A SYMBOLIC FINITE-STATE APPROACH FOR AUTOMATED PROVING OF THEOREMS IN COMBINATORIAL GAME THEORY

THOTSAPORN "AEK" THANATIPANONDA AND DORON ZEILBERGER

Abstract. We develop a finite-state automata approach, implemented in a Maple package `ToadsAndFrogs` available from our websites, for conjecturing, and then rigorously proving, values for large families of positions in Richard Guy's combinatorial game "Toads and Frogs". In particular, we prove a conjecture of Jeff Erickson.

1. Introduction

The game *Toads and Frogs*, invented by Richard Guy, is extensively discussed in "Winning Ways"[1], the famous classic by Elwyn Berlekamp, John Conway, and Richard Guy, that is the *bible* of combinatorial game theory.

This game got so much coverage because of the simplicity and elegance of its rules, the beauty of its analysis, and as an example of a combinatorial game whose positions do not always have values that are numbers.

The game is played on a $1 \times n$ strip with either Toad(T) , Frog(F) or □ on the squares. Left plays T and Right plays F. T may move to the immediate square on its right, if it happens to be empty, and F moves to the next empty square on the left, if it is empty. If T and F are next to each other, they have an option to jump over one another, in their designated directions, provided they land on an empty square. (See [1], page 14).

In symbols: the following moves are legal for Toad:

$\ldots \mathrm{T}\square \ldots \quad \rightarrow \quad \ldots \square\mathrm{T} \ldots,$
$\ldots \mathrm{TF}\square \ldots \quad \rightarrow \quad \ldots \square\mathrm{FT} \ldots \quad ,$

and the following moves are legal for Frog:

$\ldots \square\mathrm{F} \ldots \quad \rightarrow \quad \ldots \mathrm{F}\square \ldots,$
$\ldots \square\mathrm{TF} \ldots \quad \rightarrow \quad \ldots \mathrm{FT}\square \ldots \quad .$

Already in "Winning Ways"[1], there is some analysis of Toads and Frogs positions, but on *specific*, small boards, such as TTT□FF. In 1996, Jeff Erickson[2] analyzed more general positions. At the end he made five conjectures about the

1991 *Mathematics Subject Classification.* 91A46.
Key words and phrases. Combinatorial Game Theory.

values of some families of positions. All of them are "starting" positions (i.e. positions where all T's are rightmost and all F's are leftmost).

To be able to understand the present article, readers need some knowledge of combinatorial game theory, that can be found in [1]. In particular, readers should be familiar with the notion of *value* of a game. Recall that values are not always numbers (not even surreal ones).

Let's recall the *bypass reversible move rule*, the *dominated options rule* (see [1] page 62-64) and Erickson's *Terminal Toads Theorem* (see[2]).

Bypassing right's reversible move rule

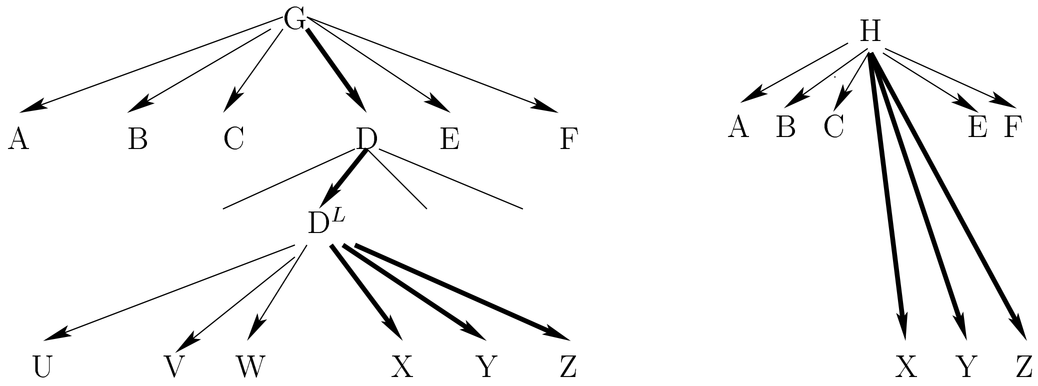

FIGURE 1. The Bypassing reversible move rule.

$G = H$ if $D^L \geq G$.

The Dominated options Rule

Let $G = \{A, B, C, ... \mid D, E, F, ..\}$.
If $A \geq B$ and $D \geq E$ then $G = \{A, C, ... \mid E, F, ..\}$.

The Terminal Toads Theorem: Let X be any position. Then
$\mathrm{XT}\square^n = \mathrm{X}\square^n + n$.

The only notation we use is $*$ $(= \{0 \mid 0\})$. We will not use any shorthand notation like $\uparrow$, $\Uparrow$, etc.

Next, we will explain the method through examples, and describe how to implement the method when applied to certain classes of positions. Finally, we discuss a new conjecture and possible future work.

Everything is fully implemented in a Maple package, `ToadsAndFrogs`, written by the first author, available from our websites.

2. A Symbolic Finite-State Method

We define two classes of positions:

Class **A**: All the positions that have a *fixed* number of occurrences of □ and F, but a *variable* (symbolic) number of T's in-between the □'s and F's.

class **B**: All the positions that have a *fixed* number of occurrences of T's and F's, but a *variable* (symbolic) number of □'s in-between the T's and F's.

Aij := the class in which we have exactly i occurrences of □ and exactly j occurrences of F.

Bij := the class in which we have exactly i occurrences of T and exactly j occurrences of F.

For any *specific* position, we can always compute its value, by using the recursive definition of the value. But this is mere *number-crunching*. After collecting enough data, and examining it, if we are lucky, we (or rather our computers) can detect a uniform *pattern*, and **conjecture** an **explicit** formula for the values of the studied family, in terms of the symbolic parameters. Once conjectured, these conjectured explicit expressions can be proved by induction on the symbolic parameters. The beauty and novelty of our approach is that everything is done **automatically**. First the *conjecturing* parts, but more dramatically, the *proving* part. We teach the computer how to conjecture, by looking for general patterns, and then how to use induction in order to prove its own conjectures.

This *activity* of **computer-generated** mathematics is in sharp contrast to the traditional approach of [2], that merely uses the computer as a calculator, to generate numerical data, and everything else, the conjecturing, and the proving (when feasible) is done by humans.

We believe that the present methodology is of potential use in many other branches of mathematics, and "Toads and Frogs" is but an instructive arena for presenting a general approach for computer-generated research.

When we analyze each class of positions, we are naturally lead, by the recursive definition of the *value* (of a game), to other classes of positions. Luckily, at least in all the cases encountered so far, there are always a **finite** number of different classes, that we can name "symbolic states". If the (symbolic) value of each "state" in the class is conjectured to have a (symbolic) explicit expression, then we can prove the truth of *all* these conjectures **all at once** by applying induction on the recurrence relations. Note that in order for this to work we need to conjecture explicit expressions for **all** the states, so we usually get much more than we bargained for.

We will demonstrate the method with the two simplest nontrivial classes: A11 and B11.

First example: Type A11: one $\square$ and one F

Let $f(a,b)$ be the value of $\mathrm{T}^a\square\mathrm{T}^b\mathrm{F}$.
Let $g(a)$ be the value of $\mathrm{T}^a\mathrm{F}\square$.

Here, of course, T^a means T repeated a times, so the 'game' $f(a,b)$, for example, stands for a doubly-infinite set of starting positions.

Recurrences:
Note that if any parameter of the function is negative then it returns NULL.

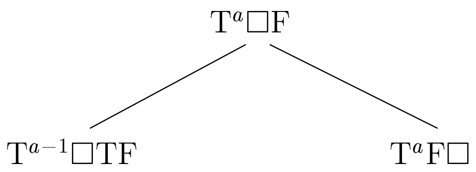

FIGURE 2. Recurrence for $f(a,0), a \geq 0$.

$$f(a,0) = \{f(a-1,1) \mid g(a)\}, a \geq 0.$$

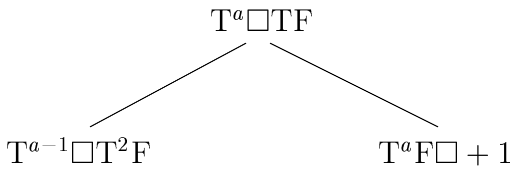

FIGURE 3. Recurrence for $f(a,1), a \geq 0$.

$$f(a,1) = \{f(a-1,2) \mid g(a)+1\}, a \geq 0.$$

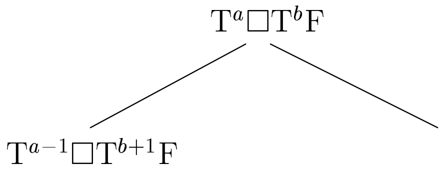

FIGURE 4. Recurrence for $f(a,b), a \geq 0, b \geq 2$.

$$f(a,b) = \{f(a-1,b+1) \mid \ \}, a \geq 0, b \geq 2.$$

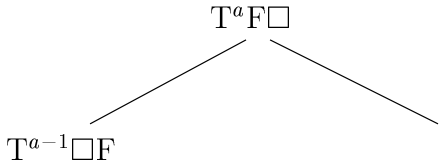

FIGURE 5. Recurrence for $g(a), a \geq 0$.

$$g(a) = \{f(a-1,0) \mid \ \}, a \geq 0.$$

The above recurrences can be easily used to crank out *numerical data* for small (and not so small) values of a and b. Then the computer *automatically* makes the following *symbolic* conjectures.

Conjectures:

$$\begin{array}{lcll}
f(0,0) & = & -1. & \\
f(a,0) & = & \{\{a-2 \mid 1\} \mid 0\} & , \quad a \geq 1. \\
f(a,1) & = & \{a-1 \mid 1\} & , \quad a \geq 0. \\
f(a,b) & = & a & , \quad a \geq 0, \quad b \geq 2. \\
g(a) & = & 0 & , \quad a \geq 0.
\end{array}$$

Once conjectured, the proof is routine, and also can (and was!) done by computer. One checks the obvious initial conditions and verifies that the above expressions satisfy the above *defining* relations. Indeed, the computer easily verifies that

$$\begin{array}{lcll}
f(a,0) & = & \{f(a-1,1) \mid g(a)\} = \{\{a-2 \mid 1\} \mid 0\} & , \quad a \geq 1. \\
f(0,1) & = & \{\ \mid g(0)+1\} = \{\ \mid 1\} = 0 = \{-1 \mid 1\}. & \\
f(a,1) & = & \{f(a-1,2) \mid g(a)+1\} = \{a-1 \mid 1\} & , \quad a \geq 1. \\
f(0,b) & = & \{\ \mid \ \} = 0 & , \quad b \geq 2. \\
f(a,b) & = & \{f(a-1,b+1) \mid \ \} = \{a-1 \mid \ \} = a & , \quad a \geq 1, b \geq 2. \\
g(0) & = & \{\ \mid \ \} = 0. & \\
g(1) & = & \{f(0,0) \mid \ \} = \{-1 \mid \ \} = 0. & \\
g(a) & = & \{f(a-1,0) \mid \ \} = \{\{\{a-3 \mid 1\} \mid 0\} \mid \ \} & \\
 & = & \{\ \mid \ \} \text{ (!! by bypass reversible move rule)} = 0 & , \quad a \geq 2.
\end{array}$$

Note that the above values for $f(a,0)$ $(a \geq 1)$ agree with the case $b = 1$ of Theorem 5.2 of [2].

Second Example: Type B11: one T and one F.

Let $f(a,b,c) := \Box^a \mathrm{T} \Box^b \mathrm{F} \Box^c$.

Now we have a *three*- parameter family!

Initial Conditions and Recurrences:

$$\begin{array}{rcll}
f(0,0,0) &=& \{\ \mid\ \}. & \\
f(a,0,0) &=& \{\ \mid (-a+1)+1\} = \{\ \mid -a+2\} & , \quad a \geq 1. \\
f(0,0,c) &=& \{(c-1)-1 \mid\ \} = \{c-2 \mid\ \} & , \quad c \geq 1. \\
f(a,0,c) &=& \{c-a-2 \mid c-a+2\} & , \quad a \geq 1, c \geq 1.
\end{array}$$

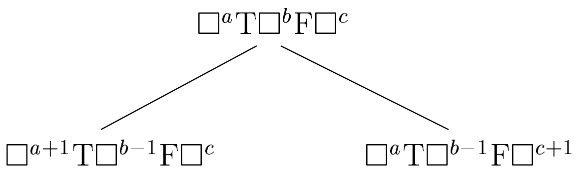

FIGURE 6. Recurrence of $f(a,b,c), a \geq 0, c \geq 0, b \geq 1$.

$$f(a,b,c) = \{f(a+1,b-1,c) \mid f(a,b-1,c+1)\}, \quad a \geq 0, c \geq 0, b \geq 1.$$

By using these recurrences *numerically*, the computer cranks out enough data, that enables it to make the following

Conjecture:

$$\begin{array}{rcll}
f(a,b,c) &=& \{c-a-2 \mid c-a+2\} & , \quad a \geq 0, c \geq 0, b \text{ is even.} \\
f(a,b,c) &=& \{\{c-a-3 \mid c-a+1\} \mid \{c-a-1 \mid c-a+3\}\} & , \quad a \geq 0, c \geq 0, b \text{ is odd .}
\end{array}$$

Proof: by induction: on b.

Base case: $b = 0$
We have

$$\begin{array}{rcll}
f(0,0,0) &=& 0 = \{-2 \mid 2\}. & \\
f(a,0,0) &=& \{\ \mid -a+2\} = \{-a-2 \mid -a+2\}, & a \geq 1. \\
f(0,0,c) &=& \{c-2 \mid\ \} = \{c-2 \mid c+2\}, & c \geq 1. \\
f(a,0,c) &=& \{c-a-2 \mid c-a+2\}, & a \geq 1, c \geq 1.
\end{array}$$

Induction step on b

Case 1) b is even and $b \neq 0$:

$$\begin{array}{rcl}
f(a,b,c) &=& \{f(a+1,b-1,c) \mid f(a,b-1,c+1)\}, \quad a \geq 0, c \geq 0. \\
&=& \{\{c-a-4 \mid c-a\} \mid \{c-a-2 \mid c-a+2\}\} \\
&& \mid \{\{c-a-2 \mid c-a+2\} \mid \{c-a \mid c-a+4\}\}\}. \\
&=& \{c-a-2 \mid c-a+2\}.
\end{array}$$

Case 2) b is odd

$$\begin{array}{rcll}
f(a,b,c) &=& \{f(a+1,b-1,c) \mid f(a,b-1,c+1)\} & , a \geq 0, c \geq 0. \\
&=& \{\{c-a-3 \mid c-a+1\} \mid \{c-a-1 \mid c-a+3\}\}. &
\end{array}$$

The second example is related to the results of Erickson [2] as follows. The case $b = 0$ is Lemma 4.1 of [2], while the case $a = 0, c = 0$ coincides with the case $a = 1$

of Theorem 5.2. Note that we need the extra elbow-room of a three-parameter family to enable the inductive argument.

3. How far can the symbolic finite state method go?

As we mentioned in the previous section, the finite state method works perfectly well when the value of every position in the class has a discernible pattern. This seems to be the case for class A. We wrote a computer program in Maple to first *calculate*, then *conjecture*, and finally *prove*, the values of general positions in class A automatically. The program now works for positions with any fixed number of □'s and with one Frog. For the class where we have more than one Frog, it is harder to find conjectures, for humans, and *even* for computers. We conjectured some classes with two Frogs(A12, A22, A32) by hand and put it in the computer program to prove the conjectures.

The list of the results for the classes A11, A21, A31, A41, A51, A12, A22, A32 can be found in both authors' websites.

As a very special case of our results for the class A32, we get a proof of Erickson's[2] conjecture 2, that claims that the value of T^a□□□FF is $\{a-2 \mid a-2\}$, ($a \geq 2$).

In [3], the first-named author of the present paper discusses the value of *any* position with one □ and *any* number of Toads and Frogs (Therefore we are done with class $A1n$, $n \geq 1$). This general class with one □ is the only general class we are able to figure out the patterns for.

We now turn our attention to class B. We solved class B11 in the previous section. For B21: TTF, we already have a difficulty. The formulas in this class are long and hard to find in a canonical form. We will discuss this in the appendix.

4. A Conjecture and Future Work

Conjecture:

1) We always have "nice compact" formulas for every position in class A.

Future Work:

1) Implement the symbolic finite state method for the class B21.

2) We have seen systems of recurrence relations arising naturally in each class. We solved the recurrences by "guessing" (automatically, of course) the answers (using predefined *ansatzes*) and then proving them by induction. It would be interesting to develop general algorithms for systematically solving the recurrences, without the need for "guessing".

APPENDICES

Appendix A. On the difficulty of class B21: TTF

B21: TTF
$f(a,b,c,d) := \square^a \mathrm{T} \square^b \mathrm{T} \square^c \mathrm{F} \square^d$.
$g(a,b,c) := \square^a \mathrm{T} \square^b \mathrm{F} \square^c$.

We already knew the solution of g since it is exactly B11.
We can now focus on f.

Recurrences:

$$
\begin{array}{rcl}
f(a,0,0,0) &=& \{\ |\ \} = 0. \\
f(a,0,0,d) &=& \{g(a,1,d-1)+d-1 \mid\ \} \\
&=& \{\{\{d-a-4 \mid d-a\} \mid \{d-a-2 \mid d-a+2\}\} \mid\ \} \\
&& ,\ \ a \geq 0,\ \ d \geq 1. \\
f(a,b,0,0) &=& \{f(a+1,b-1,0,0) \mid g(a,b-1,1)+1\} \\
&& ,\ \ a \geq 0, b \geq 1. \\
f(a,b,0,d) &=& \{f(a+1,b-1,0,d),\ \ g(a,b+1,d-1)+d-1 \mid g(a,b-1,d+1)+d+1\} \\
&& ,\ \ a \geq 0, b \geq 1, d \geq 1. \\
f(a,b,c,d) &=& \{f(a+1,b-1,c,d),\ \ f(a,b+1,c-1,d) \mid f(a,b,c-1,d+1)\} \\
&& ,\ \ a \geq 0, b \geq 0, c \geq 1, d \geq 0.
\end{array}
$$

Note: $f(a,0,0,d)$ has been discussed before as lemma 4.3 by Erickson.

A nice formula for $f(a,b,0,0)$.

For $b=1$:

$$
\begin{array}{rcl}
f(a,1,0,0) &=& \{f(a+1,0,0,0) \mid g(a,0,1)+1\} \\
&=& \{0 \mid \{-1-a \mid 3-a\}+1\} \\
&=& \begin{cases} \frac{1}{2} & ,\ a = 0,1 \\ \{0 \mid 3-a\} & ,\ a \geq 2 \end{cases}
\end{array}
$$

For $b \geq 2$, and b even:

$$
\begin{array}{rcl}
f(a,b,0,0) &=& \begin{cases} 1 & ,\ a = 0 \\ -a+2 & ,\ a\ \geq 1. \end{cases} \\
&=& \{\ \mid a\} - a + 2.
\end{array}
$$

For $b \geq 2$ and b odd.

$$
f(a,b,0,0) = \begin{cases} \{1 \mid 1\} & ,a = 0 \\ \frac{1}{2} & ,a = 1 \\ -a+2 & ,a \geq 2. \end{cases}
$$

However for $f(a,b,0,d), a \geq 0, b \geq 1, d \geq 1$, the formulas get longer and longer and we started to lose track of them, and consequently failed to find formulas in this case. It should be possible to write Maple code specifically to find a pattern for the values of positions in class B. The authors expect the formulas in other classes of type B (for example B22: TTFF) to be even more complicated than B21, since it has to build up from B21.

It appears that the positions in class B have periodicity and they need more care to formulate the right conjectures.

Appendix B. About the program

Our Symbolic Finite-State Method was implemented by one of us (TT) in Maple. He first wrote a program to recursively calculate the values of games. Then he improved the program by making use of the symbolic computation capability of Maple, to formulate conjectures, and prove the values of game-positions. The whole proof process was completely automated. Below is the short description of the program. See the authors' web sites for complete details of the program.

ToFr

Input: the specific position of the game.

Output: the value of the game in canonical form.

SVG

Input: the value of the game, could be symbolic.

Output: the value of the game in canonical form.

Note: This program can also be used for other combinatorial games.

MainConj

Input: number of $\Box$ and number of F.

Output: The list of conjectures.

Prove

Input: number of $\Box$ and number of F.

Output: the values of all of the positions in this class.

The program currently only works for one Frog with any fixed number of $\Box$. With more than one Frog, it gets harder to find conjectures. But one could find conjectures by hand and feed them to the subfunctions in Prove. The program can help verify such humanly-made conjectures.

Obviously, there is still a lot of work to be done, but let's remember that

“ **Every great artwork always starts from a rough draft**”.

References

[1] Elwyn Berlekamp, John Conway, and Richard Guy, *Winning Ways for your Mathematical Plays*, Academic Press, New York, 1982.

[2] Jeff Erickson, *New Toads and Frogs Results*, in: "Games of No Chance", 299-310, Richard J. Nowakowski, ed., Math. Sci. Res. Inst. Publ. **29**, 1996.

[3] Thotsaporn "Aek" Thanatipanonda, *Further Hopping with Toads and Frogs*, preprint (available from the author's website) .

Department of Mathematics

Hill Center-Busch Campus

Rutgers University

110 Frelinghuysen Rd

Piscataway, NJ 08854-8019

USA
E-mail address: `thot@math.rutgers.edu,zeilberg@math.rutgers.edu`