frac{c^{\left\lfloor\frac{h}{2}\right\rfloor+1}-c^{0}}{c-1}\right)}\right)^{c} = d^{c\left(\frac{c^{\left\lfloor\frac{h}{2}\right\rfloor+1}-1}{c-1}\right)} = d^{\left(\frac{c^{\left\lfloor\frac{h}{2}\right\rfloor+2}-c}{c-1}\right)}$$

$$= d^{\left(\frac{c^{\left\lfloor\frac{h+1}{2}\right\rfloor+1}-c^{1}}{c-1}\right)} = d^{\left(\frac{c^{\left\lfloor\frac{h+1}{2}\right\rfloor+1}-c^{\mathrm{mod}((h+1)-1,2)}}{c-1}\right)} \quad \square$$

This proves the inductive case for event nodes (i.e. the even levels of the tree above level 0).

By virtue of having proven the formula for the base case and even and odd levels of the tree, we have proven the formula for strategy count for the entire tree.

We now turn to proof of the formulas for the count of runs required for the normal form analysis of the tree.

Proof of Formula for Normal Form Run Count

Inductive proof of formulas for run count; $r(h) = c^{\left\lfloor\frac{h}{2}\right\rfloor} d^{(1+c)c^{\left\lfloor\frac{h}{2}-1\right\rfloor}+c\left(\frac{c^{\left\lfloor\frac{h}{2}-2\right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h,2)}$

Base case (height 0)

For height 0, we have terminal nodes.

Each such node requires only on a single run in normal-form analysis $\Rightarrow$s(0)=1

The formula gives

$$r(0) = c^{\left\lfloor\frac{0}{2}\right\rfloor} d^{(1+c)c^{\left\lfloor\frac{0}{2}-1\right\rfloor}+c\left(\frac{c^{\left\lfloor\frac{0}{2}-2\right\rfloor}-1}{c-1}\right)+\mathrm{mod}(0,2)} = c^{0} d^{(1+c)c^{-1}+c\left(\frac{c^{-2}-1}{c-1}\right)+0} = d^{c^{-1}+1+\frac{-1}{c}\left(\frac{c^{2}-1}{c-1}\right)}$$

$$= d^{c^{-1}+1+\frac{1}{c}(c+1)} = d^{c^{-1}+1+\frac{-1}{c}(c+1)} = d^{c^{-1}+1+-1-c^{-1}} = d^{0} = 1 \quad \square$$

Inductive case (height ≥ 0)

We consider in turn the case of *decision nodes* (located at odd heights) and *event* nodes (located at event heights > 0)

Decision nodes (odd heights)

For a decision node, a particular decision rule must specify one and only one decision to be made. Because these decisions are mutually exclusive, a given run associated with a particular one of the decisions *c* (i.e. lying within the subtree rooted at *c*) needs only to be executed for a subset of

decision rules – namely, decision rules that happen to specify that particular decision *c* as the outcome of this decision node. As a result, when we consider the number of runs required by the decision node as a whole, there are no combinatorial effects between the count of runs required by each child of that node. Specifically, the total number of runs required for evaluating this decision node is just the sum of the number of runs required to evaluate each possible decision of the decision node in turn. Thus $r(d)=\sum_{c\in child(d)} r(c)$

Assume that we are at a decision node at height h+1, and that the formula has held for all heights ≤ h. We now wish to demonstrate that the formula holds for height h + 1.

We note that because decision nodes are located at *odd* heights in the tree, h+1 is odd and h is even (i.e. mod(h+1,2)=1 and mod(h,2)=0; moreover $\left\lfloor \frac{h+1}{2} \right\rfloor = \left\lfloor \frac{h}{2} \right\rfloor$).

Now

$$r(h)=\sum_{i\in child(d)} r(h-1)$$

$$=\sum_{i\in child(d)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h,2)}$$

$$=\sum_{i\in child(d)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+0}$$

$$=c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1}\right)+1}= c^{\left\lfloor \frac{h+1}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h+1}{2}-1 \right\rfloor}+c\left(\frac{c^{\left\lfloor \frac{h+1}{2}-2 \right\rfloor}-1}{c-1}\right)+\mathrm{mod}(h+1,2)} \quad \square$$

Thus the result matches the formula for r(h+1). We have thus proven the correctness of the formula for *odd* heights of the tree.

Event nodes

For an event node a given decision rule must specify values for decisions associated with all different possible chance eventualities associated with the event. There is a combinatorial effect here in terms of the number of runs required: If a given branch *c* of the event node itself requires *m* distinct runs to evaluate when taken in isolation, each of these runs must be executed for each of the possible *decision rules* that must be specified for all other branches of the event node (note that in computing the number of runs *r(c)* required under node *c*, we have already considered the decision rules lying within the subtree rooted at *c*.) For a uniform tree where h+1 is the height $r(h+1)=\sum_{i\in child(e)} r(h) \prod_{b\in child(e)\&b\neq c} s(h)$.

We note that because event nodes are located at *even* heights in the tree, h+1 is even and h is odd (i.e. mod(h+1,2)=0 and mod(h,2)=10; moreover $\left\lfloor \frac{h+1}{2} \right\rfloor = \left\lfloor \frac{h}{2}+1 \right\rfloor = \left\lfloor \frac{h}{2} \right\rfloor + 1$).

Assume that we are at a event node at height h+1, and that the formula has held for all heights ≤ h. We now wish to demonstrate that the formula holds for height h + 1. Now

$$r(h+1) = \sum_{i \in child(e)} r(h) \prod_{b \in child(e) \& b \neq c} s(h)$$

$$r(h+1) = \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + \mathrm{mod}(h,2)} \prod_{b \in child(e) \& b \neq c} d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{c-1} \right)}$$

$$= \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + \mathrm{mod}(h,2)} \left(d^{\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{c-1} \right)} \right)^{c-1}$$

$$= \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + \mathrm{mod}(h,2)} d^{(c-1)\left(\frac{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^{\mathrm{mod}(h-1,2)}}{c-1} \right)}$$

$$= \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + 1} d^{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - c^0} = \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + 1} d^{c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} - 1}$$

$$= \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right) + 1 - 1} = \sum_{i \in child(e)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{(1+c)c^{\left\lfloor \frac{h}{2}-1 \right\rfloor} + c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} + c\left(\frac{c^{\left\lfloor \frac{h}{2}-2 \right\rfloor}-1}{c-1} \right)}$$

Recognizing that there are c children of t he event node, we have

$$= c\left(c^{\lfloor \frac{h}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h}{2}-1 \rfloor}+c^{\lfloor \frac{h}{2} \rfloor+1}+c\left(\frac{c^{\lfloor \frac{h}{2}-2 \rfloor}-1}{c-1}\right)}\right)$$

$$= c^{\lfloor \frac{h}{2} \rfloor+1} d^{(1+c)c^{\lfloor \frac{h}{2}-1 \rfloor}+c^{\lfloor \frac{h}{2} \rfloor+1}+c\left(\frac{c^{\lfloor \frac{h}{2}-2 \rfloor}-1}{c-1}\right)} = c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{-1}c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2} \rfloor}+c\left(\frac{c^{-1}c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{c^{-1}c^{\lfloor \frac{h+1}{2}-1 \rfloor}+cc^{-1}c^{\lfloor \frac{h+1}{2}-1 \rfloor}+cc^{\lfloor \frac{h+1}{2}-1 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c^{-1}c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{cc^{\lfloor \frac{h+1}{2}-1 \rfloor}+cc^{-1}c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{-1}c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}+\frac{-c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c^{-1}c^{\lfloor \frac{h+1}{2}-2 \rfloor}}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{cc^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}+\frac{-cc^{\lfloor \frac{h+1}{2}-3 \rfloor}+c^{\lfloor \frac{h+1}{2}-3 \rfloor}}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{(c+1)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}+\frac{-(c-1)c^{\lfloor \frac{h+1}{2}-3 \rfloor}}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}-c^{\lfloor \frac{h+1}{2}-3 \rfloor}\right)} = c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)-cc^{\lfloor \frac{h+1}{2}-3 \rfloor}}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c^{\lfloor \frac{h+1}{2}-2 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)-c^{\lfloor \frac{h+1}{2}-2 \rfloor}} = c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)}$$

$$= c^{\lfloor \frac{h+1}{2} \rfloor} d^{(1+c)c^{\lfloor \frac{h+1}{2}-1 \rfloor}+c\left(\frac{c^{\lfloor \frac{h+1}{2}-2 \rfloor}-1}{c-1}\right)+\mathrm{mod}(h+1,2)}$$ □

Thus the result matches the formula for r(h+1), thereby proving the correctness of the formula for e*ven* heights of the tree.

The formula for the number of runs in the tree is thereby proven for the base case and the inductive case for even and odd heights of the tree. This proves it the formula for all heights of the complete tree.

Proof of Formula for Count of Extensive Form Runs

Base Case (Terminal nodes, h=0)

Here, we have only a single terminal node, which implies $r_E(0)=1$

Formula $c^{\lfloor 0 \rfloor} d^{\lceil 0 \rceil} = 1$ □

Inductive Case

Decision Nodes

We are at height h+1.

Here we have h+1 odd, h even => $\left\lfloor \frac{h}{2} \right\rfloor = \left\lfloor \frac{h+1}{2} \right\rfloor$, $\left\lceil \frac{h+1}{2} \right\rceil = \left\lceil \frac{h}{2} \right\rceil + 1$

Assume true for heights ≤h, now prove true for height h+1

Here we have the count of runs required

$$\sum_{i \in child(d)} r(h) = \sum_{i \in child(d)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil} = d\left(c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil} \right) = c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil + 1} = c^{\left\lfloor \frac{h+1}{2} \right\rfloor} d^{\left\lceil \frac{h+1}{2} \right\rceil} \quad \square$$

This proves the formula for odd layers of the decision tree.

Event Nodes

We are at height h+1.

Here we have h+1 even, h odd => $\left\lfloor \frac{h+1}{2} \right\rfloor = \left\lfloor \frac{h}{2} \right\rfloor + 1 =$, $\left\lceil \frac{h+1}{2} \right\rceil = \left\lceil \frac{h}{2} \right\rceil$

Assume true for heights ≤h, now prove true for height h+1

Here we have the count of runs required

$$\sum_{i \in child(c)} r(h) = \sum_{i \in child(c)} c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil} = c\left(c^{\left\lfloor \frac{h}{2} \right\rfloor} d^{\left\lceil \frac{h}{2} \right\rceil} \right) = c^{\left\lfloor \frac{h}{2} \right\rfloor + 1} d^{\left\lceil \frac{h}{2} \right\rceil} = c^{\left\lfloor \frac{h+1}{2} \right\rfloor} d^{\left\lceil \frac{h+1}{2} \right\rceil} \quad \square$$

This proves the formula for even layers of the decision tree.

We have now proved the inductive formula for all heights of the tree greater than 0.

We have proven the correctness of the formula for run count for all heights of the tree. The formula is thus correct.