ABF: A system for automating document compilation

by JAMES SPROWL, MARTHA EVENS, and MOHAMED RAGAIE SAYED OSMAN
Illinois Institute of Technology
Chicago, Illinois
and
HENRY HARR
DePaul University
Chicago, Illinois

ABSTRACT

The ABF system, named after the American Bar Foundation, its first sponsor, was designed to aid lawyers and paralegals in the compilation of legal documents. The ABF interpreter processes a skeleton document from a library of templates, and automatically generates questions about whatever client-specific information is needed to produce a client-customized document, ready to be formatted. The user can interrupt the processor at any point to change the document; the system will then reprocess the new document using answers to questions asked previously. The templates in the ABF System Library may contain conditional expressions, loops, and complicated arithmetic expressions, like those sometimes needed in tax computations. Other documents may be included by reference. The goal is to make model documents written by legal experts available to lawyers in small offices and to law students.
INTRODUCTION

ABF is an expert system designed to assist attorneys in drafting legal documents. The system starts by extracting, from a library of legal forms, a skeleton template document that has embedded within it programming constructs such as conditionals, loops, references to other documents, and variables. The variables are later replaced by client-specific information in the course of an automated interview. Alternative and repetitive passages are included, or excluded, dynamically as the interpreter encounters conditionals and loops. The system analyzes the document and if it discovers that information is missing, it first checks to see if it already has the value stored, then it checks to see whether the document designer gave it a rule to compute the value, or supplied a special question that is to be displayed to the user when the value is needed. If these measures fail, the system generates an English question asking the user for the missing data. The user can stop the interpreter at any time, edit the draft document, and reinitiate automatic processing. Since the answers to previous questions are saved, the system will never ask a question a second time.

ABF was implemented in Pascal on an IBM PC in a joint venture with IBM.

The facilities that ABF offers to automate the assembly of legal documents will, we feel, make it just as useful in other applications, such as marketing, insurance, and hospital recordkeeping. We are currently using ABF in an experiment in the automatic generation of hospital discharge summaries.

HOW THE ABF SYSTEM WORKS

The ABF system library contains a number of document templates designed to help the law student learn how to frame simple documents of the kind that must be handled constantly in ordinary practice. One of those model documents is a simple will that begins:

I, [the name of the testator], do give my entire estate to my [the testator's spouse, a husband or wife] [the name of the testator's spouse], if my [the testator's spouse, a husband or wife] survives me by at least 30 days. If my [the testator's spouse, a husband or wife] [the name of the testator's spouse] predeceases me, then I give my entire estate to my children. . . .

Suppose the problem is to draft a will for a client named John Smith with a wife named Mary Smith and three children. To start this assignment, the user types into the system:

PROCESS Will OF John Smith

The words "PROCESS" and "OF" are typed in capitals, so that the system interpreter will recognize them as system command words.

On receiving this command, the ABF processor pulls out the model will and starts to process it. When it comes to the first set of square brackets, the processor stops and looks to see if it already knows the name of the testator. Since the processor does not know the name in this instance, the system displays the following question on the screen:

What is the name of the testator?

The user types in the answer:

John Smith

and the system displays on the screen:

I, John Smith, do give my entire estate to my wife, followed by a question about the next item of information it needs:

What is the testator's spouse, a husband or wife?

The user answers:

wife

The system now asks:

What is the name of the testator's spouse?

and the user answers:

Mary Smith

The system now displays:

I, John Smith, do give my entire estate to my wife Mary Smith, if my wife Mary Smith predeceases me, then I give my entire estate to my children. . . .

Since the processor saves the answers to the questions it asks, it does not need to ask for the same information again. Indeed the processor is asking for just the information the user would need to elicit in a client interview in order to draw up a will manually.

The version of the document produced by the system in asking questions and filling in the answers is called the DRAFT Will OF John Smith.
To see what the document will look like in finished form the user can issue the command

   DISPLAY DRAFT Will OF John Smith

and the system will display the draft on the screen. The user can make further changes by editing this draft using the command

   EDIT DRAFT Will OF John Smith

Once the document is in satisfactory shape, the user calls the formatter with the command

   FINISH DRAFT Will OF John Smith

This will cause the will to be printed out in a nicely formatted fashion.

Variable Names

The strings in square brackets are variable names. Because the variable names used in the questions generated by the system need to be self-explanatory, they are often long and cumbersome. To make the task of the document designer easier and to avoid misspellings, which confuse the system, the ABF system allows short names (abbreviations) to be associated with variable names. The association is declared by including both the short name and the long name in the square brackets, separated by a colon. Short names may not contain colons. For example, the document designer decides to use the abbreviation “testname” for the name of the testator, and “spname” for the name of the spouse, and “horw” for “the testator’s spouse, a husband or wife.” Our model document then becomes much shorter:

   I, [testname: the name of the testator], do give my entire estate to my [horw: the testator’s spouse, a husband or wife] [spname: the name of the testator’s spouse], if my [horw] survives me by at least 30 days. If my [horw], [spname], predeceases me, then I give my entire estate to my children.

The system stores both short and long variable names. It also stores the answers to questions as the values associated with the variable names. The long name is always used in asking questions, however.

Various Ways to Answer Questions

Most of the time the user will type in an answer and the system will accept that answer and insert it in the correct spot in the document. But occasionally the system receives an answer it does not understand. In that case, it will explain what kind of answer is appropriate and ask the same question again. For example, if the system expects a number and gets text instead, it will explain that a number is needed and then give the user another chance to answer the question.

If the system asks a question that the user is not ready to answer, the user types a question mark. Suppose the system asks for the address of the residuary legatee and the user does not know it yet. If the user types a question mark, the system will continue to process the will, without asking that question again. When the will is printed out, it will look perfectly normal except that where the address should be there will appear instead “[the address of the residuary legatee].” This address can easily be filled in using the editor as soon as the address is known, or the document may be reprocessed.

Help is also available. F1 brings up help with the editor, and F2 brings up help with the document processor. If a question is not clear, then the user presses the F2 function key. The system will display an assistance document at this point if it has one, that is, if the document designer realized that users might get confused at this point. Shift F1 and shift F2 bring up, respectively, the editor and the system help documentation in outline form.

Alternative Passages

The last sentence in our model will is not really appropriate if the testator is childless. This is one of many situations in the law when different circumstances call for alternative passages. To make our model more sophisticated, we can rewrite the last sentence to include:

   IF the testator HAS any children
       INSERT I give my entire estate to my children.
   END IF

When the system arrives at the variable named “the testator HAS any children,” it displays the question:

   Has the testator any children?

If the user answers this with a “yes,” this passage will be inserted.

   In a situation like this where the name of the variable is preceded and followed by a capitalized command word, it is unnecessary to enclose it in square brackets. We could have written this section without the brackets:

   IF the testator HAS any children
       INSERT I give my estate to my children.
   END IF

This variable is different from those we saw earlier in another way. Most variables have many possible values, but this has only two: true or false. Such variables are called propositions. The kind of “What is . . .?” questions that the system generates for other variables would sound very strange with propositions. If there is a capitalized word in the proposition, like the “HAS” in our example, it is moved to the front when the question is formed. If there is no capitalized word, the system asks the question: “Is it true that . . .?”

At some time we may want to insert one passage in one situation, but a different one in another situation. In our sample we will need to insert one passage if there is to be a single executor and a slightly different passage if there is to be
an executor and an alternate. ABF gives us a very simple way to express what we want to do here:

IF you DO wish to name only one executor, rather than an executor and an alternate
INSERT
I name as my executor [exnam: the executor's name] who shall not be required to post security upon [exposs: his or her, the possessive pronoun for the executor] bond.
OTHERWISE INSERT
I name [exnam] to be my executor, but if [expers: he or she, the personal pronoun for the executor] is unable or unwilling to serve, I name [exalt: the alternate executor's name] as executor. My executor shall not be required to post security upon his bond.
ENDIF

The ABF interpreter will ask the question:

Do you wish to name only one executor, rather than an executor and an alternate?

If the answer is “yes” the first passage will be inserted; if the answer is “no,” the second passage will be inserted.

At some time we may want to insert an already existing document in the middle of another document. For example, one or the other, or both, the alternative passages in the IF statement may be an already prepared document. To incorporate such a document by reference, simply place its name, enclosed in brackets, in the other document.

IF [the testator HAS any children]
INSERT [the gifts to minors passage]
OTHERWISE INSERT [the unborn child passage]
ENDIF

Document names are thus a kind of variable name.

Rules and Replacement Questions

The user of ABF will see the system mainly as a collection of nagging questions, so it is important to cut down the number of questions wherever possible and to make the remaining questions both clear and crisp.

The document designer may notice that the value of a variable can be computed by the system once the values of other variables are known. For example, if the system knows the value of [sex: the testator IS a man], then it should be able to compute the values of [spperspron: the personal pronoun of the testator's spouse, he or she] and [spposspron: the possessive pronoun of the testator's spouse, his or her]. The document designer writes a special rule to tell the ABF system how to perform the computation. The first step is to enter the command:

EDIT RULE spperspron

Then the designer enters the rule body:

IF [sex: the testator IS a man]
LET spperspron = (she)
LET spposspron = (her)
OTHERWISE
LET spperspron = (he)
LET spposspron = (his)
ENDIF

When the system needs the value of spperspron or spposspron, it will activate the rule instead of asking a question.

Sometimes a question cannot be avoided, but it can always be rephrased. The system-generated question may sound very awkward. From the variable “chnam: the names of my children” the ABF system will form the question:

What is the names of my children?

The document designer who looks at this question will doubtless decide to change it immediately by entering the command

EDIT QUESTION chnam

and then type in a substitute, perhaps:

List the names of the testator's children.

Client Data Files

As a document is processed, the system automatically writes a record of the values obtained for variables in a file. As John Smith's will is processed, the system produces a file called DATA OF John Smith. It will begin:

LET the name of the testator = (John Smith)
LET the testator's spouse, a husband or wife = (wife)
LET the name of the testator's spouse = (Mary Smith)

As you can see, this file takes the form of a rule. If it is necessary to stop in the middle of an interview, the client data file will be saved. When the system starts to reprocess the draft, the rule contained in John Smith's data file will be executed first and the values will be assigned to the variables. This means that the questions asked in the previous interview will not be asked again. The processor will start from the point where the previous interview finished.

The client data file also gives the user a way to correct earlier mistakes. If the user discovers that a previous answer was wrong, it is easy to change the values by editing the client data file. This may be done in mid-interview.

Constructing Lists and Repetitive Passages

Our model will also include an optional paragraph containing a list of special gifts of property. It uses a REPEAT clause to construct this list.
Does the testator wish to give another gift of property to someone?

The result of processing the REPEAT block is the addition of the following two sentences to the draft of John Smith's will:

I give my golf clubs to my nephew Russell Walton if living 30 days after my death. I give my coin collection to my cousin Peter Ames if living 30 days after my death.

The variables in the REPEAT block look distinctly different from those we have seen before; they are prefixed with a number sign. The function of the number sign is to indicate that we do not need one, but a number of copies of each variable in the REPEAT block. In fact, the system gives us a new copy of each variable every time it jumps back to the beginning of the block. In technical terms, a variable prefixed with a number sign is an array variable.

Why is this necessary? Consider what would happen if we had only one copy of the first variable [the testator DOES NOT wish to give another gift to someone]. Suppose we answered the question "Does the testator wish to give another gift to someone?" with a yes on our first trip through the REPEAT block. Then the second time through the block the system would look the variable up in its tables, discover that it already had a value, and start to process the rest of the block without asking the question again. It would do the same thing the third time, the fourth time, and so on. The processor would never escape from that block; it would go on asking about gifts until the user gives up in disgust and walks away.

If the number sign were omitted from another variable, the name of the recipient, for example, the system would be convinced after the first time through the block that it knew who the recipient was and would not ask that question again. All gifts would go to the same recipient.

A look at John Smith's client data file at this point would show that the processing of the REPEAT block added a lot of information:

ASSERT #1 the testator DOES wish to give another gift of property to someone
LET #1 the description of the gift = (golf clubs)
LET #1 the relationship of the recipient to the testator = (nephew)
LET #1 the name of the recipient = (Russell Walton)
ASSERT #1 the recipient IS an individual
ASSERT #2 the testator DOES wish to give another gift of property to someone
LET #2 the description of the gift = (coin collection)
LET #2 the relationship of the recipient to the testator = (cousin)
LET #2 the name of the recipient = (Peter Ames)
ASSERT #2 the recipient IS an individual
ASSERT #3 the testator DOES NOT wish to give another gift of property to someone
Proposed Extensions

Experience with earlier versions of ABF has already led to a number of design changes, particularly the integration of the editor with the system.¹² Our work during the last two years³⁴⁵ has suggested a number of additions to ABF. We want to add the ability to SELECT a passage from several alternatives, or even to RANDOMLY SELECT one, to facilitate CAI applications. We would also like to add the concepts MAY, MUST, SHALL, and SHOULD to the ABF language to increase our ability to model legal reasoning. We also feel that it is time to experiment with these same techniques in other applications.

OTHER APPLICATIONS OF ABF

The ABF system will prove useful in any situation where the user needs to write a series of customized documents, as in marketing or insurance. Imagine a marketing application in which the user is the owner of a computer store called PC World. An invoice document template is used to draw up invoices automatically, as it asks the client a series of questions about the configuration being purchased. This process will establish a client data file that can then be used to produce customized follow-up letters. A very simple invoice template might look like this:

```plaintext
[customer's name]
[customer's address]
IF pc: the customer DOES want a pc
   INSERT P C [modelno] [pcprice]
OR IF at: the customer DOES want an at
   INSERT A T [modelno] [at price]
END IF
IF the customer DOES want more memory
   INSERT additional memory [addon: the additional memory desired / 64000 * 49]
END IF
IF hard: the customer DOES want a hard drive
   INSERT hard drive [hard price]
OR IF [double: the customer DOES want two floppies]
   INSERT double floppy [double price]
OTHERWISE
   INSERT one floppy [single price]
END IF
IF mono: the customer DOES want a monochrome monitor
   INSERT monochrome [monoprice]
OTHERWISE
   INSERT color monitor [colorprice]
END IF
```

While the invoice is being processed, a client data file will be set up for this customer and kept in the library. This will allow us to process a customized follow-up letter without asking for further data at a later date.

```plaintext
IF [mono]
   INSERT [customer's name]
   [customer's address]
   Dear [customer's name]:
   On [date] you purchased
   IF [pc]
      INSERT a P C
   OR IF [at]
      INSERT an A T
   END IF
   with a monochrome monitor. Here at P C World we are running a special on color monitors at a one time price of [special price].
   END IF
IF memsize < 257
   INSERT if you want to extend your memory size from your initial [memsize] to 640K, a special added discount price of 20% will be available to purchasers of color monitors.
   END IF
```

This is just a simple illustration of the possibilities.

SUMMARY

The ABF environment permits one to design arbitrarily complex interviewing and document assembly systems for use in any application where standardized text is routinely assembled into customized drafts, in accordance with pre-defined rules. While this technology was originally developed to serve the specialized needs of the legal profession, it appears promising for applications in insurance and marketing. We are working on automating the construction of medical discharge summaries in using ABF also.

The ABF approach makes all the constructs of structured programming available to word processing specialists in a very supportive environment.

REFERENCES
