Framework for Tax Preparation

Charles Duan

January 25, 2020

Contents

This is a software system for automating the computation and filling of tax forms. Much like commercial software for tax preparation, the object of the system is to allow a user to enter data from received tax forms and other sources, and then answer interview questions, to produce completed tax forms ready for filing.

Unlike commercial tax software, however, this system is not intended to comprehensively cover all tax forms and options. Instead, a second object of this system is to provide a framework to make it reasonably easy for programming-adept users to add tax forms as necessary to their personal situations and needs.

1 Overview of Operation

Use of the system proceeds in three phases.

The first phase is the computation of form values. The user enters relevant data from received tax forms and other sources into files, and then writes a short script that imports that data and computes the values for tax forms to be prepared. This produces a plain text file containing the line item data to be filled into the tax forms.

The second phase is the identification of form fields. For each tax form to be filled in, the user provides the PDF file for the blank form and also

the file of line item data. A program displays a graphical interface where the user can click on the blank spaces where each line item should be entered. The result of this is a second plain text data file that lists the coordinates of each of the form fields.

The third phase is the actual filling of the forms. The user writes another short script that receives the data files from the first and second phases, and places the line items from the first-phase file into the PDF form. If any changes need to be made subsequently, the user may simply rerun the first-phase computation script and then rerun the third-phase filling script, thereby updating all of the tax forms automatically.

2 Recommended Procedure for Annual Filing

This section provides recommendations on preparing an annual tax filing. This includes updating the software in view of changes to the tax laws and forms, as well as preparing the returns themselves.

First, prepare the script for running the computations, and enter the data for received forms (W-2s, 1099s, etc.). It is preferable to do this first rather than updating the tax form computations, because it allows for partially running the program in advance. This has two benefits as you update the computations: It reveals bugs earlier, and it lets you assess with real data whether it is worth implementing a new computation.

At this step, be sure to review the informational forms in the Appendix and ensure that you have entered all relevant information required there.

Next, update the computations. Start with the main form being computed (e.g., Form 1040), and work line by line through the script. I found it preferable to address each form dependency as it came immediately, resulting in a depth-first update. That is, when the 1040 calls for a result from Schedule B, I would pause work on the 1040 and go update Schedule B. The advantage of this is that you can continue running partial tests of the computations. The main disadvantage is that it becomes hard to keep track of which form you are working on; I generally addressed this by leaving comments in the files as I worked on them.

Once all the computations are done and run without errors, move on to phase 2 and mark the form fields. I found it useful to mark both the forms and the worksheets, even though the worksheets do not need to be filed, because it made it easier to review everything for accuracy.

Then prepare the script for filling in all the forms (phase 3), and fill them in to produce a draft return. This will help to identify any computation errors.

Once you have reviewed the draft return and corrected any issues, you can optimize the return (e.g., try different filing statuses, move dependents around, choose standard versus itemized deductions, and so on). Once you are satisfied, you can print, sign, and file your returns.

3 Object Classes and Data Structures

The tax computation system is written in Ruby and is organized around two main classes: the TaxForm and the FormManager.

3.1 The TaxForm

Generally, each tax form is represented as an instance of a Ruby class for that form, a subclass of TaxForm. A TaxForm has four essential methods:

name The name of this form. The word "Form" should not precede names.

year The tax year for which this form has been updated. This is used to check whether the form has been updated yet. (An out-of-date form will still be executed; it will just produce a warning.)

compute Compute the form. This method should fill in all the lines of the form.

needed? Whether the tax form is ultimately needed for inclusion in the return. This method should be called after compute, and it should inspect the results of the lines of the form to determine if the form needs to be filed.

A form is essentially a hash table that maps line numbers to data values. Line numbers may be any strings so long as they contain no spaces. Values may be atomic values (numbers, strings, dates), or may be arrays of those atomic values. The order of lines is remembered (and can be altered manually).

The TaxForm has several methods with the name "line" for getting and setting lines:

- line[#] returns the value of that line. If the value is an array or is unset, an error is raised.
- line[#, :opt] returns the value of the line, or if the line is unset it returns BlankZero.
- line[#, :all] returns an array for this line's values. It always returns an array, even if the line only has an atomic value.
- line[#, :sum] returns the sum of this line's values.
- line[#, :present] returns true or false depending on whether the line is set.
- line[#] = x will set the line to an atomic value; it will raise an error if given an array. If the line is already set, a warning will be raised.
- line[#, :all] = [x, y, z] will set the line to an array value.
- line[#, :add] = x will force the line to be an array and then will append the given value to that array.
- line[#, :overwrite] = x will set the line to an atomic value, and will not warn if the line was already set.

As a shortcut, the "method_missing" method for TaxForm will infer line numbers from methods named "line_#".

Every TaxForm is associated with a FormManager. Many of the methods of TaxForm are delegated to its FormManager as a matter of convenience.

3.2 How Tables Are Stored in Forms

Consider a form that requires input like so:

Line 1a Business name	**1b Income**	**1c Expenses**
Acme Enterprises	5000	400
Foobar Inc.	10000	
Big Corp.	15000	600

This would be stored in a TaxForm as three lines named 1a, 1b, and 1c as follows:

- Line 1a: [“Acme Enterprises”, “Foobar Inc.”, “Big Corp.”]
- Line 1b: [5000, 10000, 15000]
- Line 1c: [400, –, 600]

It may appear transposed from what one ordinarily thinks of as “lines,” but it corresponds better to the line numbering that tax forms tend to use. It also makes for somewhat easier processing since the usual situation is that a form calls for summation over a column, which can be easily obtained (e.g., “line[‘1b’, :sum]”).

A method “add_table_row” takes a hash mapping line numbers to values for a single row, and updates each line in a manner that reflects a “row” being added to a table of those lines.

3.3 Special Line Names

Generally line names in a form may be any text that has no spaces. But several line names will receive special treatment.

Lines ending with an exclamation mark. Such lines are meant for storing metadata or informative data for a form, when there is no space for entering the data. A common use is to create a line for transferring data to another form. Consider, for example, a worksheet that instructs to copy line 7 to a schedule if line 7 is greater than zero, but to copy line 15 otherwise. Instead of having to put logic into the schedule to choose between copying line 7 and line 15, the worksheet can designate a line “fill!” and the schedule can take the value from that line.

Lines ending with the text “_explanation”. This is for explanations that should appear on a continuation sheet. The value should be an array where the first item is the title of the explanation and the subsequent items are the text of the explanation, in troff format.

3.4 The TaxForm File Format

As described above, data for tax computation is presented in TaxForm objects. These objects are serialized to plain text files in the following format.

A form begins with a line starting with the word "Form", a space, and the name of the form. Lines of the form follow; they must be indented with whitespace, and the line number should be separated from the value with whitespace. For example:

```
Form W-2
     first_name          John
     last_name           Doe
     a                   123-45-6789
     b                   98-7654321
     c                   Acme Widgets Co.
     1                   50000
     2                   15000
```

Further forms may be included in the same file, delineated with the start word "Form" at the beginning of a line.

The line value may be any of the following:

blank zero A single dash, representing a zero value.

number A number, possibly with a decimal point and possibly led with a minus sign.

date Formatted as mm/dd/yyyy.

array A list of values, each of which is one of the above types. Arrays may be specified in two forms: A comma-separated list surrounded by square brackets, or another way described below.

boxed data Formatted as <*num*|*delim*|*data*>, where *num* is a number of boxes (usually the length of *data*), *delim* is a separator for the boxes (usually a null string for splitting per character, but a social security number would take a hyphen to fit it into a three-box unit), and *data* is a number or text string.

text string Anything else.

An array may be constructed by placing each array element on a separate line in the file, replacing the line number with a double quote mark. For example:

```
A   15
"   18
"   20
```

is identical to "`A [15, 18, 20]`".

When forms are read in from a file, they are instantiated as objects of class NamedForm, even if a class specific to the form exists. This helps to distinguish forms read from input from those that were computed. Generally it should be unnecessary to use any of the methods specific to particular forms on forms read from files, since those forms have already been fully computed.

No Form. If there are no forms of a certain type, the directive "No Form [#]" may be included in the import file. This avoids a warning that is produced when the FormManager tries to find a form that is not present. (The purpose of the warning is to avoid accidental omissions of forms.)

Table. If there are many TaxForm objects of the same type (e.g., Charity Gift), you can save some typing by entering them in a tabular format:

```
Table Charity Gift
    amount   cash?   name
    500      yes     Red Cross
    30       no      Salvation Army
    1000     yes     NPR
```

The last of these columns may contain values with spaces in them; the earlier ones may not.

3.5 The FormManager

The FormManager maintains a complete tax return and manages the forms in that return. Its methods mostly deal with adding, computing, and querying forms.

One interesting feature is that a FormManager has a method "forms(name)" that returns an array of forms with the given name. The returned object is an Array delegate that has an additional method "lines" for querying all the lines of all the returned forms.

Submanagers. A FormManager for an entity may contain "submanagers," namely references to other FormManager objects for different entities or tax years. This is a convenient way to draw information from a related filing.

Submanagers are added using the add_submanager method, which requires a name for identifying the submanager. Common names are :last_year and :spouse. The add_submanager_from_file method creates a new FormManager, imports data from the file, and adds a new submanager.

The Interviewer. A FormManager maintains an Interviewer that allows for TaxForms to query the user for information during execution. The Interviewer can also store the responses to a file so that the questions need not be asked again.

I'm generally trying to move away from Interviewer questions, instead putting all information into informational TaxForm structures described in the Appendix.

4 Phase One: Computation

This section describes the operation of the first phase, computation of tax form line items.

The usual process is as follows:

1. A FormManager is created and set up with the relevant parameters, submanagers, interviewer, and so on.

2. Input forms are read using the FormManager's "import" method.

3. The FormManager's "compute_form(class)" method is called to compute a new form. The FormManager creates an instance of the given TaxForm class, adds the instance to the manager, executes that instance's "compute" method, and then removes the form if the "needed?" method returns false. The "compute" method will likely make further calls to "compute_form" if further forms are needed.

4. The computed forms are saved to a file using the FormManager's "export" method.

5 Phase Two: Form Field Marking

The program that operates this phase is "mark_fields.rb". It must be supplied with the tax form data produced in phase 1, using the "-i" argument. With no further arguments, the program lists the forms for which the fields have not been marked yet.

To mark fields in a form, enter the form name as the first non-option argument to the command. If the form has not been marked yet, you must provide the name of the PDF file for the form as a second command line argument.

The program will automatically save the location of the PDF file and the coordinates of the marked fields to a file, which by default is named "pos-data.txt". This file name may be changed using the "-p" argument.

5.1 Boxed Data

In some cases, the PDF form will have individual boxes for each character or portion of text, rather than a single field for all the text. The "boxed data" feature enables filling in these form fields correctly.

To use this feature, the relevant TaxForm should call the method "box_line(line, count, split)" where line is the line number to be placed in boxes, count is the number of boxes available, and split is how to separate the line's data into segments (the default is an empty string, which means each character is separated individually). All this does is mark the relevant line in the form as a boxed-data line; computation proceeds unchanged.

When the form field marking program is invoked, it will detect boxed-data lines and treat them like arrays. Thus, each box will need to be marked individually. However, if there are more letters than boxes available, the program will truncate data values to fit.

6 Phase Three: Form Filling

To fill in the forms, create a script that creates a FormManager object with the tax forms from phase 1 imported. Then create a MultiFormManager to

handle the filling of forms. For each form to be filled in, use the fill_form method of the MultiFormManager.

The MultiFormManager can automatically produce continuation sheets when there is insufficient space. This requires providing two options:

continuation_bio Biographical text for the continuation sheets, just for identification purposes.

continuation_display Either :show, which will print the continuation page to the screen; :raw, which will print the raw troff code; or :append, which will compile the troff code and append the continuation sheet to the filled tax form.

7 Wish List

These are features that I'd like to implement some day.

Overall conditional tests. Right now I don't implement a variety of forms for situations like foreign income or special rules for fishermen; I just leave comments indicating that those are not implemented. Better would be to have some sort of up-front survey to confirm conditions that I don't expect (basically interview questions), and then assert those conditions in places where the relevant functionality is not implemented.

Traceable computations. Rather than storing result data, form lines could store objects that maintain a tree of computations. For example, if line 7 is the sum of lines 5 and 6, line 7 would be presented as an object referencing those two lines and carrying an "add" instruction. This would help with tracing errors and optimizing computations. It will also help with detecting whether form or line information was never used, since that is likely an error.

Asterisks for line notes. Currently if there is an explanatory note to a line, it just goes in some place determined during the form marking phase; there's no reference to the note near the line itself.

Sequence numbers. These could most simply be implemented as instance methods on each form class and used to order the output, though that would mean that the sequence numbers are lost after the computation phase. They could also be stored as a line in each form. Or they could be maintained in some external database, though that seems unnecessarily difficult to maintain.

Generalized manager. Rather than having to write a script that computes the 1040, there could be a higher-level form that performs the functions of the script in a more generalized way, among other things computing the 1040. This has the benefit that it could compute ancillary information as a cover sheet, such as a manifest of forms to file, the mailing address, and some summary information about the results.

Better Interviewer questioning. Right now, the questions themselves are the keys for uniquely identifying the questions. This is not great for length reasons. I'd prefer to use short names for the questions, and then have some sort of translation table for presenting the complete question, perhaps with help texts.

Line explanations. I would like to be able to display explanations for each line, to assist in reviewing computation results without having to fill in the forms. This would require some sort of database mapping line numbers to descriptions, either dispersed throughout the form classes or in some unified file.

Appendix

A Additions to Forms

A.1 1095-B

This is the health coverage form. No information other than the items listed below is needed.

hdhp? Whether this plan was a high deductible plan that qualifies for HSA contributions.

coverage "individual" or "family".

months The months of coverage for this plan. (Currently it is assumed that all persons on the form are covered for the same months.) These should be all-lowercase three-letter abbreviations of months.

A.2 1098

property The name of the Real Estate form with which this 1098 corresponds.

balance The average mortgage balance on this loan. See Pub. 936. You can compute this by averaging the balances on January 1 and December 31, or you can multiply the interest paid by the lowest interest rate of the year.

A.3 1099-DIV

qualified_exception If this form shows qualified dividends, this flag must be set to indicate whether an exception applies. See 1040 instructions.

A.4 1099-R

2b.not_determined? The relevant checkbox under line 2b.

2b.total_distribution? The relevant checkbox under line 2b.

ira-sep-simple? Whether the checkbox on the form with this name is checked.

destination Where this distribution went. It can be "Roth conversion" (currently the only one supported).

B Informational Forms

B.1 Alimony

Alimony received. See Pub. 504.

amount The amount of alimony received.

B.2 Asset

A form for a capitalized asset.

date The date the asset was put into service.

amount The dollar value of the asset when purchased.

179? Whether the asset is a section 179 deductible asset.

listed? Whether the asset is a listed asset, see Pub. 946 chapter 5.

dc_category The depreciation category for the asset in DC, see the FP-31 instructions, under Depreciation Guidelines.

dc_type Choose from "reference", "fixed", or "other" to categorize the asset for DC Form FP-31, lines 1–3.

description A description of the asset.

B.3 Biographical

Biographical information for an individual tax filer. Spouses should have a separate Biographical form.

whose The individual to whom this form pertains. May be "mine" or "spouse".

first_name The first name and middle initial.

last_name The last name.

phone Telephone number.

ssn Social Security number, with dashes.

home_address The first line of the home address.

apt_no The apartment number for the address.

city_zip The city, state, and ZIP code.

birthday The person's birthday.

blind? Whether the person is blind.

occupation Your occupation

B.4 Business Expense

A deductible business expense. See IRS Pub. 535, chapter 11.

date The date of the business expense.

amount The dollar amount of the expense.

category The category of expense. "Meals" and "Utilities" will automatically be halved, the former per the IRS 50% rule and the latter on the assumption that 50% of the expense was for non-business purposes. Other common values are "Supplies", "Travel", and "Membership".

description A description of the expense.

B.5 Charity Gift

A charitable contribution.

amount The dollar value of the contribution.

cash? Whether the contribution was in cash (as opposed to in-kind).

name The name of the charity.

documented? Whether the donation was documented. See Pub. 526. This is checked if the contribution is over the $250 threshold.

B.6 Dependent

A dependent.

name The dependent's name.

ssn The dependent's social security number.

relationship The dependent's relationship with the form provider.

qualifying Whether the dependent qualifies for a tax credit. Options are "child" for the child tax credit, "other" for the credit for other dependents, and "none" where the dependent is not qualifying.

B.7 Estimated Tax

Estimated tax payment made to the IRS.

amount The amount of estimated tax paid.

B.8 HSA Contribution

Records of contributions to an HSA account.

ssn The SSN of the beneficiary of the HSA account.

contributions Contributions made to the HSA, but not including employer contributions or HSA/Archer MSA rollovers or HSA funding distributions.

B.9 Home Office

A portion of a home used as an office for a particular business.

type The type of business associated with this home office. Currently the only supported type is "partnership".

ein The EIN of the relevant business.

method Either "simplified" or "actual".

sqft Square footage of the home office area.

daycare? Whether the relevant business was a daycare.

property The name of the Real Estate form with which this home office is associated.

B.10 State Estimated Tax

Estimated tax payment made to a state.

amount The amount of estimated tax paid.

state The two-letter code for the state.

B.11 State Tax

A record of state taxes paid in the relevant tax year (usually for the previous year). This is the amount actually paid to the state tax office, (i.e., the amount due), not the total tax including what was already paid via withholdings.

amount The amount of tax paid.

name A description (not used for computation).

B.12 Traditional IRA Contribution

A contribution to a traditional IRA.

amount The amount contributed.

tax_year The tax year of the contribution.

date The date of the contribution.

B.13 Partner

A single partner within a partnership.

name The partner's name.

liability Either "general" or "limited".

nationality Either "domestic" or "foreign".

type The entity type of the partner. Currently only "Individual" is supported fully. Other possibilities include "Estate". (This should also, in the future, include corporations, partnerships, trusts, nonprofits, and foreign governments, among others.)

share This partner's share of profits and losses, as a decimal fraction such that all the partners' shares add to 1. Currently it is assumed that shares of profits and shares of losses are equal.

capital This partner's percentage contribution of capital to the partnership, as a decimal fraction such that all the partners' shares add to 1.

ssn The SSN of the partner.

country The country of citizenship of the partner, relevant to Form 1065 Schedule B-1, part II, column iii.

address The first line of the partner's address.

address2 The second line of the partner's address.

B.14 Partnership

Biographical information for a partnership.

name The name of the partnership.

address The first line of the address.

address2 The second line of the address.

business The line of business of the partnership, Form 1065 line A.

product The product or service of the partnership, Form 1065 line B.

code The business code, Form 1065 line C.

ein The employer identification number of the partnership.

start The start date of the partnership.

accounting "Cash", "Accrual", or other accounting method.

B.15 Real Estate

A description of real property, used in a variety of contexts and to link other forms that deal with a particular property.

property A name for the property that will uniquely identify the property across tax forms.

basis The basis price for the property, i.e., the purchase value. See Pub. 551.

sqft The square footage of the property.

purchase_date The date on which the property was purchased.