Fusion in Card Collecting Games: A Probable Outcome

Lindsay Bradley, Winthrop University, bradleyl4@winthrop.edu
Emili Moan, North Carolina State University, evmoan@ncsu.edu
Zoe Vernon, Washington University in St. Louis, zoe.vernon@wustl.edu
Advisor: Kristen Abernathy, Winthrop University, abernathyk@winthrop.edu

June 30, 2017

1 Abstract

Card Collecting Games (CCGs), as well as many games in other genres, often employ a mechanic referred to as gacha-fuse-evolve where players randomly draw items with different levels of rarity (common, uncommon, and rare) that can be fused and evolved to create stronger items. With the free-to-play model that many online companies use, it is important that CCG developers keep the game easy enough that players want to continue to play but difficult enough that players want to spend money to better their experience. To achieve this, developers need to ensure fusions occur often enough to keep the non-paying players engaged, but seldom enough to entice players to purchase additional fusion opportunities. For this project, we explore the probability of players drawing four different types of fusion (unique fusion, quad-fusion, evolutionary trees, and recipe fusion) in a given time period. We also run a sensitivity analysis to determine which parameters - deck size, number of rare cards, or length of play - are most sensitive. Finally, we create a C++ program to run simulations and verify the results of the unique and quad-fusions probabilities.

This research was completed as part of participation in the Preparation for Industrial Careers in the Mathematical Sciences (PIC Math) program. Support for this Mathematical Association of America (MAA) and Society for Industrial and Applied Mathematics (SIAM) program is provided by the National Science Foundation (NSF grant DMS-1345499).

2 Introduction

With the introduction of mobile gaming on phones and tablets, the video game industry is in the process of transitioning from the traditional business model of buying a game with one initial payment to the free-to-play model. In the free-to-play model, a game is free to download but will contain advertisements or in-app purchases in order to make profit. The key to the free-to-play model is that the game is entertaining enough to make a player continue playing and challenging enough to ensure players purchase items in the application. In our paper we consider a genre of games called Card Collecting Games (CCGs).

2.1 Background and Terminology

Many game developers use a model known as gacha-fuse-evolve. Gacha is short for gachapon, referring to the toy capsules one gets from coin dispensing machines. Fuse is abbreviated for fusion and evolve for evolution. In this model, players randomly choose items that are of different levels of rarity - often common, uncommon, and rare. The items can then be combined in various ways, such as through fusion or evolution, in order to make more powerful items. For the purpose of our paper, the items in the gacha-fuse-evolve model will be cards.

Definition *Fusion* occurs when two identical cards are combined to create a more powerful card.

For example, two man cards could be combined to create a warrior card.

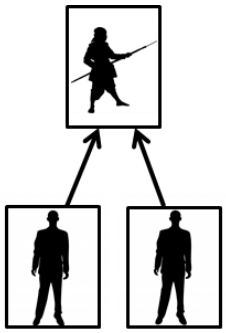


Figure 1: An example of fusion.

Definition *Evolution* is when an even number of identical cards are combined to reach higher and higher levels.

For example, two man cards could be combined to create a warrior and two warrior cards could be combined to create a knight.

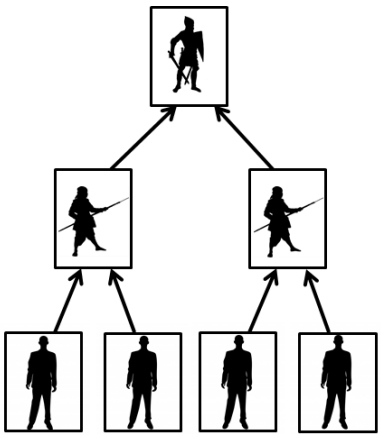


Figure 2: An example of evolution.

2.2 The Problem

The traditional gaming model involves buying a game with a single initial payment. In this model, every aspect of the game is given to players at the time of purchase. With the introduction of mobile gaming on phones and tablets, the video game industry is in the process of transitioning from this traditional model to a *free-to-play model*.

Definition In a *free-to-play* video game model, players are able to download the game for free. The game will then contain advertisements and in-app purchases in order to make a profit.

In-app purchases can be purely aesthetic. These can include wallpapers or skins/outfits for the characters. The purchases can also improve the players' scores or gaming experiences. Purchases such as these include gaining more lives or gaining certain skills that players who are not paying will not be able to obtain. There are multiple considerations to be made when designing a free-to-play game.

- The game must be entertaining enough that it will draw in new players. This is not difficult since the game is free.
- The game must be easy enough that there is no real learning curve and almost anyone can play it.
- The game must be difficult enough that players are willing to spend money in order to make the game more entertaining or improve their scores.

In the past, only "hardcore" gamers were interested in card collecting games, the most well-known being *Magic: The Gathering* and *Pokémon* (*1, 2*). Only recently, with the development of the free-to-play model of these games, have titles such as *Reign of Dragons*, which was a popular Facebook game, taken the forefront of the gaming world (*3*). The free-to-play model has caught the attention of a variety of players that would not be considered hardcore gamers. As we discussed above, in these free-to-play games, it is important to keep the game easy enough that players want to continue to play but difficult enough that players want to spend their money to better their experience.

(a) Magic the Gathering

(b) Reign of Dragons

In card collecting games, the gacha-fuse-evolve model is often used to make a more entertaining experience for the players. Here, the developer will create cards of different rarity levels. The player will download the game and receive some beginning cards. The gamer may receive a card after a certain amount of time has passed or after he has completed certain tasks within the game. Thus, the developer must decide how many cards of each type of rarity should be in the deck and the number of cards that a player can draw in a certain time period. The developer can then

determine if the game is too easy or too difficult for players and create different ways for players to pay for a better gaming experience. For example, perhaps the player receives one new card a day, then he could purchase another card or purchase a token that increases the probability of drawing cards of certain rarities.

2.3 Approach

We will present formulas that can be used by developers of card collecting games employing the gacha-fuse-evolve model to determine the probability of players completing certain types of fusions and evolutions within a given time period.

We make various assumptions about the decks of cards:

- Each card in a deck of m cards is unique. There are three levels of rarity - common, uncommon, and rare. If c is the number of common cards in the deck, u the number of uncommon, and r the number of rare, then $m = c + u + r$ and $c > u > r$.
- The formulas will consider a deck with replacement, and each card selected is replaced with an identical copy of the selected card. This is because gaming through the free-to-play model is transitioning almost exclusively to electronic platforms, and electronic decks can be easily programmed to replace cards immediately after they are selected.
- Players will perform fusions and evolutions whenever possible.
- The player is allowed to choose one card per day.

We will explore four types of card combinations - fusion, quad-fusion, evolutionary trees, and recipes.

Definition (Modification of the Fusion Mechanic) *Fusion* is the combination of two identical rare cards to create a unique card.

Definition *Quad-Fusion* is the combination of two unique cards (identical or not) to create a super-unique card.

Definition An *evolutionary tree* is a combination of different levels of fusion involving an even number of cards.

Definition A *recipe* is a combination of non-identical cards to create a new, more powerful card.

We will present formulas that can be used, over given periods of time with the different mechanics discussed above, by game developers to decide the deck size, number of rare cards, and number of draws necessary to obtain desired probabilities of their players fusing cards. This may help game developers ensure balance within their game by preventing them from making small changes that may have large impacts on the chance players have of moving forward in the game.

For game developers, the goal is to allow the player to fuse exactly once over a given time period (for example, per month). This allows the non-paying player to progress in the game, but motivates players to spend money to progress more quickly (for example, the player could purchase more draws which would allow them to fuse more quickly than once a month or fuse more than once in a month). After considering the probabilities that a given fusion would occur exactly once in a time period, we quickly realized it would be more beneficial to game developers if they knew when the majority of their players would be able to fuse. For this reason, we also explored the probability of creating at least one fusion in a given time period. The comparison of these two probabilities provides useful information that may be lost when only considering the probability of creating exactly one fusion.

It should be noted that to allow for flexibility in game developing, the parameter value n in all formulas below simply refers to the number of draws a player has made. Because of the simplifying assumption above, a player draws a card once per day, and so our formulas will give us the probability of completing a fusion in n days. However, if developers wish to allow players more draws per day, they can simply read our probabilities in terms of number of draws rather than number of days.

In Section 3.1 we present the formula for performing exactly and at least one fusion in a certain time period. We also provide an example of the formula. In Section 3.2 we present the formula for performing exactly and at least one quad-fusion in a time period and give an example. In Section 4, we compare the formulas for fusion and quad-fusion.

In Section 5 we present the formula for obtaining an evolutionary tree in a certain time period. In Section 6, we present formulas for completing two and three card recipes, along with an illustrative example. In Section 7, we run a sensitivity analysis of the fusion, quad-fusion, and two-card recipe formulas to determine the variables that are the most sensitive to change. In Section 8, we discuss our C++ program created to run simulations and test the fusion and quad-fusion formulas. Finally, in Section 9 we present conclusions and discuss our results and in Section 10 we discuss potential future questions.

3 Probability Formulas

3.1 Fusion

3.1.1 Exactly

We begin our analysis by developing probability formulas to determine the probability that a player will complete a fusion in a given time period. In each fusion case, we assume that if the player has the cards necessary to obtain a fusion that they will do so. Thus, we only have to determine the probability that the player will draw the cards needed for the fusion. So, we can use a binomial distribution (*4*). The parameters in our formulas are the number of cards in the deck m, the total number of cards chosen n, the number of rare cards chosen k, and the number of rare cards in the deck r. We use the notation $(r)_k$ to denote the falling factorial, defined as

$$(r)_k = r(r-1)\cdots(r-(k-1)). \tag{1}$$

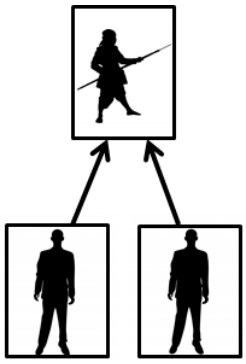


Figure 4: Fusion of two rare cards.

We first explore the probability of completing a fusion. Recall that fusion occurs when two identical rare cards are combined to get one unique card. Let X be the number of unique cards. We consider the probability of getting exactly one unique card in a certain time period:

$$P(X=1) = \frac{1}{m^n}\left[\sum_{k=2}^{r+1}\left[\binom{n}{k}\binom{k}{2}(r)_{k-1}(m-r)^{n-k}\right] + \sum_{k=3}^{r+2}\left[\binom{n}{k}\binom{k}{3}(r)_{k-2}(m-r)^{n-k}\right]\right]. \tag{2}$$

To understand this formula, we first note that to calculate this probability we divide the number of ways we can obtain one unique card in n days by the number of ways we can draw any combination of cards over n days. To compute the number of ways we can obtain exactly one unique card in n days, we notice that we can form exactly one unique card with two or three identical rare cards to complete a fusion. Let's first consider drawing two identical rare cards to form our fusion. We first determine the number of combinations of days on which we could draw the rare cards using $\binom{n}{k}$. We then place our rare cards on those days with the falling factorial $(r)_{k-1}$, accounting for the pair of rare cards

with $\binom{k}{2}$. We then place the remainder of our non-rare cards with $(m-r)^{n-k}$ and sum over the possible number of rare cards we could have chosen.

Since we could create exactly one unique card by choosing up to three of one rare card, and no more than one of the remaining rare cards, we repeat our process above but we consider choosing three of the same rare card to produce our unique card. We sum these two terms and divide by our sample space m^n to find our probability.

3.1.2 At Least

We also find the probability of getting at least one unique card in a certain time period:

$$P(X \geq 1) = 1 - \left[\frac{1}{m^n}\left[\sum_{k=0}^{r}\binom{n}{k}(r)_k(m-r)^{n-k}\right]\right]. \tag{3}$$

In order to do this, we find the probability of getting no unique cards and subtract this number from 1. In order to get no unique cards, we cannot have more than one of any rare card. So, we choose the k rare cards we can draw with $(r)_k$ and place them with $\binom{n}{k}$. We then place the remaining non-rare cards with $(m-r)^{n-k}$.

Because the probability of getting exactly one unique card is often extremely low, the probability of getting at least one unique card is sometimes more useful for game developers. This value shows the probability that the general population will have a unique card over a certain time period, allowing game developers to more appropriately select their deck sizes, the number of cards chosen, and the number of rare cards in the deck.

3.1.3 Example

We use an example to illustrate our formulas. When there are 60 cards in the deck, 10 rare cards in the deck, and the player chooses one card a day over a 30 day period, the probability that the player will choose exactly one unique card is :

$$P(X=1) = \frac{1}{60^{30}}\left[\sum_{k=2}^{10+1}\left[\binom{30}{k}\binom{k}{2}(10)_{k-1}(60-10)^{30-k}\right] + \sum_{k=3}^{10+2}\left[\binom{30}{k}\binom{k}{3}(10)_{k-2}(60-10)^{30-k}\right]\right] \approx 0.3981$$

Moreover, the probability of getting at least one unique card is ≈ 0.6227.

This tells us that on average 62.27% of the population will have a unique card after 30 days.

3.2 Quad-Fusion

3.2.1 Exactly

We now find the probability that a player obtains a super-unique card in a certain time period using quad-fusion. Recall that quad-fusion occurs when two unique cards, identical or not identical, are combined to get one super-unique card. Let Y be the number of super-unique cards. We consider the probability of getting exactly one super-unique card.

We can obtain a super-unique card with either two or three unique cards. Thus, there are fifteen cases that we must consider. For each case, we consider how many rare cards are involved in the fusion. Multipliers are used when the cards can be arranged differently (e.g. if we select 5 of one rare card and 2 of another, we multiply this case by 2). We then place the rare cards involved in the fusion over n days and place the remaining rare cards not involved in the fusion (these are denoted by k). We then place the remaining non-rare cards.

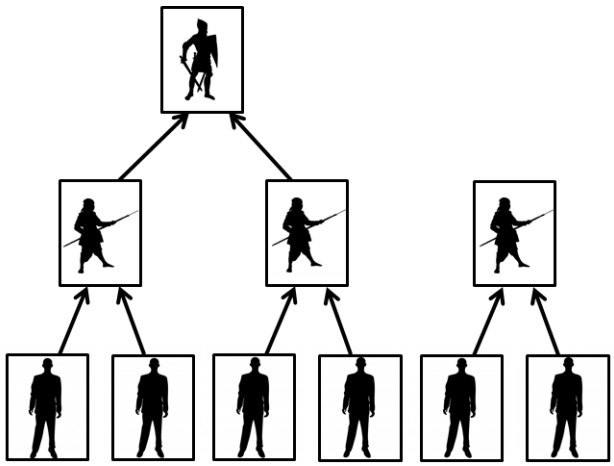


Figure 5: Quad Fusion using identical unique cards.

3.2.1.1 Identical Unique Cards In this subcase, we only consider super-unique cards created from identical unique cards. Because unique cards are made from rare cards, we can view the super-unique cases in terms of rare cards as well. We can have two or three of the same unique card and still create only one super-unique fusion. So, in this case, we can have between four and seven of the same rare card type. As before, m is the number of cards in the deck, n is the number of cards chosen (we will assume that the players choose 1 card a day for n days), r is the number of rare cards in the deck, k is the number of rare cards chosen, and l is the number of rare cards involved in the quad-fusion:

$$P(Y = 1 \text{ with identical cards}) = \frac{1}{m^n}\left[\binom{r}{1}\Big[\sum_{l=4}^{7}\binom{n}{l}\sum_{k=0}^{r-1}\binom{n-l}{k}(r-1)_k(m-r)^{n-l-k}\Big]\right] \tag{4}$$

We first find the number of ways in which we could choose the rare card that will be used in the fusion using $\binom{r}{1}$. We will choose between 4 and 7 of this card. We will denote this number by l. We then determine the number of combinations of ways that we could draw the l rare cards involved in the fusion over a period of n days.

Though we can have two or three of the unique cards we are using in the fusion (4 to 7 rare cards), in this case we can not have one of any other type of unique card (2 or 3 rare cards). In other words, we can have at most one of each type of rare card not involved in the fusion. We will denote the number of rare cards that we use that are not involved in the fusion by k. So, we determine the number of combinations of days in which we could draw those k cards over a period of $n-l$ days using $\binom{n-l}{k}$. We then determine that there are $(r-1)_k$ ways to place the k rare cards not involved in the fusion over the $n-l$ days and $(m-r)^{n-l-k}$ ways to place the non-rare cards on the remaining days. We sum over both the possible number of rare cards that are not involved in the fusion $0, 1, ..., r-l$ and the number of rare cards involved in the fusion $4, 5, 6, 7$. We then divide by the total number of ways that we could choose n cards from a deck of size m with repetition to find the probability.

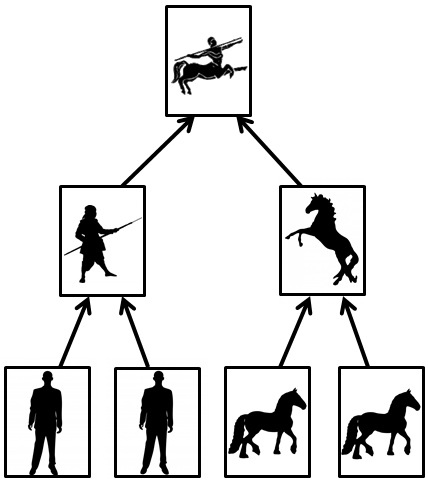


Figure 6: Quad Fusion using two different unique cards.

3.2.1.2 Two Different Unique Cards We can also combine two different unique cards to create a super-unique card. There are seven cases in which this occurs. Recall that we can have two or three unique cards involved in the fusion (4 to 7 rare cards). In the first case we have two of each rare card type, in the second case we have two of one rare card type and three of the other (we multiply this by 2 because we could have 2 of rare card type A and 3 of rare card type B or vice versa), and in the third case we have three of each rare card type. In the fourth case we have two of one card and four of the other, in the fifth case we have three of one card and four of the other, in the sixth case we have two of one card and five of the other, and in the seventh case we have three of one card and five of the other. We multiply each of these by two because of we have two rare card types. We use the sum rule to combine the cases:

$$
\begin{aligned}
&P(Y=1 \text{ with two different cards}) = \\
\frac{1}{m^n}\Bigg[&\binom{r}{2}\Big[\binom{n}{2}\binom{n-2}{2}\sum_{k=0}^{r-2}\binom{n-4}{k}(r-2)_k(m-r)^{n-4-k} \\
&+2\binom{n}{3}\binom{n-3}{2}\sum_{k=0}^{r-2}\binom{n-5}{k}(r-2)_k(m-r)^{n-5-k} \\
&+\binom{n}{3}\binom{n-3}{3}\sum_{k=0}^{r-2}\binom{n-6}{k}(r-2)_k(m-r)^{n-6-k} \\
&+2\binom{n}{4}\binom{n-4}{2}\sum_{k=0}^{r-2}\binom{n-6}{k}(r-2)_k(m-r)^{n-6-k} \\
&+2\binom{n}{4}\binom{n-4}{3}\sum_{k=0}^{r-2}\binom{n-7}{k}(r-2)_k(m-r)^{n-7-k} \\
&+2\binom{n}{5}\binom{n-5}{2}\sum_{k=0}^{r-2}\binom{n-7}{k}(r-2)_k(m-r)^{n-7-k} \\
&+2\binom{n}{5}\binom{n-5}{3}\sum_{k=0}^{r-2}\binom{n-8}{k}(r-2)_k(m-r)^{n-8-k}\Big]\Bigg]
\end{aligned}
\tag{5}
$$

We determine these formulas exactly like we determined the formula for a single unique card, except we must determine the number of combination of days to pick each type of rare cards used in the fusion. So we have two combinations (e.g. $\binom{n}{2}\binom{n-2}{2}$) instead of the one combination $\binom{n}{l}$.

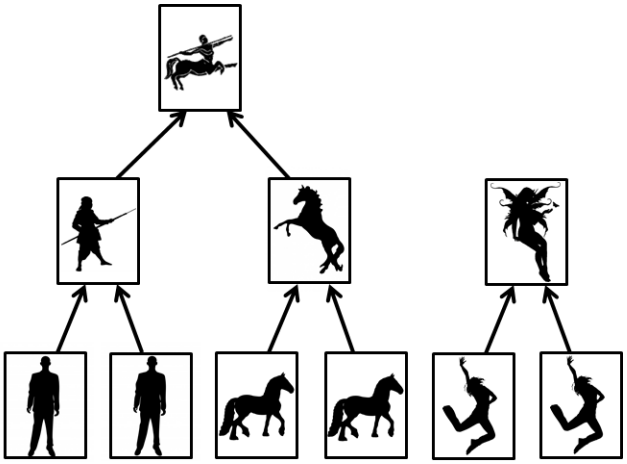


Figure 7: Quad Fusion using three different unique cards.

3.2.1.3 Three Different Unique Cards Finally, we can have three different unique cards contribute to the creation of a super-unique card. There are four cases in which this can occur. In the first case there are two of each of the three rare card types. In the second case there are two of one of the three rare card types and three of the other rare card types (we multiply this by 3 because there are three rare card types). In the third case there are three of one of the three rare card types and two of the other rare card types (we multiply this by 3 because there are three rare card types). In the fourth case there are three of each rare card type. We combine the cases using the sum rule:

$$
\begin{aligned}
&P(Y = 1 \text{ with three different cards}) = \\
&\frac{1}{m^n}\Bigg[\binom{r}{3}\Bigg[\binom{n}{2}\binom{n-2}{2}\binom{n-4}{2}\sum_{k=0}^{r-3}\binom{n-6}{k}(r-3)_k(m-r)^{n-6-k} \\
&\quad +3\binom{n}{3}\binom{n-3}{2}\binom{n-5}{2}\sum_{k=0}^{r-3}\binom{n-7}{k}(r-3)_k(m-r)^{n-7-k} \\
&\quad +3\binom{n}{3}\binom{n-3}{3}\binom{n-6}{2}\sum_{k=0}^{r-3}\binom{n-8}{k}(r-3)_k(m-r)^{n-8-k} \\
&\quad +\binom{n}{3}\binom{n-3}{3}\binom{n-6}{3}\sum_{k=0}^{r-3}\binom{n-9}{k}(r-3)_k(m-r)^{n-9-k}\Bigg]\Bigg]
\end{aligned}
\tag{6}
$$

We then use the sum rule to combine all of the cases.

$$
\begin{aligned}
P(Y=1) = \\
&\frac{1}{m^n}\Bigg[\binom{r}{1}\Big[\sum_{l=4}^{7}\binom{n}{l}\sum_{k=0}^{r-1}\binom{n-l}{k}(r-1)_k(m-r)^{n-l-k}\Big] \\
&+\binom{r}{2}\Big[\binom{n}{2}\binom{n-2}{2}\sum_{k=0}^{r-2}\binom{n-4}{k}(r-2)_k(m-r)^{n-4-k} \\
&+2\binom{n}{3}\binom{n-3}{2}\sum_{k=0}^{r-2}\binom{n-5}{k}(r-2)_k(m-r)^{n-5-k} \\
&+\binom{n}{3}\binom{n-3}{3}\sum_{k=0}^{r-2}\binom{n-6}{k}(r-2)_k(m-r)^{n-6-k}\Big] \\
&+\binom{r}{2}\Big[2\binom{n}{4}\binom{n-4}{2}\sum_{k=0}^{r-2}\binom{n-6}{k}(r-2)_k(m-r)^{n-6-k} \\
&+2\binom{n}{4}\binom{n-4}{3}\sum_{k=0}^{r-2}\binom{n-7}{k}(r-2)_k(m-r)^{n-7-k} \\
&+2\binom{n}{5}\binom{n-5}{2}\sum_{k=0}^{r-2}\binom{n-7}{k}(r-2)_k(m-r)^{n-7-k} \\
&+2\binom{n}{5}\binom{n-5}{3}\sum_{k=0}^{r-2}\binom{n-8}{k}(r-2)_k(m-r)^{n-8-k}\Big] \\
&+\binom{r}{3}\Big[\binom{n}{2}\binom{n-2}{2}\binom{n-4}{2}\sum_{k=0}^{r-3}\binom{n-6}{k}(r-3)_k(m-r)^{n-6-k} \\
&+3\binom{n}{3}\binom{n-3}{2}\binom{n-5}{2}\sum_{k=0}^{r-3}\binom{n-7}{k}(r-3)_k(m-r)^{n-7-k} \\
&+3\binom{n}{3}\binom{n-3}{3}\binom{n-6}{2}\sum_{k=0}^{r-3}\binom{n-8}{k}(r-3)_k(m-r)^{n-8-k} \\
&+\binom{n}{3}\binom{n-3}{3}\binom{n-6}{3}\sum_{k=0}^{r-3}\binom{n-9}{k}(r-3)_k(m-r)^{n-9-k}\Big]\Bigg]
\end{aligned}
\tag{7}
$$

3.2.2 At Least

We also find the probability of getting at least one super-unique card in a certain time period:

$$
\begin{aligned}
P(Y\geq 1) = 1 - \Big[\frac{1}{m^n}\Big[&\sum_{k=0}^{r}\binom{n}{k}(r)_k(m-r)^{n-k} + \binom{r}{1}\binom{n}{2}\sum_{k=0}^{r-1}\binom{n-2}{k}(r-1)_k(m-r)^{n-2-k} \\
&+\binom{r}{1}\binom{n}{3}\sum_{k=0}^{r-1}\binom{n-3}{k}(r-1)_k(m-r)^{n-3-k}\Big]\Big]
\end{aligned}
\tag{8}
$$

Once again, we find the probability of getting no super-unique cards and subtract it from 1. In order to have no super-unique cards, we can have no more than one of any unique card (i.e. no more than three of any rare card). We first find the probability of having no unique cards (i.e. no repeated rare cards). We then add the probability of having one unique card with two of one type of rare card and with three of one type of rare card.

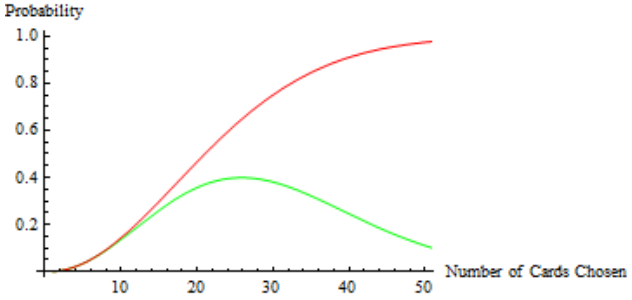
3.2.3 Continued Example

We continue our probability example from the previous section. Recall, there are 60 cards in the deck, 10 rare cards in the deck, and the player chooses one card a day over a 30 day period. The probability of getting exactly one super-unique is ≈ 0.2177 and the probability of getting at least one super-unique is ≈ 0.2246. So, on average 22.46% of the population will have at least one super-unique after playing for 30 days.

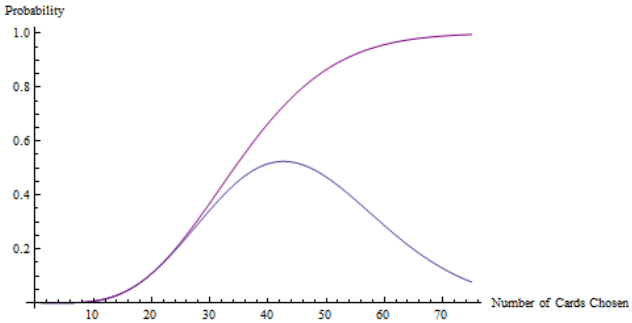
4 Comparing Fusion and Quad-Fusion

Using a deck size of 60 with 10 rare cards (as in the examples above), we can allow the number of cards chosen to vary and look at the resulting probability. We first compare the probability of getting exactly and at least one unique card. We find that the probability of getting exactly one unique card gradually increases until about 0.3 and then gradually decreases. However, the probability of getting at least one unique rapidly increases and then plateaus around 1.

We then compare the probability of getting exactly and at least one super-unique card. Here, we find results similar to the unique probabilities, where the probability of getting exactly one super-unique card peaks at about 0.55.



(a) Unique

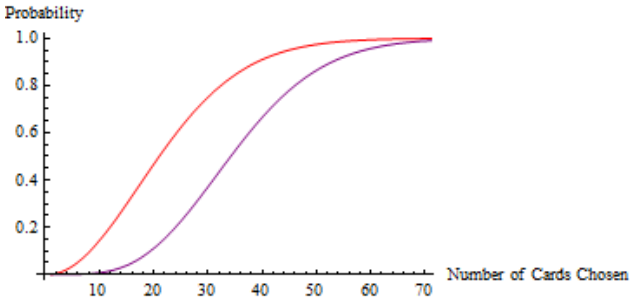


(b) Super-Unique

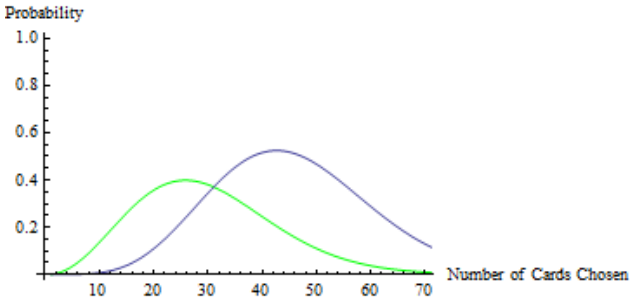
Figure 8: Probability of getting exactly (green) and at least (red) one unique card and exactly (blue) and at least (purple) one super-unique card.

Additionally, we compare the probability of getting at least one unique and at least one super-unique. Though the probabilities have similar distributions, the probability of getting a unique card is higher than the probability of getting a super-unique card.

We also compare the probability of getting exactly one unique and exactly one super-unique. The probabilities follow similar distributions; however, it is less likely early on to get a super-unique card than a unique card as it takes more cards to do so.



(a) At Least



(b) Exactly

Figure 9: Probability of getting at least one unique (red) and super-unique (purple) and exactly one unique (green) and super-unique (blue).

5 Evolutionary Trees

Figure 10: "Evolution in *Reign of Dragons*." (reignofdragons.wikia.com)

We now find the probability of completing an evolutionary tree over a certain time period. Recall that evolutionary trees involve different levels of fusion. For example, in the game Reign of Dragons, evolutionary trees can be created using four, six, or eight cards. We let Z be the number of evolutionary trees. We find the probability of completing exactly one evolutionary tree with y cards. Because we need two cards for fusion, we make the assumption that y is even:

$$P(Z=1) = \sum_{k=y+1}^{r+y} \left[\binom{n}{k} \frac{(m-r)^{n-k}}{m^n} \left[\binom{k}{y+1} (r)_{k-y} + \binom{k}{y} (r)_{k-y-1} \right] \right] \tag{9}$$

Here we alter our formula for completing exactly one fusion by incorporating the number of cards required to complete an evolutionary tree y.

5.1 Example

When there are 70 cards in the deck, 2 rare cards in the deck, the player chooses one card a day over a 365 day period, and the evolutionary tree requires 8 cards, the probability that the player will complete exactly one evolutionary tree in 365 days is :

$$P(Z=1)=\sum_{k=8+1}^{2+8}\left[\binom{365}{k}\frac{(70-2)^{365-k}}{70^{3}65}\left[\binom{k}{8+1}(2)_{k-8}+\binom{k}{8}(2)_{k-8-1}\right]\right]\approx .0161$$

6 Recipe

6.1 2-card recipes

A recipe fusion occurs when two non-identical cards are combined to get a new, more powerful card. For example, a knight card and a horse card could be combined to get a cavalryman card. We consider the probability of getting exactly one recipe in a certain time period. We denote the number of recipes in the deck p. Additionally the total number of cards in the deck is m and the number of cards picked is denoted n:

$$P(\text{exactly one two card recipe})=p\binom{n}{2}\sum_{k=0}^{p}\binom{n-2}{k}\frac{(2p)_k(m-2p)^{n-k-2}}{m^n} \tag{10}$$

To derive this probability formula, we first pick which of the recipes we will use from the p choices. Then we choose the two days that those two cards are picked. In the first case those are the only two potential recipe cards picked so the remaining $n-2$ days are filled with the non-recipe cards. The sum accounts for the possibilities of choosing additional recipe cards, but not choosing two of the same, as to create a second recipe. Here we again choose the recipe and day of the recipe followed by choosing the remaining k days that contain recipe cards. For the first of those days there are $2p$ choices, followed by $2p-1$ choices for the next day, and so on. This restricts the player from picking an ingredient in the recipe multiple times. We recognize that choosing a single ingredient more than once would not lead to the creation of a recipe, but it inhibits the possibility of having another kind of fusion.

As an example we show the probability of getting exactly one two card recipe with a deck size of 100, 6 possible recipes, and a time period of 30 days:

$$P(\text{exactly one two card recipe})=6\binom{30}{2}\sum_{k=0}^{6}\binom{30-2}{k}\frac{(12)_k(100-12)^{30-2-k}}{100^{30}}=0.175$$

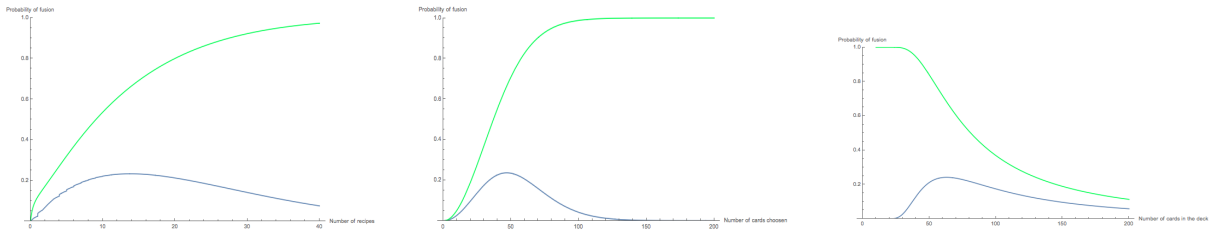
Therefore a player would have a 17.5% chance of getting exactly one recipe with those parameters.

6.2 Comparing the probability of selecting exactly one recipe and at least one recipe

In this section we compare the probability of getting exactly one recipe fusion to the probability of getting at least one recipe fusion. Knowing how often a player will get exactly one recipe is useful information for a developer to determine the number of recipes they would like to include in their deck, but comparing that with the probability of getting at least one recipe provides additional information about the behavior of the deck. The probability of creating at least one recipe is given by the following formula

$$P(\text{at least one two card recipe})=1-\frac{1}{m^n}\sum_{k=0}^{p}\binom{n}{k}(2p)_k(m-2p)^{n-k}. \tag{11}$$

We analyze the difference in the two distributions by holding two of the parameters constant and allowing the third to vary. In Figure 3(a) we vary the number of recipe cards in a deck containing 100 cards, chosen 30 times. We see that as the number of recipes increases, the probability of getting exactly one recipe increases until a point where the player is now likely to select multiple recipes. In this example that point appears to be around fifteen recipes. The probability of getting at least one continues to increase and approaches 1 as the number of recipe cards reach the size of the deck. This analysis illustrates that even if the probability presented from the formula for picking exactly one recipe is low, developers must ensure that it is not because there are a disproportionate amount of recipes leading to an increased chance of the player not only strengthening their deck through one recipe fusion, but multiple fusions. We note that we see a similar pattern when we vary the number of cards selected with a deck of 100 cards, containing 6 recipes. This is shown in Figure 3(b).



(a) Varying number of recipes (b) Varying number of cards selected (c) Varying deck size

Figure 11: Comparison of the probability of exactly one (blue) to at least one (green) while varying parameters

When we vary the size of the deck we see that the probability for both exactly one and at least one decreases with larger values of m. This is shown in Figure 3(c) for a deck with 6 recipes and 30 cards selected. Again there appears to be a maximum value where the probability of selecting exactly one recipes declines, but this is not because more recipes will be selected. Instead it is because the chance of selecting any recipe decreases. When the deck size is small in comparison to the number of recipes then the probability of getting at least one is high, while the probability of getting exactly one is lower.

6.3 3-card recipe

In this section we will let k be the number of additional recipe cards, p the number of recipes, and l the number of recipes with a pair of ingredients chosen. As with the two-card recipes we do not allow a player to select an ingredient in a recipe more than once.

While inspecting the various values of k, we determined that the cases and subcases depend on the number of pairs chosen in a given hand l. If $k < 2l$, then there are not enough recipe cards to have l pairs in the hand. Also, if there are p recipes, then when $k > p$ there must be some number of pairs to fill the k days that contain ingredients in a recipe. For example if there are 6 recipes then we can have $l = 0$ only when we select 6 or fewer additional cards. For this reason we introduce the Heaviside functions $U_p(k)$ and $U_l(k)$.

$$U_p(k) = \begin{cases} 0 & : k > p + l \\ 1 & : k \leq p + l \end{cases}$$

$$U_l(k) = \begin{cases} 0 & : k < 2l \\ 1 & : k \geq 2l \end{cases}$$

Additionally, recall that $(p)_k = p(p-1)\cdots(p-(k-1))$ denotes the falling factorial function. We use these facts to determine a formula for the probability that exactly one three card recipe is chosen:

$$P(\text{exactly one three card recipe}) = \frac{1}{m^n} p \binom{n}{3} \left[\sum_{k=0}^{2p} \binom{n-3}{k} (m-3p)^{n-3-k} (U_p(k) 3^k (p)_k + \right.$$
$$\left. 2U_p U_l(k) 3^{k-1} (p)_{k-1} + 2^2 U_p(k) U_l(k) 3^{k-2} (p)_{k-2} + \cdots + 2^l U_l(k) 3^{k-l} (p)_{k-1}) \right] \tag{12}$$

7 Sensitivity Analyses

In this section, we present sensitivity analyses of our probability formulas to give developers an idea of which parameters would have the most effect on the decks they create.

7.1 Fusion

We consider the probability of getting at least one unique card and conduct a sensitivity analysis upon the size of the deck m, the length of the time period n, and the number of rare cards in the deck r by varying each by intervals of -25%, -15%, -10%, -5%, -1%, 1%, 5%, 10%, 15%, and 25%. Our base for the sensitivity analysis is $m = 100$, $n = 30$, and $r = 35$.

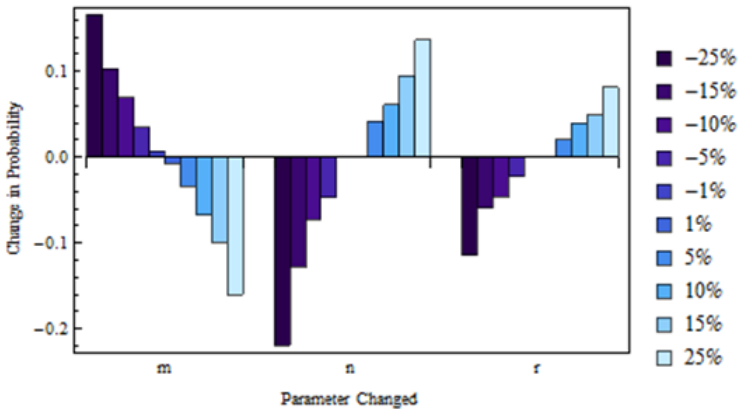


Figure 12: Sensitivity Analysis: Fusion

Surprisingly, we find that the number of rare cards in the deck is the least sensitive parameter. The most sensitive parameter is the number of cards chosen with a change of up to -0.22. A decrease in the number of cards chosen and the number of rare cards in the deck decreases the probability of getting a unique card, while a decrease in the number of cards in the deck increases the probability of getting a unique card. We expect this pattern, as choosing more cards and having more rare cards in the deck would increase the probability of getting rare cards, while having fewer cards overall would also increase the probability of getting a rare card.

7.2 Quad-Fusion

Here we consider the probability of getting exactly one super-unique card. Again, our base for the sensitivity analysis is $m = 100$, $n = 30$, and $r = 35$.

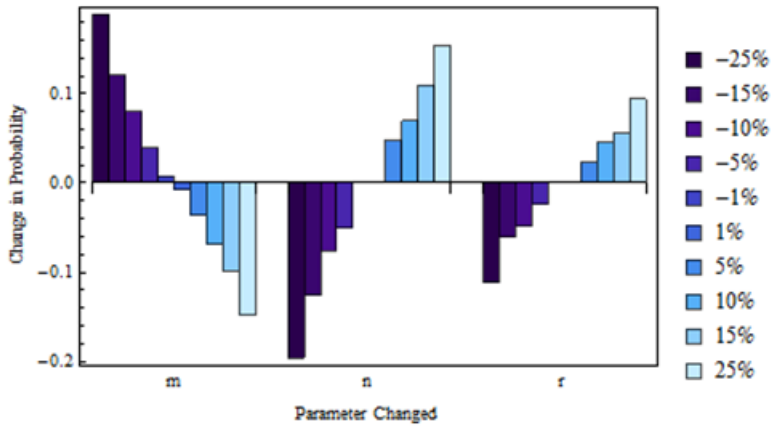


Figure 13: Sensitivity Analysis: Quad-Fusion

The results of the sensitivity analysis on Quad-Fusion are similar to those on at least one unique card. Once again, the most sensitive parameter is the number of cards chosen, with a change of up to 0.19 and the least sensitive parameter is the number of rare cards in the deck.

7.3 Two-Card Recipe

We also conduct a sensitivity analysis for two-card recipes with a base of $m = 100$, $n = 30$, and $p = 6$.

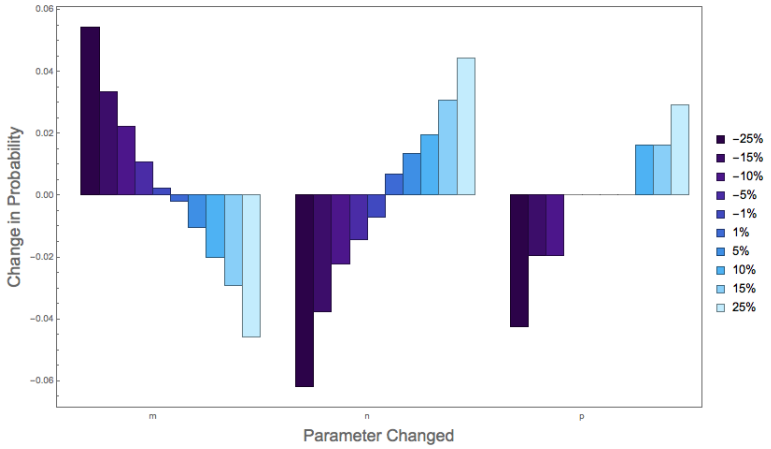


Figure 14: Sensitivity Analysis: Two-card recipes

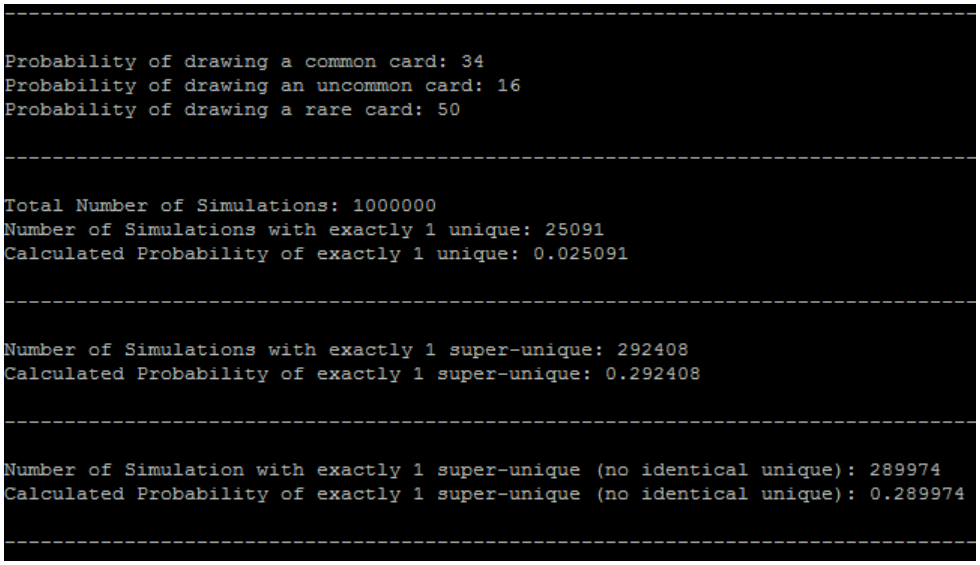
We find that the number of cards selected and the number of cards in the deck have close to the same sensitivity. The number of cards selected appears to be only slightly more sensitive with a maximum change in probability of -0.06 for the 25% decrease in n. In conjunction with the the unique and quad fusion formulas the least sensitive parameter is the number of recipes in the deck.

8 Simulating the Situation

To ensure our calculated probabilities are realistic, we have developed a C++ program using *Visual Studio 2013* and the Linux Portal *Putty* to simulate the scenario. This program, when given the pre-defined variables for deck size (m),

days cards are drawn (n), number of each type of card in the deck (c, u, and r), and the probabilities of drawing each type of card on a single day, can cycle through a million simulations in a short amount of time. It's important to note that we use the same assumptions outlined in Section 2.3 to run our simulations and test the simulated probabilities against our theoretical formulas. In each case tested, our simulations matched the results from the equations described in Sections 3. During execution, the program tracks how many of the simulations meet the criteria we set forth. The output of the program is divided between two files, **Simulations** and **Results**. The Simulations file contains a record of all the simulations, complete with what card was drawn on what day, how many of each type of card was drawn, and so forth. The Results file contains the most vital information for us. It outputs the number of simulations that obtain exactly one unique and those that obtain exactly one super-unique, along with the expected probabilities based upon this data.

Below shows an example of the output from the Results file. In this example, we ran 1,000,000 simulations with a 100 card deck that has 34 common cards, 16 uncommon cards, and 50 rare cards. The coding for this program along with explanation can be found in the Appendix.



The next step for our program is to use it as a shell to create a second program that will allow a game-developer to input the size of a deck, the time period, and the desired unique fusions during that time period and return the amount of rare cards in the deck necessary to make such an event highly probable. Since our equations for calculating the probability are so complex, we have not been successful in "reverse engineering" them to calculate the necessary number of rare cards mathematically. Therefore, the idea behind the program is to cycle through x number of simulations with z amount of rare cards in the deck, just as the first program did, but to do this for all possible values of z such that $1 \leq z \leq m$, where m is the deck size. Then, the program will return the top three z values that match the desired outcome best.

We would also like to modify our program so it will be useful for recipe fusion. To do so will require re-creating the entire deck of cards so that every card is distinct as well as developing recipes to use for the simulations. Two options for this are to have the user input the desired recipes or to have the program create random recipes to search for. If we are able to create such a program, game-developers will be able to determine the optimal deck composition to achieve the determined recipes.

All together, we hope to provide game-developers with a set of programs that can be useful for simulating outcomes when players are able to draw one random card a day with different parameters that support the mathematical equations. To do so will be of great use to many developers of Card Collecting Games.

9 Conclusion

We have developed formulas that incorporate fusion, quad-fusion, evolutionary trees, and recipes in card collecting games. Given the number of cards in the deck, the number of rare cards or recipes in the deck, and the number of cards a player chooses, we have found the probability that a player creates any of these fusions. We have also created a C++ simulation program that verifies our results and allows game developers to see the results of varying any of the parameters. Game developers will be able to use our results to enhance their card collecting games in a way that will both better the experience for players and increase profit for the game. Using our probability formulas and C++ program, developers can determine the number of each type of card to place in their deck in order to ensure that the general population of gamers will be able to obtain different levels of achievement in a manner that encourages both continued game play and money spent. Moreover, with the results in this paper, developers will be able to quickly determine the effects that changes in deck size, number of rare cards, and number of picks per play will have on the number of fusions in a given time period. Developers can use this information to make changes in their game before sending their modified games to beta testers, thus saving time and money.

10 Future Work

The use of mathematics to understand and analyze both the mechanics and strategy of video games, including card collecting games, is a field with numerous open problems. In terms of our project, there are a few follow up problems that arise immediately from the work presented in this paper. In addition to further analyzing the three card recipe formula, we would like to develop probability formulas for recipe fusion with multiple sizes of recipes in a single deck. For example, we would need a formula that encompasses recipes with both two and three ingredients. We recognize that if we condition that the cards in the recipes are independent, we would simply compute the product of two formulas, but we would like to find a formula that does not require this condition. Another useful tool would be to determine a formula that incorporates both unique and recipe fusion. We also believe it would be interesting mathematically to investigate the point in our formula where the probability of getting exactly one fusion begins to decline in terms of the number of items that have the potential to be fused as well as the number of cards a player selects. Lastly, we would like to extend our computer program to simulate the scenarios involving recipes. We would also like to develop a program that accepts a desired probability of creating one rare card over n days from the user and cycles through simulations of different deck compositions (i.e. one rare in a 100 card deck, 2 rares in a 100 card deck, 3 rares in a 100 card deck, etc.) and returns the parameters of the simulation that was closest to the desired probability.

11 Acknowledgements

This research was completed as part of participation in the Preparation for Industrial Careers in the Mathematical Sciences (PIC Math) program. Support for this Mathematical Association of America (MAA) and Society for Industrial and Applied Mathematics (SIAM) program is provided by the National Science Foundation (NSF grant DMS-1345499). Special thanks to Anthony Pecorella of Kongregate for support and guidance on this project.

References and Notes

1. W. of the Coast LLC, Magic: The gathering (2015).
   URL `http://magic.wizards.com/`
2. Pokemon/Nintendo, The official pokemon website (2015).
   URL `http://www.pokemon.com/us/`
3. Reign of dragons wiki (2014).
   URL `http://reignofdragons.wikia.com/wiki/Reign_of_Dragons_Wiki`

4. A. Tucker, Applied combinatorics, John Wiley & Sons, Inc., 2006.