Behavior Research Methods, Instruments, & Computers
2002, 34 (1), 108-116

C-quence: A tool for analyzing qualitative sequential data

STARKEY DUNCAN, JR., and NICHOLSON T. COLLIER
University of Chicago, Chicago, Illinois

C-quence is a software application that matches sequential patterns of qualitative data specified by the user and calculates the rate of occurrence of these patterns in a data set. Although it was designed to facilitate analyses of face-to-face interaction, it is applicable to any data set involving categorical data and sequential information. C-quence queries are constructed using a graphical user interface. The program does not limit the complexity of the sequential patterns specified by the user.

Over time, studies of face-to-face interaction have proceeded from simple counts of actions by participants (e.g., Argyle & Cook, 1976; Harper, Wiens, & Matarazzo, 1978; Weitz, 1974) to methods for sophisticated analyses of action sequences (e.g., Bakeman & Quera, 1995; Magnusson, 2000). This report describes *C-quence*, a program designed to calculate rates of action sequences. In calculating these rates, there is no principled limit on the complexity of the action sequences specified for the numerator and the denominator. That is, action sequences can contain a large number of actions, a complete set of logical relations between these actions, and any number of occurrences of the actions.

C-quence is designed for studying interaction processes. For example, in studying such processes, the program may be used to examine such phenomena as the relative influence of each participant on the other, the amount of change of influence over time, the effect of certain actions or sequences of actions on changing or preserving the current course of the interaction, and whether or not the interaction process is Markovian. For phenomena such as these, C-quence can be used either to test specific hypotheses, or to pursue exploratory studies. In any event, C-quence can be used as appropriate in any data set containing categorized events and information on the sequences in which these events occur.

C-quence is intended to complement and extend the programs of Bakeman and Quera (1995) and of Magnusson (2000), in that it permits asking questions of sequential data that are qualitatively different from those asked by the preceding programs. Magnusson's *Theme* searches for sequential patterns based on expected proximity of actions within sequences and builds more complex patterns on the basis of simpler ones. Bakeman and Quera's *SDIS* and *GSEQ* perform, among other things, time-series analyses. C-quence returns the rates of specified action sequences (numerator), given other action sequences (denominator). For this reason, it may prove useful to investigators. Like Magnusson's and Bakeman and Quera's programs, C-quence was developed for studies of face-to-face interaction. However, it can be used with any data set containing categorical data and having information on the sequences in which the categorized actions occurred.

The program will be described in general terms. Its use will then be illustrated by applying it to the case of a specific interaction: a mother spoon-feeding her 10-month-old son.

In the following discussion, the term *instance* is used to refer to the occurrence of a particular type of interaction, such as spoon-feeding, games, or discipline in a family, and exchange of speaking turns in conversations between adults. Because discipline often occurs at various times during the day, each discipline interaction would be termed an instance. Other interactions, such as spoon-feeding and games, tend to occur repeatedly one after another. For example, turns in a game may be repeated or alternated a number of times in succession (Bruner, 1985; Duncan & Farley, 1990). Similarly, spoon-feeding occurs repeatedly at short intervals during a meal. It is convenient in this case to refer to instances of spoon-feeding and games as *cycles*. A spoon-feeding cycle is defined later in the Examples of Queries section.

C-QUENCE

C-quence is a software application designed to search for user-defined sequences of actions in variable-length occurrences of those actions and to calculate the rates of

Preparation of this paper was supported in part by National Institute of Mental Health Grant MH-38344, McCormick-Tribune grant to Starkey Duncan, Jr., and Social Science Research Computing at the University of Chicago. From conceptualization to implementation, this project has been the beneficiary of the generous suggestions from many colleagues, including Lawrence J. Brunner, Fraeda Friedman, Larry Hedges, Thomas R. Howe, I. Riley Jackson, Thomas McDougal, Paul Sallach, and Christine Winkowski. The authors are deeply grateful to participants and family members for their hospitality and patience during the videotaping process. Correspondence should be addressed to S. Duncan, Jr., Department of Psychology, University of Chicago, 5848 S. University Ave., Chicago, IL 60637 (email: dunc@ccp.uchicago.edu).

these sequences within a data set. In calculating rates, the user can define sequences of actions for both the numerator and the denominator.

These abstract properties can be described more specifically in terms of research on face-to-face interaction for which C-quence was designed. Assume that the investigator has observed one or more occurrences of a particular type of interaction, such as a conversation or the spoon-feeding interaction described later. On a more detailed level, one might observe a number of instances of a particular type of event within a single interaction. Here, examples would be exchanges of speaking turns and interruptions within a conversation, or the cycles of spoon-feeding within a meal, where a *cycle* is defined as beginning when the mother spoons food from a container, and ending when the spoon is returned to the container, regardless of whether or not the infant has been fed.

Regardless of the interaction process involved, the investigator's observations are based on a set of qualitative categories such that, at any moment in the interaction, the investigator can note those categorized actions that are occurring. In addition, information is retained on the sequence in which these actions occur.

On the basis of the investigator's observations, a database is created containing (1) a set of instances of the interaction being observed (e.g., spoon-feeding cycles) and (2) the sequence of actions occurring within each of these instances. The number of actions observed may vary from one instance to another. Furthermore, for each instance, it is necessary to include only those actions that actually occurred. Other actions that might have, but did not, occur in an instance may simply be omitted from the sequence of actions for that instance.

C-quence permits calculations of rates of sequences defined by the investigator. The investigator's definition of the sequence of interest determines a numerator, and the definition of instances that provide an opportunity for this sequence to occur determines a denominator. (The denominator may be either all the instances in the database or some subset of them.) As will be illustrated in a later section, the investigator specifies an action or action sequence for both the numerator and the denominator in the rate calculation. There is no principled limit on the length or complexity of these action sequences. Specified sequential actions need not be adjacent to each other, and it is possible to specify that certain actions not occur in the sequence in question.

Specifying the numerator and the denominator for desired rates is accomplished using a graphical user interface. This interface has been designed to make C-quence easy and intuitive to use, as well as to minimize user error.

C-quence was written in Java and will run on any platform with a Java 1.3 compatible virtual machine (among other platforms, this includes Windows 9x/2000/NT, and most varieties of Unix, including Linux). When running on Intel-based hardware, a Pentium 200 (or higher) with at least 32 MB of RAM is recommended.

Constructing Queries

Both numerator and denominator are specified in the same way. We shall term the specification of a rate calculation as a *query*. Although each query may consist of single actions, C-quence is designed to handle sequences. These sequences are composed of two parts: (1) a set of actions joined by the Boolean operators "AND," "NOT," "OR," and "XOR," and (2) the number of times (if at all) the actions must occur to satisfy the query. Specifying the number of times is accomplished by the arithmetic operators "=," ">," "≥," "<," and "≤," followed by a number N, where $N \geq 0$. For each query, the investigator thus indicates what actions to search for and how many times (if at all) that action must be found to satisfy the query. There is no limit on the number of actions in a query. The following examples apply to specifying both the numerator and the denominator of the query.

The expression "$X > 2$" means find those instances of the interaction in which the action X occurs more than two times. The expression "$X < 3$" means find instances in which X occurs less than three times. "$X = 0$" means find instances in which X does not occur.

The "AND" and "NOT" Boolean operators specify the order in which the actions are to be found relative to each other, regardless of any intervening nonspecified actions. For example, the query "$X > 2$ AND $Y = 1$" means look for instances in which action X occurred at least three times, then, within those instances and beginning with the action after the third X, find those instances in which Y occurred once and only once. In this case then, the sequence "*X, Z, X, Y, X, Y, Z*" would match, and the sequence "*X, Z, X, Y, X, Y, Z, Y*" would not. In this latter case, the search fails because the action Y is found twice after the requisite three Xs are found.

The "NOT" operator works similarly and is in fact shorthand for "AND NOT." Consequently, a query such as "$X > 2$ NOT $Y = 1$" means find action X at least three times and then find Y not one time (i.e., more than once or not at all). The absence of an action can also be specified while preserving this notion of relative order. For example, the query "$X > 0$ AND $Y = 0$ AND $Z > 1$" means find action X at least once and then find Z at least twice while making sure that action Y does not occur between the first X and the second occurrence of Z.

The Boolean operators "OR" and "XOR" specify different kinds of alternation. For example, the query "$X > 2$ OR $Y = 1$" means find action X at least three times or find action Y once and only once. "$X > 2$ XOR $Y = 1$" means find either action X at least three times or action Y once and only once, but not both.

Parentheses. Sets of actions can be parenthesized through the user interface, changing the order in which action sequences are evaluated. For example, without parentheses, the query "$X > 0$ AND $Y > 0$ OR $Z > 0$" means find action X at least once and then find action Y at least once in the same instance, or find action Z at least once. Similarly, a query such as "($X > 0$ OR $Y > 0$) AND ($A >$

0 OR $B > 0$)" evaluates to "($X > 0$ AND $A > 0$) OR ($X > 0$ AND $B > 0$) OR ($Y > 0$ AND $B > 0$)."

Numerator-only or full searches. Queries may indicate sequences for the numerator only or for both numerator and denominator (a full search).

Numerator-only searches. A numerator-only search runs the specified query against each instance of the interaction, attempting to match the pattern specified by the query. It then reports the total number of matches divided by the total number of instances in the database. In the simplest case, one could ask, for example, for all spoon-feeding cycles in which the infant is actually fed. C-quence would return this number, the number of cycles in the database, and the ratio of these two numbers.

Full searches. When the investigator specifies both the numerator and the denominator, C-quence first searches for the specified denominator. Then, for all instances in which the denominator specification is satisfied, C-quence performs the numerator search. C-quence then reports (1) the number of successful denominator searches, (2) the number of successful numerator searches, and (3) the percentage of successful numerator searches, given a successful denominator search. Thus, for all instances in which *X* (the denominator) occurs, a full search answers the question, How often does *Y* (the numerator) occur?

In a full search, the point at which the numerator search begins is customizable. In a full search, once the denominator specification is satisfied for an instance, the numerator search can begin (1) at the beginning of the instance (labeled in the options screen "AT FIRST ACTION"), (2) at the point where the denominator search is satisfied ("ACTION WHERE DENOMINATOR SEARCH SATISFIED"), or (3) immediately after the point where the denominator search is satisfied ("ACTION IMMEDIATELY AFTER DENOMINATOR SEARCH SATISFIED").

Specifying where the numerator search begins has some important ramifications and adds power and flexibility to the queries. Consider a query in which the denominator is "$X = 2$ AND $Y > 0$" and the numerator is "$Y > 1$." Given the sequence "*X, Z, Y, X, Y,*" and a numerator search beginning "AT FIRST ACTION," the query will succeed. In this case, the denominator search succeeds in finding action *X* twice and only twice and then action *Y* at least once; then, the numerator search, returning to the beginning of the sequence, succeeds in finding *Y* at least twice. In fact, *Y* is found three times when the search is begun "AT FIRST ACTION."

The query also succeeds when the numerator search begins where the denominator search was satisfied ("ACTION WHERE DENOMINATOR SEARCH SATISFIED"). Here, the denominator search succeeds as above when the penultimate action, *Y*, of the instance is found. Beginning the numerator search at this point, a *Y* is found (the same *Y* that satisfied the denominator search), and then another *Y* is found, and the numerator search also succeeds.

However, when the numerator search is begun at the action immediately following the action that satisfies the denominator search ("ACTION IMMEDIATELY AFTER DENOMINATOR SEARCH SATISFIED"), the query fails. The denominator search succeeds as above, but the numerator search then starts after the penultimate *Y* that satisfied the denominator search, and only a single *Y* is found before the end of the instance. Consequently, the query fails.

Implementation

In a full search, C-quence processes a user's specified queries by building two parse trees for the numerator and denominator searches. The structure of the tree is roughly equivalent to the structure of a search itself. The topmost node is an expression that represents the search in its entirety and is typically an expression that represents the joining of other expressions by the Boolean operators. This top expression may or may not itself be made up of additional Boolean expressions. Boolean expressions are themselves made up of subexpressions, either additional Boolean expressions or individual or terminal expressions. Individual or terminal expressions contain an element and a count of that element, such as "$X = 1$" or "$Y > 2$." Boolean expressions also coordinate how their subexpressions process a query. For example, a sequence expression represents two clauses joined by the Boolean operator "AND." This expression will contain two child or subexpressions and will ensure that both these subexpressions succeed in their searches before it itself succeeds.

An example of a parse tree is shown in Figure 1. Here, the topmost expression is "$X = 1$ AND ($Y > 2$ OR $Z = 1$)." This is a sequence expression composed of an equals expression, "$X = 1$," and an alternation expression, "($Y > 2$ OR $Z = 1$)." The equals expression is a terminal expression—that is, it cannot contain any child or subexpressions and is composed of an element "*X*" and a count "1." The count is the number of times the element should appear. The alternation expression contains two terminal expressions: a greater-than expression and an equals expression.

Once such a tree is built, C-quence iterates through the set of instances, searching each instance. A single instance is passed to the top expression of the parse tree for processing. This top expression in turn passes the instance to its subexpressions for processing. This is known as *walking* or *traversing* the parse tree. The order of traversal is a modified *preorder*—that is, visit the root, then visit the left subtree, then visit the right subtree, and then visit the parent of those subtrees. In our example parse tree in Figure 1, the order of traversal is "$X = 1$ AND ($Y > 2$ OR $Z = 1$)," "$X = 1$," "$Y > 2$ OR $Z = 1$," "$Y > 2$," "$Z = 1$," "$Y > 2$ OR $Z = 1$," and "$X = 1$ AND ($Y > 2$ OR $Z = 1$)." (Depending on the type of parent expression, C-quence may not visit all the subtrees. For example, if the left side of an alternation expression is true, then there is no need to test the right side.) In our example, the first terminal expression that we reach is the leftmost equals expression, "$X = 1$." When the search reaches a terminal expression such as this equals expression, the terminal expression will

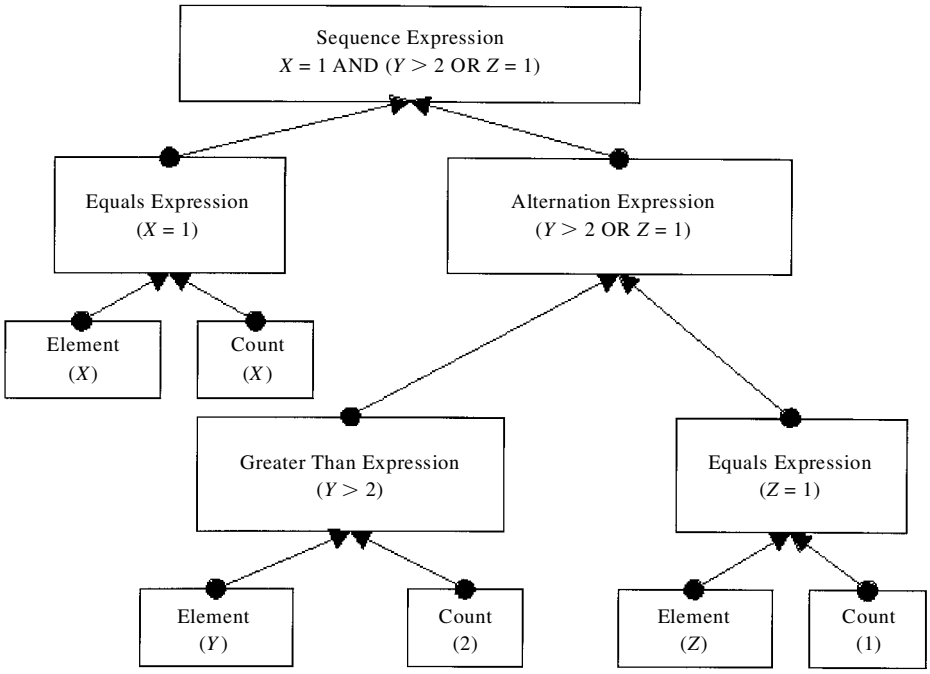

Figure 1. C-quence parse tree for a simple expression.

iterate through the actions of the instance until it achieves a match or the instance ends. In our case, it will look for a single "*X*" element in the instance. If this is not found, the search of that instance terminates, and a new search begins with a new instance. If the search succeeds, then C-quence continues to walk the tree—in this case, moving on to the alternation expression. The success of nonterminal expressions such as this alternation expression depends on their type and the success of their component terminal expressions. For example, our alternation expression will succeed if either the greaterThan terminal expression ("$Y > 2$") or the equals expression ("$Z = 1$") succeeds. Assuming success, C-quence then walks back up the tree to the sequence expression and checks it for success. If the topmost expression is a success, then the search itself succeeds for that instance.

There are a few complications in walking the tree, however. In addition to performing their own search, terminal expressions will also look for any actions that might cause a previous expression to fail. For example, if a sequence expression represents "$X = 1$ AND $Y > 1$," the sequence expression will pass the instance of actions to the terminal expression representing "$X = 1$." Assuming this succeeds, the sequence expression will then pass the instance to the terminal expression representing "$Y > 1$," along with an index into the instance identifying where to start the search. This terminal expression will then look for *Y*s while making sure that no additional *X*s occur. The search results (success or failure) are passed up the tree. C-quence totals up the number of successful searches and returns the result to the user.

User Interface

The user interface was designed to be as intuitive as possible, and to minimize user error in constructing queries. Figure 2 shows the C-quence graphical user interface with the drop-down combo box showing the available actions for the example provided in the next section. The numerator and the denominator for each query are specified via the same process. Specifications for numerator and denominator are created by choosing desired actions and arithmetic operators from the combo boxes, and entering, the number of actions to find in a text box.

Actions can be added to the query by clicking the "More" button that displays a combo box identical to and below the first one. The investigator again specifies the actions and Boolean operators. The user can then choose the appropriate Boolean operator and customize the new clause as desired. This process can be repeated as many times as required.

Levels of parenthetical nesting are visually represented through indentation and controlled by the left and right arrow buttons that appear next to a clause. To delete a clause, the user can click on it to highlight it and then click the "Delete" button.

The various menus allow the user (1) to save a query together with some explanatory comments, (2) to load a previously saved query, and (3) to edit C-quence properties, such as the type of search, where the numerator search should start, where to save the results of queries, and so forth. The "Tools" menu allows the user to load a different database, to import data files into the format used by C-quence, and to define the actions themselves. Figure 3

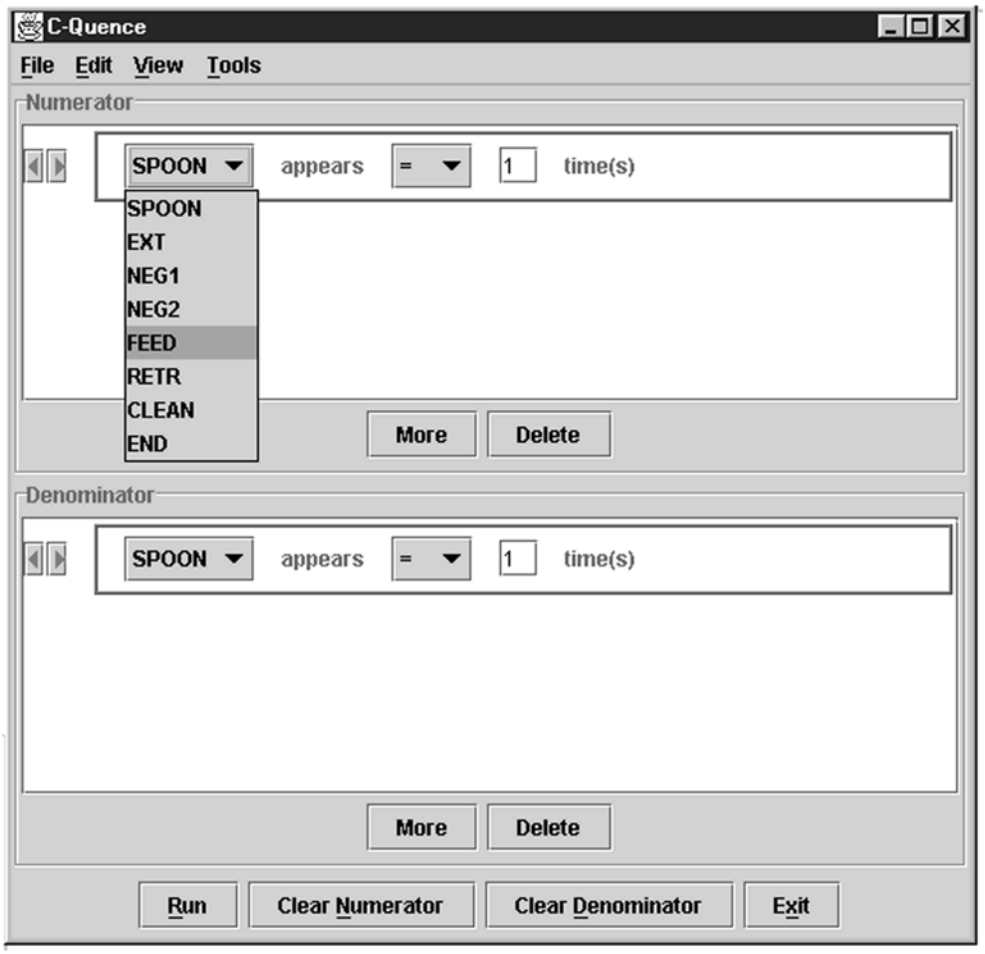

Figure 2. C-quence menu for specifying numerator and denominator for searches (numerator drop-down combo box shown).

shows the "Edit" menu in which the investigator can specify the search type and the point at which the numerator search should start.

C-quence displays the results of a query in the results window (see Table 1) and saves these results to a user-specified file in a tabular, tab-delimited format suitable for importing into a wide variety of applications.

EXAMPLES OF QUERIES

As mentioned earlier, use of C-quence will be illustrated by application to a spoon-feeding interaction. In studying this interaction, initial exploratory research examined the possibility that it was convention-based or rule-governed. Sequential regularities in the actions of mother and child supported this possibility (Friedman, 1996; Friedman, Duncan, & Hedges, 2002). The rules and other elements hypothesized for an observed interaction will be termed a *structure*.

It is convenient to use flowcharts to represent structures. In these flowcharts, rectangular shapes indicate *obligatory actions*: those that must occur at that point in the interaction. The diamond shapes indicate *optional actions*: points at which the indicated participant has a choice between two actions or between whether or not to enact an action. As the flowchart indicates, this choice has an effect on the immediately ensuing course of the interaction.

Figure 4 provides an example of the structure hypothesized for a spoon-feeding interaction between a mother and her son observed on videotapes made over a 2-day period when the child was 10 months old (Friedman, 1996; Friedman et al., 2002). (Extensive discussions of the notion of convention-based interaction and methods for studying it can be found in Duncan, 1991, Duncan & Fiske, 1977, and Duncan, Fiske, Denny, Kanki, & Mokros, 1985.) The structure applies to one cycle of the spoon-feeding, defined as beginning when the mother spooned food from a container and ending when she returned the spoon to the container, whether or not the child had been fed.

The structure fit all 277 uninterrupted cycles observed on the videotapes made at 10 months. A *fit* is defined as

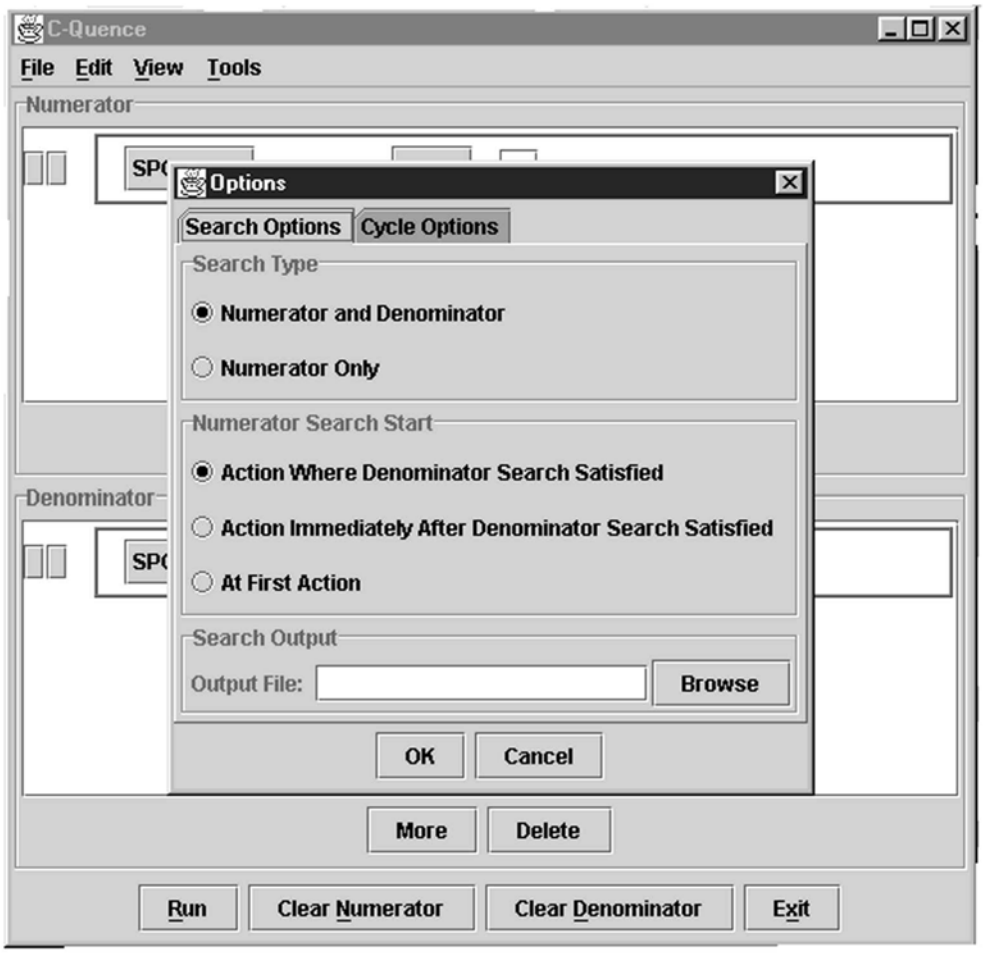

Figure 3. C-quence edit menu specifying type of search and point at which the numerator search should start.

a complete correspondence between the transcribed data and the structure. Any single discrepancy between the structure and the transcribed data for a cycle is considered a failure of the model to fit that entire cycle. Not counted as a failure to fit would be certain events external to the feeding that resulted in an *interruption* of the interaction. In the case of these data, there was one interruption: The father entered the kitchen and began to talk to the mother. The mother returned the spoon to the container before feeding and did not begin to feed again until the father left. This interruption was removed from the data. Thus, there was a total of 278 cycles observed when the child was 10 months old, but 277 uninterrupted cycles.

This particular spoon-feeding interaction was of interest because it was frequently highly conflictual, unlike most

Table 1
Illustrations of C-quence Query Results

Line	Rate	Numerator	Denominator	Query	Type of Search
1	1.00	277	277	SPOON = 1	NUMERATOR SEARCH
2	1.00	157	157	FEED >= 1 / NEG1 = 0 AND NEG2 = 0	FULL SEARCH; STARTS AT SATISFIED
3	.60	6	10	FEED >= 1 / NEG2 >= 3	FULL SEARCH; STARTS AT SATISFIED
4	1.00	144	144	FEED >= 1 / CLEAN >= 1	FULL SEARCH; STARTS AT START
5	.14	20	144	FEED >= 1 / CLEAN >= 1	FULL SEARCH; STARTS AT SATISFIED
6	.35	25	71	NEG2 >= 2 / NEG1 = 0 AND NEG2 >= 1 AND EXT >= 1	FULL SEARCH; STARTS AT START
7	.35	25	71	NEG2 >= 1 / NEG1 = 0 AND NEG2 >= 1 AND EXT >= 1	FULL SEARCH; STARTS AFTER SATISFIED

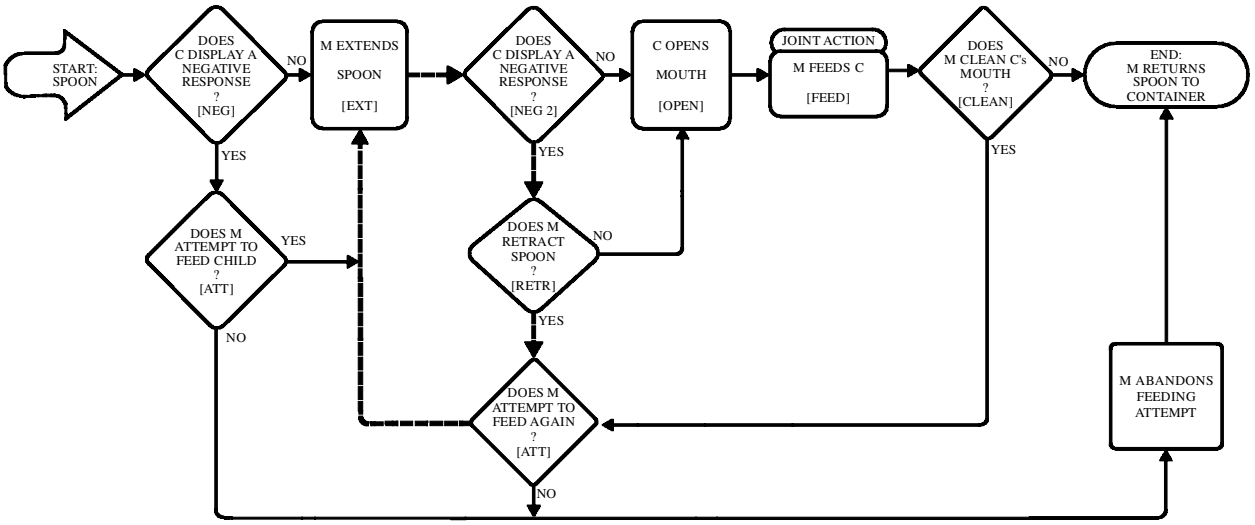

Figure 4. Hypothesized structure for spoon-feeding Alex (9–12 months of age).

spoon-feedings in other families on our videotapes. The child often chose to display a wide variety of resistive actions (e.g., crying, screaming, waving his arms, averting his head, and hitting at the extended spoon). These actions are represented by the optional element: "Does child display a negative response?" As mentioned in the preceding paragraph, the grouping of these actions in the "negative response" element was based on sequential analysis, rather than intuition. This element is considered to occur when any one or more of its constituent actions were displayed by the child. The element appears twice in the flowchart because it has a different effect on the course of the interaction depending on whether it occurs before or after the mother's extending the spoon. In the following discussion, we shall refer to the negative response occurring before the extension as *NEG1* and the negative response occurring after the extension as *NEG2*, as shown in Figure 4.

The top line of the flowchart represents the simplest feeding sequence: The child is fed with no negative responses. As indicated in the flowchart, when there is a negative display by the child, the mother might respond by choosing (1) to persist in attempting to feed the child, or (2) to abandon the current feeding attempt and end the cycle.

It may be seen that there are multiple paths through the flowchart, each path being jointly constructed through the choice of actions by both mother and child. The term *strategy* will be used to denote the process by which actual interactions are jointly constructed by the participants within the framework of the structure. This particular structure provides for considerable variation in the type and number of actions making up a cycle (instance).

Among other things, interaction structures can be used to analyze the patterns of influence that each participant exerts on the partner as they progress through an interaction—in this case, a single cycle of spoon-feeding. In the context of early socialization, it becomes interesting to examine processes of bidirectional influence between caregiver and child (e.g., Anderson, Lytton, & Romney, 1986; Bell, 1968; Cohn & Tronick, 1988; Dunn, 1988; Gottman & Ringland, 1981; Gralinski & Kopp, 1993; Grusec & Lytton, 1988; Kuczynski, Kochanska, Radke-Yarrow, & Girnius-Brown, 1987; Lytton, 1980; Maccoby & Martin, 1983; Patterson & Bank, 1989; Zahn-Waxler & Chapman, 1982).

Note that the flowchart has three loops, each involving either a conflict or a potential conflict. Following the standard mathematical definition, a *loop* is defined as a semipath in a flowchart (or directed graph) that returns to the point at which it began. We shall focus on the most frequently traversed loop indicated by dashed lines. This involves the mother's extending the spoon (EXT), the child's displaying a negative response (NEG2), the mother's momentarily retracting the spoon (RETR), and then extending it again (EXT) in a continued attempt to feed

SPOON EXT FEED CLEAN END
SPOON NEG1 END
SPOON NEG1 EXT FEED CLEAN END
SPOON EXT NEG2 RETR EXT NEG2 RETR EXT NEG2 RETR EXT NEG2 RETR EXT NEG2

Figure 5. Illustration of lines from a C-quence database.

the child. For convenience, we shall term this *Loop A*. The loop can be exited either by the child's acceding to the mother and accepting the food, or by the mother's abandoning her attempt to feed on that particular cycle. C-quence queries indicate that, when the child was 10 months old, this loop occurred from 0 to 4 times in a cycle.

The database for this structure contains the set of sequentially ordered elements occurring on each cycle (instance) of the spoon-feeding interactions observed on the videotapes. The flowchart elements in the database are indicated by their abbreviations within parentheses in the shapes. Figure 5 illustrates lines from this database. Each line represents a cycle of spoon-feeding.

Table 1 illustrates results from queries of the database for spoon-feeding in this family when the child was 10 months old. These results are just as they are returned by the program. More complex queries can be built by adding specifications to the numerator and the denominator. Rates have been truncated to two decimal places.

As already described, queries begin with the denominator. The cycles satisfying the denominator specification are then searched to find those satisfying the numerator specification. C-quence returns (1) the rate, (2) the numerator, (3) the denominator, (4) the query itself with Boolean and arithmetic operators, (5) the type of search (numerator only or full), and (6) the point at which the numerator search starts in a cycle, given that the denominator search has been satisfied.

Line 1 is a simple numerator search that, in effect, counts the number of cycles in the database. Because "SPOON" begins each cycle, the numerator search is satisfied on every cycle. There are 277 cycles.

Line 2 follows the flowchart across the top line, showing that, when there are no negative displays, the child is always fed. This path is followed 157 times. "FEED" is specified as "≥ 1" because the child might be fed again after the mother cleans his face. "FEED = 1" would exclude cycles having the FEED $\rightarrow$ CLEAN sequence.

Line 3 shows that, when there are three or more displays of NEG2—that is, at least two transits of Loop A—the rate of feeding drops appreciably, although the number of times there are at least two transits of the loop is not large. Note that the query ignores the presence or absence of NEG1 in a cycle. The effect of NEG1 can be assessed by including it in the query.

The query in Line 4 indicates that, after the denominator "CLEAN" is found at least once in a cycle, the search returns to the beginning of the cycle to find one or more "FEEDs." This query involves an antecedent probability: Given that there is cleaning, what is the probability that it was preceded by feeding? It is not surprising that cleaning occurred only after feeding.

By changing the point at which the numerator search begins, Line 5 asks a different question concerning feeding and cleaning: Once there is at least one "CLEAN," how many times does the mother succeed in feeding the child again? In this case, the numerator and the denominator are the same as in Line 4, but the point at which the numerator search begins is different. The result contrasts sharply with that of Line 4.

Line 6 asks for the number of times that, after at least one Loop A, the child displays another NEG2, given that there is no NEG1 in the cycle. In this query, the denominator requires that (1) there is no NEG1, (2) there is at least one NEG2, and (3) the mother extends at least once again after the initial NEG2 (as opposed to ending the cycle), thus completing Loop A. For those cycles in which the denominator search is satisfied, the numerator search returns to the beginning of the cycle ("AT FIRST ACTION"), requiring that there are at least two NEG2s. That is, the search includes the NEG2 found in the denominator search, plus at least one more NEG2. Thus, the first NEG2 is included in both the denominator and the numerator. (The alternative to an additional NEG2 is that the child simply accepts the food on the mother's second extension.)

Obviously, the same query can be specified in different ways. Line 7 illustrates a different way of making the same query as that described in Line 6. However, in this case, the search specifies the numerator search begins with "ACTION IMMEDIATELY AFTER DENOMINATOR SEARCH SATISFIED." Accordingly, the denominator search remains the same, but the numerator specification is changed to "≥ 1." As a result, the first NEG2 is counted only in the denominator. After this, at least one more NEG2 need be found.

Considering the number of paths and semipaths through the flowchart, there are many other questions that can be asked of the data. Perhaps more importantly, when interaction structures persist in the same form across months, as this one does between 9 and 12 months of age, it becomes possible to analyze the way in which rates of various sequences change longitudinally. That is, even when an interaction structure remains the same, strategies within the structure may change, revealing developmental effects. By the same token, two or more conversations between adults may follow the same rules for turn-taking, but they may differ in strategy, such as the rates of attempts to take the turn, interruptions, and the like. Similarly, when different families have the same structure for a given type of interaction, it becomes possible to analyze the differences in strategy, if any, between the families. In the case of the interaction used as an example, C-quence is being used in an ongoing study to quantify bidirectional influence over the course of a spoon-feeding cycle.

Conclusion

C-quence is designed to answer both simple questions and questions of complexity limited only by the database being analyzed. It can be applied to any sort of interaction, including both human and nonhuman interactions. The interaction structure was used only as an example and is not a prerequisite for C-quence use. It is hoped that C-quence will provide a powerful tool for investigators of interaction and other sequential phenomena, complementing existing programs in the area.

Availability

The software described in this report is available for public examination and nonprofit use under the Gnu General Public License (GPL). The terms of the GPL open source license are described at the Web site http://www.gnu.org/copyleft/gpl.html. The code may be downloaded using anonymous FTP from ftp.src.uchicago.edu/pub/C-quence/cquence.zip.

REFERENCES

Anderson, K. E., Lytton, H., & Romney, D. M. (1986). Mothers' interactions with normal and conduct-disordered boys: Who affects whom? *Developmental Psychology*, **22**, 604-609.

Argyle, M., & Cook, M. (1976). *Gaze and mutual gaze*. London: Cambridge University Press.

Bakeman, R., & Quera, V. (1995). *Analyzing interaction: Sequential analysis with SDIS and GSEQ*. New York: Cambridge University Press.

Bell, R. Q. (1968). A reinterpretation of the direction of effects in studies of socialization. *Psychological Review*, **75**, 81-95.

Bruner, J. S. (1985). The role of interaction formats in language acquisition. In J. P. Forgas (Ed.), *Language and social situations* (pp. 31-46). New York: Springer-Verlag.

Cohn, J. F., & Tronick, E. Z. (1988). Mother–infant face-to-face interaction: Influence is bidirectional and unrelated to periodic cycles in either partner's behavior. *Developmental Psychology*, **24**, 386-392.

Duncan, S. D., Jr. (1991). Convention and conflict in the child's interaction with others. *Developmental Review*, **11**, 337-367.

Duncan, S. D., Jr., & Farley, A. M. (1990). Achieving parent–child coordination through convention: Fixed and variable-sequence conventions. *Child Development*, **61**, 742-753.

Duncan, S. D., Jr., & Fiske, D. W. (1977). *Face-to-face interaction: Research, methods, and theory*. Hillsdale, NJ: Erlbaum.

Duncan, S. D., Jr., Fiske, D. W., Denny, R., Kanki, B. G., & Mokros, H. B. (1985). *Interaction structure and strategy*. New York: Cambridge University Press.

Dunn, J. (1988). *The beginnings of social understanding*. Cambridge, MA: Harvard University Press.

Friedman, F. (1996). *Spoon-feeding in mother–infant interaction from 9 to 12 months*. Unpublished master's thesis, University of Chicago, Chicago.

Friedman, F., Duncan, S. D., Jr., & Hedges. L. (2002). *Analyzing caregiver–infant bidirectionality: Interaction structure and strategy*. Manuscript submitted for publication.

Gottman, J. M., & Ringland, J. T. (1981). The analysis of dominance and bidirectionality in social development. *Child Development*, **52**, 393-412.

Gralinski, J. H., & Kopp, C. B. (1993). Everyday rules for behavior: Mothers' requests to young children. *Developmental Psychology*, **29**, 573-584.

Grusec, J. E., & Lytton, H. (1988). *Social development: History, theory, and research*. New York: Springer-Verlag.

Harper, R. G., Wiens, A. N., & Matarazzo, J. D. (1978). *Nonverbal communication: The state of the art*. New York: Wiley.

Kuczynski, L., Kochanska, G., Radke-Yarrow, M., & Girnius-Brown, O. (1987). A developmental interpretation of young children's noncompliance. *Developmental Psychology*, **23**, 799-806.

Lytton, H. (1980). *Parent–child interaction: The socialization process observed in twin and singleton families*. New York: Plenum.

Maccoby, E. E., & Martin, J. A. (1983). Socialization in the context of the family: Parent–child interaction. In E. M. Hetherington (Ed.), *Handbook of child psychology: Vol. 4. Socialization, personality, and social development* (pp. 1-101). New York: Wiley.

Magnusson, M. S. (2000). Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods, Instruments, & Computers*, **32**, 93-110.

Patterson, G. R., & Bank, C. L. (1989). Some amplifying mechanisms for pathologic processes in families. In M. R. Gunnar & E. Thelen (Eds.), *Minnesota Symposia on Child Psychology: Vol. 22. Some amplifying mechanisms for pathologic processes in families* (pp. 167-209). Hillsdale, NJ: Erlbaum.

Weitz, S. (Ed.) (1974). *Nonverbal communication: Readings with commentary*. New York: Oxford University Press.

Zahn-Waxler, C., & Chapman, M. (1982). Immediate antecedents of caretakers' methods of discipline. *Child Psychiatry & Human Development*, **12**, 179-192.

(Manuscript received February 28, 2000;
revision accepted for publication August 4, 2001.)