

Complete Manual

Hofstadter v1.1

Nigel Chan

HKU – The University of Hong Kong

Robert Lech

The University of Toronto

Michael Margel

The University of Toronto

Introduction

This document provides a complete reference to searching and generation of Meta-Fibonacci recursions using Hofstadter. It explains the notation that the program used and the functionalities that it supports.

***What is Hofstadter?***

Hofstadter is a flexible, portable, accessible and efficient program for generating and analyzing Meta-Fibonacci Explorer. Hofstadter is:

|  |  |
| --- | --- |
| **Flexible** | Hofstadter is embedded with a robust parser for mathematical expression. |
| **Portable** | Hofstadter is implemented in C++, dynamically linked with Qt development library. No extra package or software has to be pre-installed in order to have it run smoothly on Windows, Mac or Linux. |
| **Accessible** | Hofstadter comes with this complete reference containing rigorous definition and the methodology, allowing user to understand the results thoroughly. |
| **Efficient** | Hofstadter implements efficient algorithms to generate recursions and its summary statistics. It offers excellent performance with respect to runtime and memory usage. Users can enjoy state-of-the-art algorithms within reach. |

***Getting Help***

Any question can be directed to:

* Robert Lech at [robert.lech123@gmail.com](mailto:robert.lech123@gmail.com) or
* Michael Margel at [michael.margel@utoronto.ca](mailto:michael.margel@utoronto.ca).

User Guide

This manual provides step-by-step guidance on how to generate set of Meta-Fibonacci Sequences using Hofstadter. It contains various details encountered during usage.

**Contents**

Selecting the Recursion.……………………………………………………………………….. 4

Parameters..…………………………………………………………………………………………. 8

Initial Conditions ..………………………………………………………………………………. 10

Filters.………………………………………………………………………………………………… 12

Summary Options.……………………………………………………………………………… 14

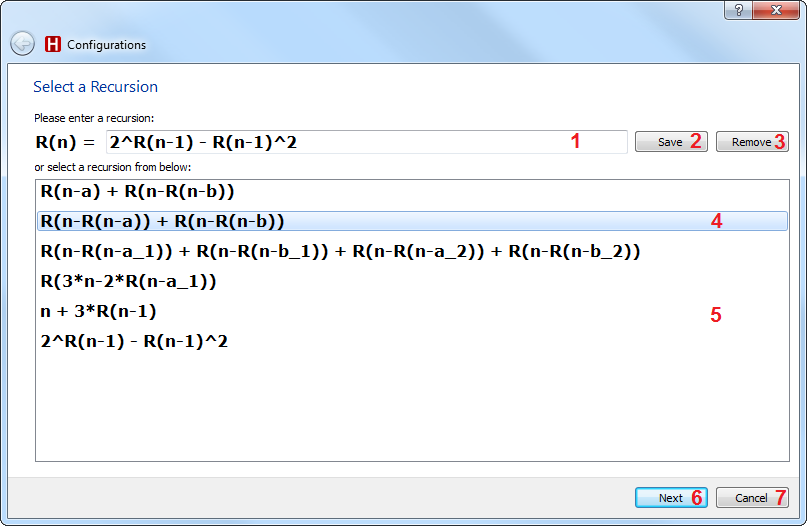
Result Page ………………………………………………………………………………………… 16

Change Log ………………………………………………………………………………………… 19

***Selecting the Recursion***

This is the first page you will see when running Hofstadter.

From here, you can select the recursion that will be calculated.

******

1. The recursion entry field. This can be updated by typing a new recursion here, or by selected a recursion from the list below (5).
2. Adds the recursion in the entry field (1) to the recursion list (5) so it can be reused at a later time.
3. Removes the selected recursion (4) from the recursion list (5).
4. The currently selected recursion is highlighted.
5. The list of all saved recursions. Selected a recursion from the list will update the recursion entry field (1).
6. Proceeds to the Filter Page (Page 7) using the recursion in the entry field (1).
7. Closes the program.

***Entering a Recursion Manually***

If a user does not wish to use an existing recursion, they can enter a new one my typing it into the recursion entry field.

Hofstadter uses the following notation when interpreting the recursion.

|  |  |
| --- | --- |
| R(…) | All recursion must be denoted by the sequence R(…), where the expression inside the parentheses evaluates to an integer value. |
| (), ^, +, -, \*, / | All operators are calculated using the standard operator precedence.  Parentheses can be nested as many times as required.  Hofstadter will treat “/” as the integer division.  ie: “a/b” will be evaluated as floor(a/b).  Note that multiplication must be done using the asterisk (\*) symbol. Therefore, Hofstadter will not treat 3R(n-1) as 3\*R(n-1) |
| 1, 2, … | All numbers are interpreted literally.  Only positive integers are supported. |
| a, b, … | All other text will be treated as a parameter name.  Parameter names are delimited by R(…), operators, and parentheses.  Parameter names can include underscores, numbers, and special characters. They cannot include hyphens.  The parameter “n” is reserved by the program. Using it will result in unusual behavior. |

**Note: All whitespace in the recursion will be ignored.**

Examples

>> R(n) = R(n-s-R(n-j))+R(n-s-j-R(n-2\*j))

>> R(n) = n – R(R(n-1)

>> R(n) = n – R(R(R(n-1)))

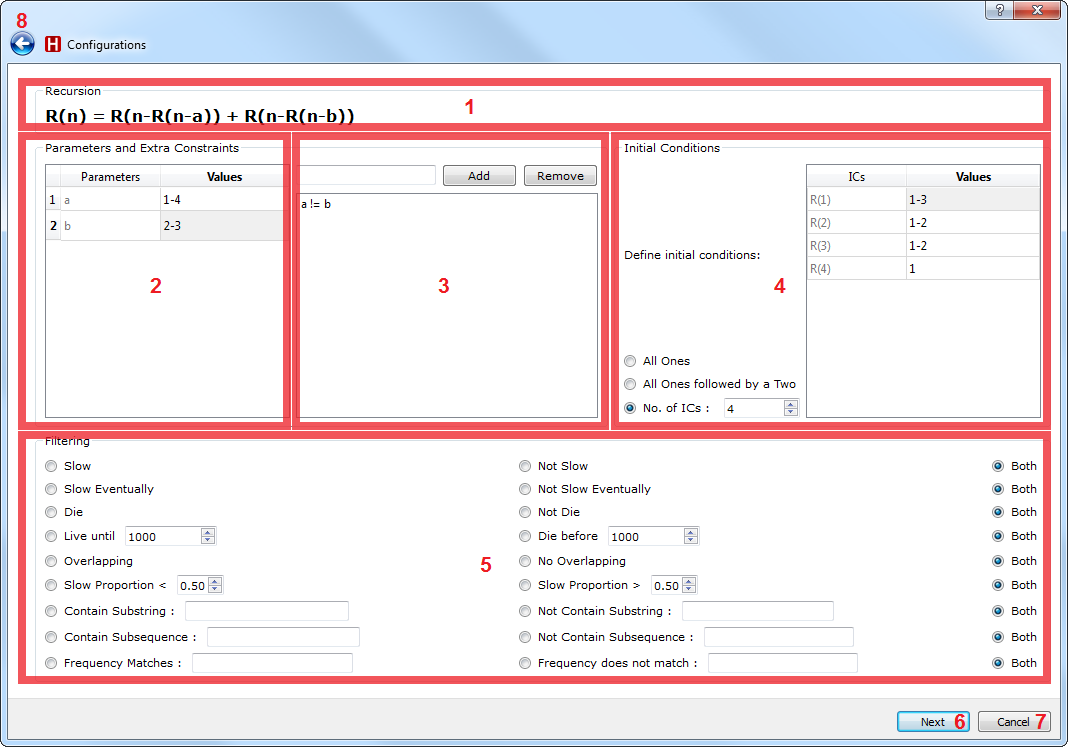
>> R(n) = R(n-a\_1-R(n-b\_1)) + R(n-a\_2-R(n-b\_2))

OR

>> R(n) = R(n-a1-R(n-b1)) + R(n-a2-R(n-b2))

***Configuring the Filters, Parameters, and Initial Conditions***

After pressing the *Next* button on the recursion input page, you will be taken to this page, which will let you configure the filters, parameters, and initial conditions for the recursion.



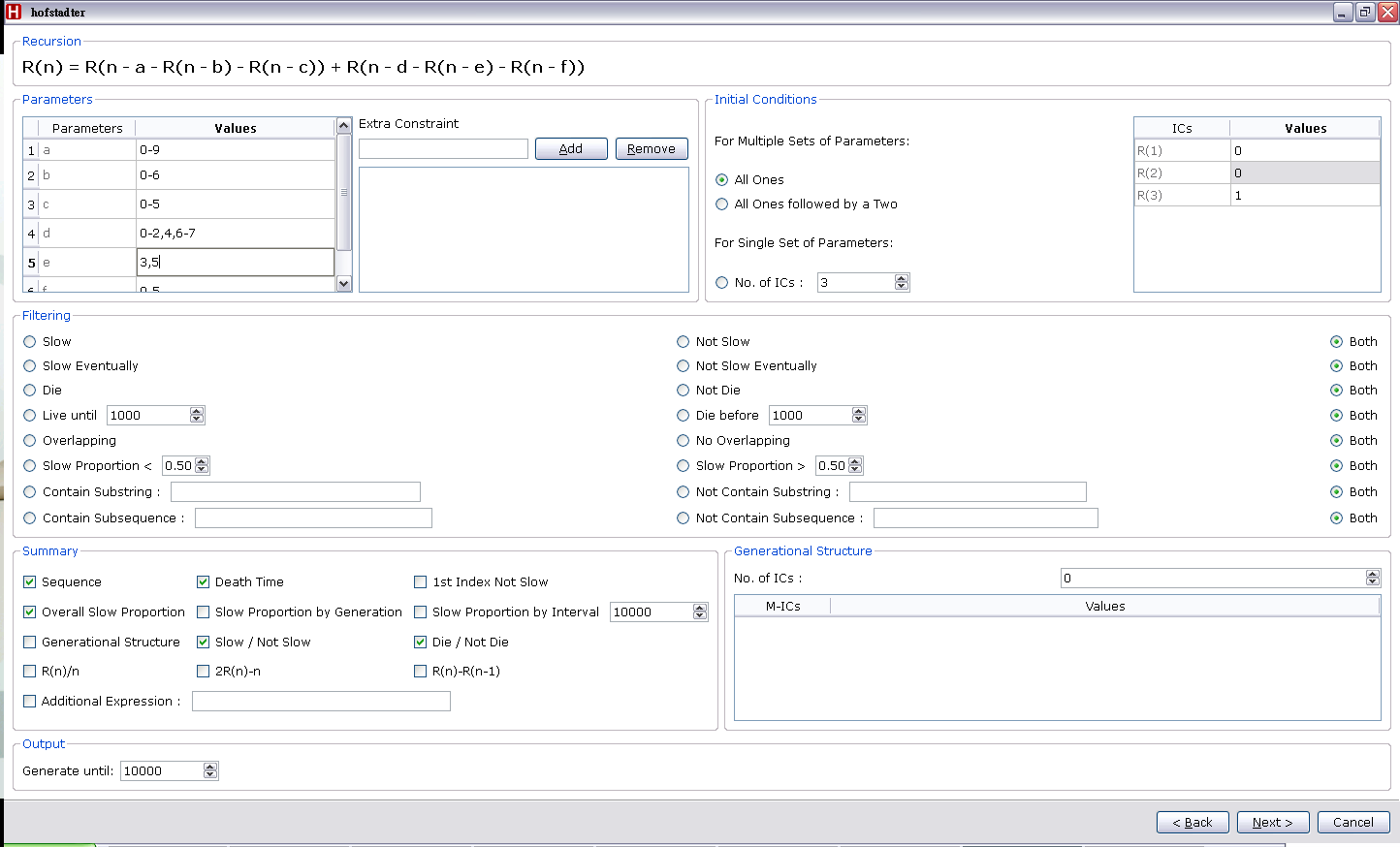
1. Shows the current recursion.
2. The Parameter Input Panel, detailed below.
3. The Parameter Constraint Panel, detailed below.
4. The Initial Condition Panel, detailed below.
5. The Filter Panel, detailed below.
6. Proceeds to the next page, using the options selected here.
7. Closes the program.
8. Returns to the recursion input page.

***Parameter Input Panel***

You can use the Parameter Input Panel to enter values for the parameters.

By setting values for the parameters, you can determine which values Hofstadter will use to calculate the recursion. These values will be shown in the recursion box on all subsequent pages.

The recursion will be calculated once for each combination of values (as well as Initial Conditions, Page 10)



By double-clicking on an entry in the *Values* column, you can edit the values for that parameter.

You can use hyphens to indicate a range of values, and use commas to separate ranges.

Examples

“4-7” 🡪 {4, 5, 6, 7}

“1, 4, 5” 🡪 {1, 4, 5}

“1-3, 6-8, 10” 🡪 {1, 2, 3, 6, 7, 8, 10}

**Note: Using a large number of combinations for parameter values and initial conditions can be very computationally complex, and may take a significant amount of time to calculate.**

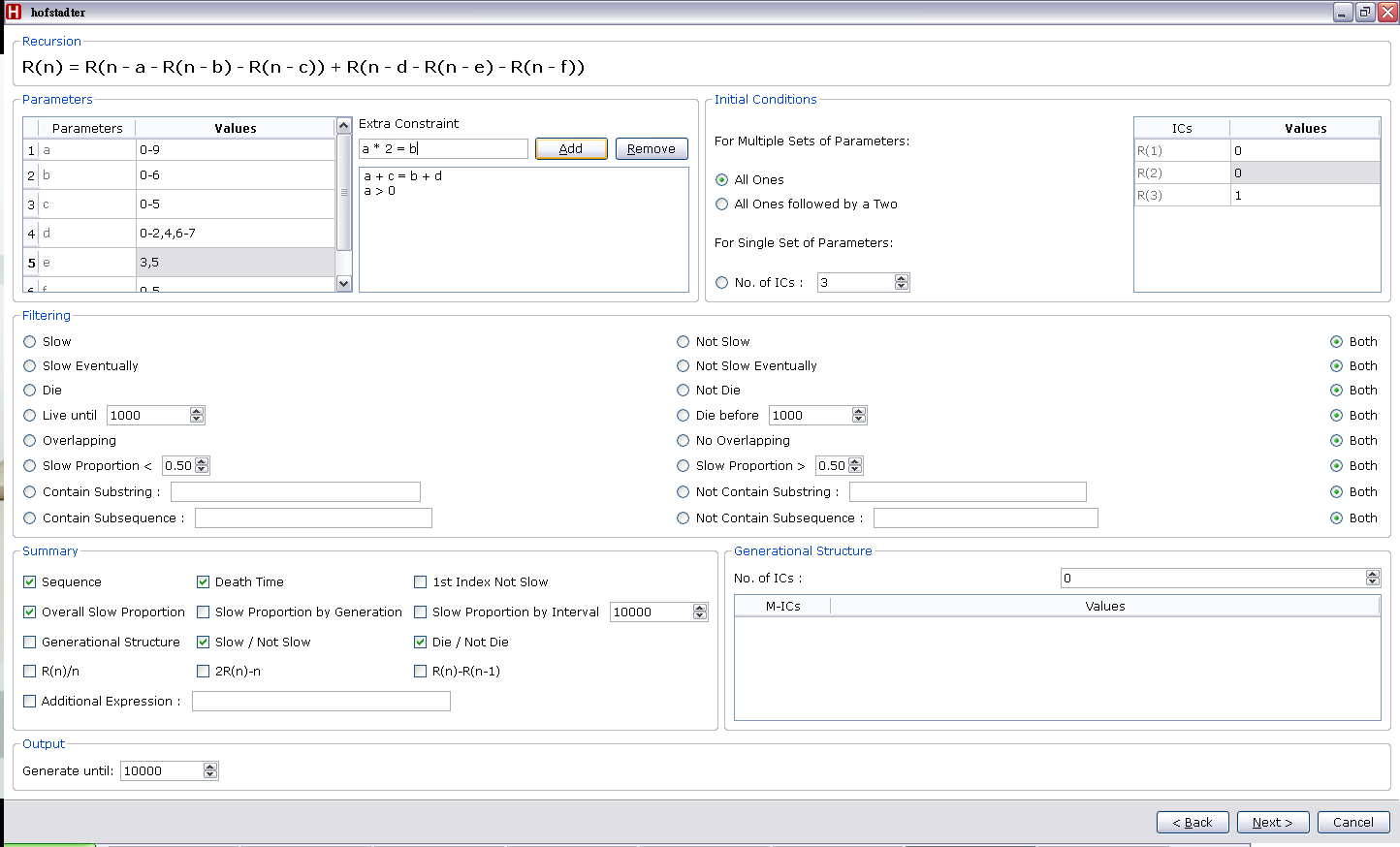
**For instance, using 8 combinations for parameter values and 4 combinations for initial conditions will cause Hofstadter to calculate the recursion 32 times.**

***Parameter Constraint Panel***

You can use the Parameter Constraint Panel to set constraints on the parameter values.

By adding constraints, you can restrict the combination of parameter values for which the recursion is calculated.

For instance, adding the constraint “a < b” will only calculate the recursions for parameter combinations where a < b.



You can add a new constraint by entering one in the text box, and pressing the *Add* button.

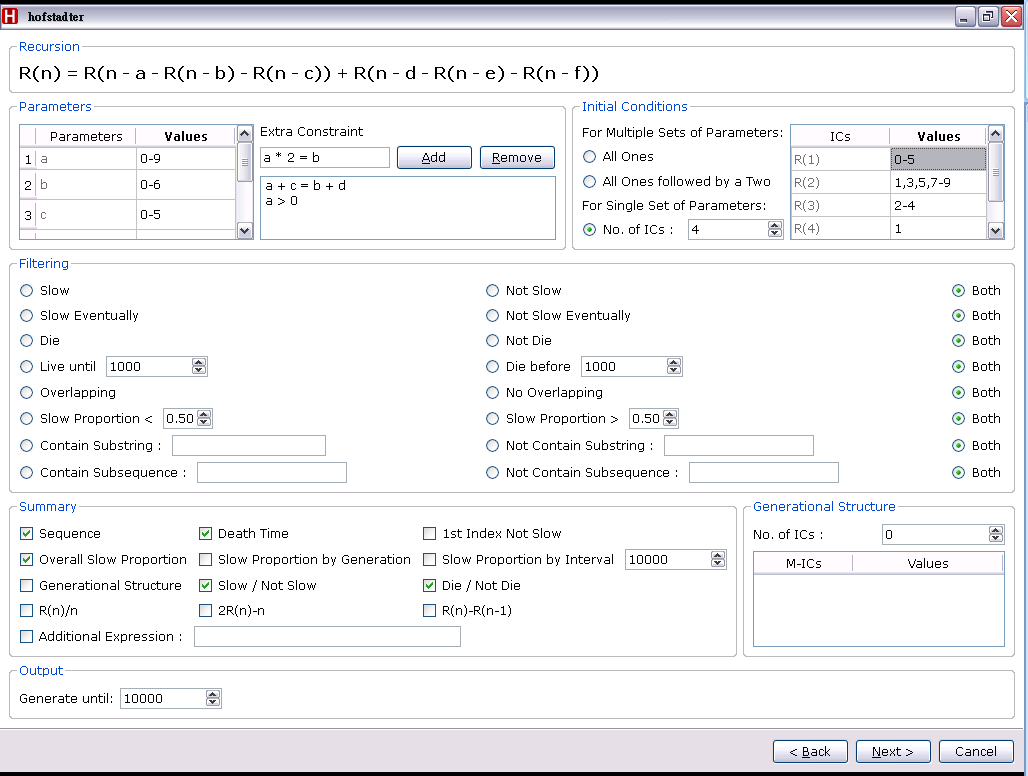
You can remove an existing constraint by selected it in the list, and pressing the *Remove* button.

Hofstadter supports the following comparisons:

Equal **“=”**, Greater Than **“>”**, Greater Than or Equal To **“>=”**, Less Than **“<”**, Less Than or Equal to **“<=”**.

***Initial Conditions***

The Initial Conditions Panel can be used to determine the initial conditions used to calculate each recursion.



**All Ones:**

Hofstadter will use the value 1 as values for each R(n) until some R(n) can be calculated. If the first 100 terms cannot be calculated, then the series will not be calculated.

Example:

“R(n) = R(n-4) + 1” will use R(1) = R(2) = R(3) = R(4) = 1, since none of those terms can be calculated recursively, and then it will use “R(5) = R(1) + 1”, since this can be calculated.

**All Ones followed by a Two:**

Hofstadter will use the value 1 as values for each R(n) until some R(n) can be calculated. The last R(n) that cannot be calculated recursively will use R(n) = 2. If the first 100 terms cannot be calculated, then the series will not be calculated.

Example:

“R(n) = R(n-4) + 1” will use R(1) = R(2) = R(3) = 1 and R(n) = 2, since none of those terms can be calculated recursively, and then it will use “R(5) = R(1) + 1”, since this can be calculated.

**Custom Initial Conditions:**

Users can define their own initial conditions, for any number of values.

The number of initial conditions can be changed by either clicking on the arrows beside the number entry box, or by entering a value in it.

Similar to the parameter values (Page 8), users can use hyphens to indicate a range of values, and commas to separate ranges. The recursion will be calculated for all combinations of initial values.

Example:

Using “R(1) = 2-3”, “R(2) = 1-2”, and “R(3) = 1,3” will calculate the recursion for the following combinations: {2, 1, 1}, {2, 1, 3}, {2, 2, 1}, {2, 2, 3}, {3, 1, 1}, {3, 1, 3}, {3, 2, 1}, {3, 2, 3}.

**Note: Using a large number of combinations for initial conditions and parameter values can be very computationally complex, and may take a significant amount of time to calculate.**

**For instance, using 8 combinations for parameter values and 4 combinations for initial conditions will cause Hofstadter to calculate the recursion 32 times.**

Please note that users must click the radio button to select that option, and changing the number of initial conditions or their values will not cause the program to use the user-specific initial conditions.

**Note: There is currently a bug where recursions are not always shown on the final page when the user specifies both parameter values and initial condition values.**

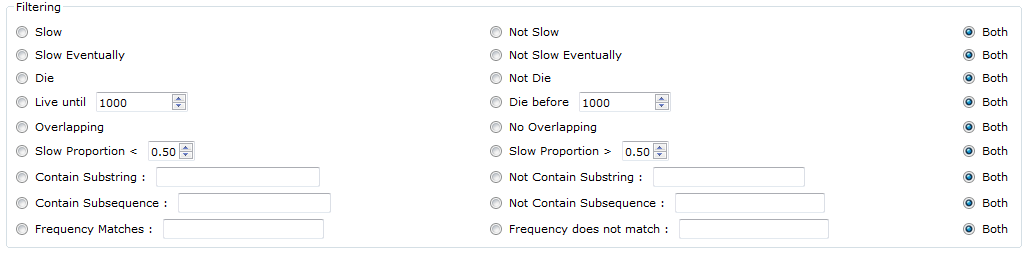
**This seems to only occur for sequences where the sequence dies immediately, or if the largest parameter value is greater than the number of initial conditions.**

**Please keep this in mind when using filters to restrict the output.**

***Filtering***

Filters can be used to restrict the output shown on the results page.

A recursion will only be shown if it does not violate any of the filters.



|  |  |
| --- | --- |
| Slow | True if the difference between R(n) and R(n-1) never exceeds 1. |
| Not Slow | True if it is not slow. |
| Slow Eventually | True if the last 50% of the sequence is slow. |
| Not Slow Eventually | True if the last 50% of the sequence is not slow. |
| Die | True if the sequence dies. |
| Not Die | True if the sequence does not die during the calculation. This checks for all values n <= N, which is defined on the Summary Page (Page 14). |
| Live until x | True if the sequence is defined for all n <= x. |
| Die before x | True if the sequence is undefined for some n < x. |
| Overlapping | True if there exists a value of starting point of a generation less than the ending point of its previous generation. |
| Slow Proportion <= x | If m is the number of n such that R(n)-R(n-1) <= 1, then this is true if m/N <= x. |
| Slow Proportion >= x | If m is the number of n such that R(n)-R(n-1) <= 1, then this is true if m/N >= x. |
| Contain Substring S | True if the string S is contained in the recursion. |
| Does not contain substring S | True if the string S is not contained in the recursion. |
| Contain subsequence S | True if the sequence S is contained in the recursion. |
| Does not contain subsequence S | True if the sequence S is not contained in the recursion. |

**Both:**

Selecting the “Both” option for a filter will disable that filter.

**Substrings & Subsequences:**

Consider the sequence [1,1,2,3,5,8,13,21,34,55]

[1,2,3,5] is a substring and a subsequence.

[1,3,5,55] is a subsequence, but not a substring.

[1,4,5] is neither a substring nor a subsequence.

A substring is always a subsequence.

**Note: The “Overlapping” and “Frequency” filters are untested, and their behavior may be unpredictable.**

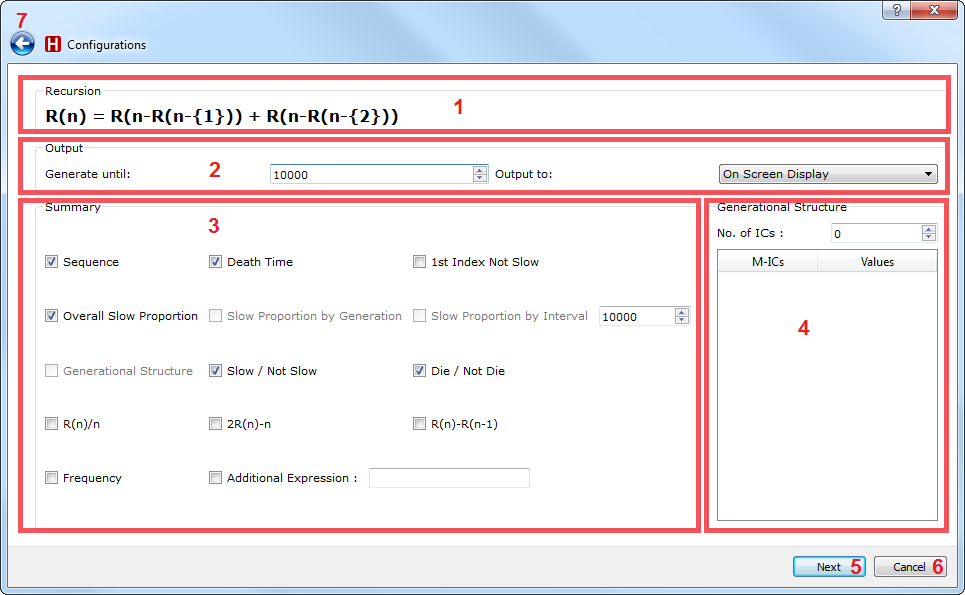
***Generational Structure***

Hofstadter allows user to define **a single set** of initial conditions for the maternal sequence of the recursion.

**Note: This may not work correctly with the current version.**

***Summary Options***

From the Summary Options page, users can select the information that is shown on the Results Page and the number of terms for which the recursion is calculated.



1. The recursion text. The parameter names are replaced by their values.
2. The output control panel. The user can indicate the number of terms to calculate and the output method. **Note: In this version, the results will be shown on the screen, regardless of the selected output.**
3. Options with a checked box will be shown on the results page. Options with an unchecked box will not be shown.
4. This allows the user to define the terms in the generational structure. **Note: This is untested with the new code, and may have unpredictable behavior.**
5. Calculates the recursions.
6. Closes the program.
7. Returns to the

**Summary Options**

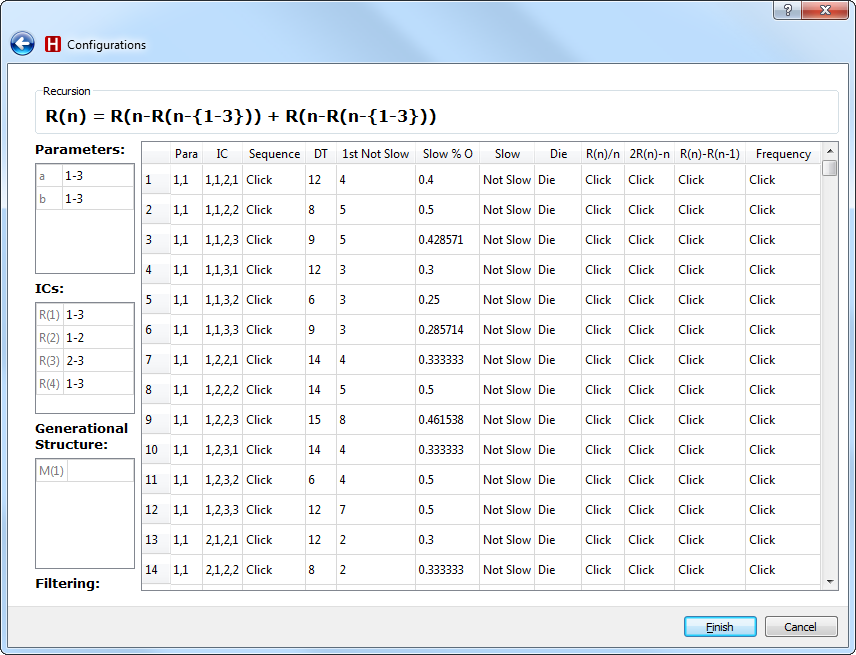
The summary options are as follows

|  |  |
| --- | --- |
| Sequence**\*** | Shows the sequence generated by the recursion. |
| Death Time | Shows the value for n where the sequence dies, or -1 if it does not die. |
| 1st Index Not Slow | Shows the smallest n for which R(n)-R(n-1) > 1. |
| Overall Slow Proportion | Shows the total proportion of slow terms. |
| Slow Proportion by Generation | ***-Disabled-*** |
| Slow Proportion by Interval | ***-Disabled-*** |
| Generational Structure | ***-Disabled-*** |
| Slow / Not Slow | Shows whether the sequence is slow or not slow. |
| Die / Not Die | Shows whether the sequence dies or not. |
| R(n) / n**\*** | Shows the sequence generated where the nth term is R(n) / n. |
| 2R(n) - n**\*** | Shows the sequence generated where the nth term is 2\*R(n) - n. |
| R(n) - R(n-1) **\*** | Shows the sequence generated where the nth term is R(n) - R(n-1). |
| Frequency | Shows the frequency of each value in the sequence |
| Additional Expression**\*** | Shows the sequence generated by some user-specified expression. **This option may be unstable.** |

Options indicated by a **\*** will have their output shown in a popup window.

***Results***

In addition to the summary options selected, the parameter values and initial conditions are also shown.

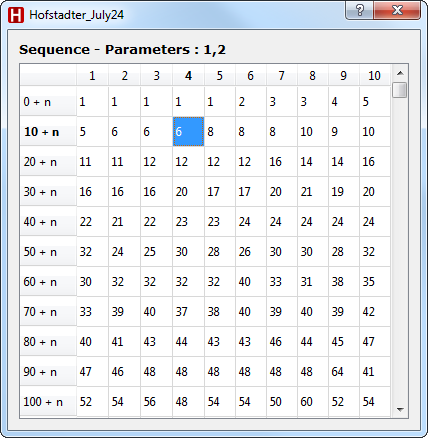


The left hand side shows the data used to calculate the recursions, and the large grid shows the results of those calculations.

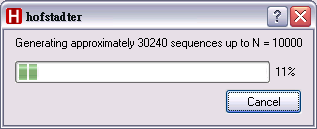
**Popup**

The cells in the result table that show “Click” will have their output shown in a popup such as the one below.

Currently, you can have multiple popup windows open at a time, for easy comparison.

***Progress Bar***

Hofstadter will show the progress during the generation of sequences. User can terminate by pressing the “Cancel” button. Note that the number of sequences represents the number of permutations for the given sets of parameters and initial conditions, as defined by the user.



When using a large number of combinations of initial conditions and parameters, it may seem as though the program has frozen. This behaviour is quite normal, and happens because calculating the recursion so many times can be very computationally intense.

***Change Log***

***1.0***

* The SUM and R^k(n) operators are no longer supported. These will be added again in a future build.
* Parameters can now have names longer than 1 character, and can contain any characters except operators and parentheses.
* The code used for calculating recursions has been replaced entirely.
* Most of the backend code has been simplified.
* Users can now specify both initial conditions and parameter values.

***Manual Change Log***

***1.0***

* The documentation for SUM and R^k(n) have been removed.
* Updated notation section.
* Updated notation examples.
* Added information for configuration page.
* Added information for filter page.
* Added information for parameter values.
* Added information for parameter constraints.
* Added information for initial conditions.
* Added information for summary page.
* Added information for result page.
* Added screenshots for all sections.