Sd

A Square Dance Caller's Helper

William B. Ackerman and Stephen Gildea

July 12, 1996

for Sd version 31.72

Table of Contents

Introduction

Sd is a square dance caller's helper. The program assists you in writing sequences for Western square dancing by doing the checker pushing. You tell the program what call you want to call next, and it computes the resulting setup and shows it to you.

Sd is intended to be used to write challenge-level dances where the sequences are often complex and the checker pushing tedious and error-prone. Most challenge callers write out the sequences they will call before they get to the dance, unlike Mainstream callers, who often invent the sequences on the fly from the stage.

What Sd Is Not

Sd knows nothing about timing, body flow, or esthetics.

This program is not appropriate for Traditional square dancing. A good Traditional square dance sequence requires, among other things, extremely accurate control of timing and phrasing. The program is not capable of this.

Since the emphasis here is on checker pushing, you will find various Mainstream staples missing, such as Circle Left, Grand Square, and Do-Si-Do. Since these calls are *zeros*, there is no reason to have them in a checker-pushing program. However, if you wish to write sequences containing calls such as these, you can use the `insert comment` command to write them into the sequence.

In short, if you are writing a Traditional or Mainstream dance, you may find that this program is not for you.

It is most emphatically not the goal of Sd to present a polished graphical display of the dancers, or to show animation. The goal is to write sequences so that you can create pleasing animation among live people. If you want to enjoy the esthetic visual patterns created by square dance choreography, we recommend that you turn off your computer and go to a dance.

The use of this program is not a good way to learn to call or to improve your calling or dancing skills. In fact, reliance on a computer program to write material could easily make you a worse caller, or interfere with your attempts to improve your calling skills. While a program may be a useful reference at times for calls or concepts that you do not understand, this is only a tiny part what you need to learn in order to be a good caller or dancer. To call successfully, you need to master many skills, such as timing, flow, judging difficulty, floor interaction, and choosing precisely the right words to say. This last skill becomes increasingly important at higher levels. Subtle differences in the words you choose, and their timing and inflection, can have a tremendous influence on the success of the dancers. Sd attempts to print the correct words in all cases, but you will only succeed if you have a deep understanding of what the words mean and how they should be delivered. That understanding can only be obtained through a great deal of experience calling at that level.

If you do not have a good understanding of a call or concept, such that you could explain it to a dancer after the tip is over, you should not use it. Do not rely on Sd to provide the necessary insight into challenge dancing.

For beginning callers in particular, the best thing to do is to receive instruction from a qualified teacher or coach, and to practice.

This is not to say that computers have no place in the education and training of callers. A number of caller training programs have been written that may help you develop such skills as formation management and sight resolving. Sd is not suitable for this.

Quality and Correctness

Quality, correctness, and reliability are fundamental and extremely important design objectives of Sd and Sdtty. Before any version is released, it passes very rigorous diagnostic tests. These tests include verification of large amounts of C4 material from recent National Advanced and Challenge Conventions, and other C4 weekends, with particular emphasis on material written by competing programs. The goal is to make a program that never makes mistakes.

Authenticity of Calls and Concepts

We have endeavored to use the current Callerlab lists and definitions as the source for call and concept names and definitions wherever possible. Where this is not possible, either because Callerlab does not publish lists or definitions at all levels, or because a definition is unclear in some area, we have used other well-respected encyclopedias, along with our best attempts to make things clear and sensible. Particularly at extremely high challenge levels, where there are no standardization bodies to rein in callers' tendencies to change the definitions or usage of calls, concepts, and fundamental assumptions, it is not always possible to do this to everyone's satisfaction.

Spelling variations that would be insignificant in normal calling become a serious problem for computer programs. We have attempted to use the spelling in official lists where possible, but sometimes even these lists are careless. In such cases, we have attempted to correct the errors where possible, using a variety of sources. Priority is given to those sources that have shown the greatest care in their editing.

The `toggle ignoreblanks` command, or the "-ignoreblanks" command line switch, or the line "ignoreblanks" in the "options" section of the initialization file, will make Sdtty accept words that are run together or the use of blanks instead of hyphens. Hence, misspelled concepts and calls such as `tandem based triangles` (the Callerlab spelling is with a hyphen) and `inroll circulate` will be accepted.

Variations of the Program

Several user interfaces are available. They come in two general types, and the program name is different for these types.

Sd is the name of the program that uses menus and a *graphical user interface*. Calls are selected by clicking with the mouse on the chosen menu item. The principal version of Sd runs on Unix-like systems with the X Window System interface. Sd is also available for Domain/OS on Apollo systems. It is not yet available for PC compatibles.

Sdtty is the name of the program that uses a character-oriented keyboard interface. Calls are selected by typing their names. Sdtty runs on Unix-like systems and under DOS, Windows 3.1, and Windows 95 on PC compatibles. PC users may take comfort from the fact that most serious callers prefer Sdtty rather than Sd.

Documentation

If you received the program in the form of source files, the file '`INSTALL`' contains instructions for building and installing Sd and for formatting and printing its documentation. If you have received the program in pre-built form, you shouldn't need to read it.

The files '`sd_doc.txt`' and '`sd_doc.ps`' are this manual, in plain text and formatted form, respectively. The file '`db_doc.txt`' gives internal information on the database format, useful only to people editing the calls database.

See the file 'Relnotes' for information about what is new in this release.

Printed copies of the manual are available from the authors, or may be printed from the file 'sd_doc.ps'.

If you are on the Internet, you can read Sd documentation using the World Wide Web. Give the locator `http://www.x.org/misc/sd/` to your WWW browser program.

Getting Sd or Sdtty

You can obtain Sdtty for DOS on PC-compatible computers, on a 3.5-inch HD disk, with printed documentation. The price is $4 (price valid for 1996). See [Author Contact], page 7. If you want a different format instead, please say so.

Sd and Sdtty are also available via anonymous FTP from the directory 'misc/sd/' on host ftp.x.org. FTP is the Internet file transfer protocol.

The file 'install.exe' is the Sdtty program for installation on PC's. It is the same file as 'INSTALL.EXE' that is furnished on a floppy disk.

If you are on the Internet, you can download the program using the World Wide Web. Give the locator `http://www.x.org/misc/sd/` to your WWW browser program.

Pre-built versions for various Unix-style systems are available in the same FTP directory; check to see if there is a binary for your platform. The file 'Manifest' contains a description of all the files in this directory.

You must transfer the files in FTP's binary mode.

If your mailer can interpret MIME messages, we can also send you Sd by email. MIME is a draft Internet standard for structured mail, and it includes support for mailing large binary files.

Installing Sdtty on a PC

To install the program from a floppy disk to a PC-compatible, you must be running in DOS. There are two ways to go from Windows to DOS. The simplest is to click the `MS-DOS prompt` icon.

In Windows 3.1, this is typically in the `main` program group. In Windows 95, this is typically in the `Programs` sub-menu of the `Start` menu. To return to Windows after starting DOS in this way, just type `exit` to DOS.

The other way to start DOS is to exit from Windows. To exit from Windows 3.1, double-click the button in the upper left corner of the Program Manager window. To exit from Windows 95, select `Shut Down` in the `Start` menu and select `Restart the computer in MS-DOS mode`. To return to Windows 3.1, type the `win` command to DOS. To return to Windows 95, just type `exit` to DOS.

Once in DOS do this if you are installing Sdtty for the first time:

```
MD C:\SD
```

If you are upgrading to a new version of Sdtty, you don't need to issue the above command.

Then insert the floppy disk and do this:

```
CD C:\SD
A:INSTALL
```

If you are upgrading to a new version, it will ask whether it is OK to overwrite existing files. Type "Y" in each case, assuming that you want to replace the existing version. It will not damage or delete any sequence transcript files or initialization files.

Once the installation is complete, you can run Sdtty under DOS or Windows. To run it from DOS, type

```
CD C:\SD
sdtty
```

You can also start Sdtty directly from Windows, once you have completed the steps in the next section.

You may safely delete any documentation files from your hard disk after installing the program.

Making Sdtty Start Under Windows

You can arrange for Windows to start Sdtty directly, without explicitly going to DOS. You only need to do the following steps once. After you have done them, upgrades to new versions of Sdtty will work without repeating this.

This assumes that Sdtty was installed, as per the instructions above, in the directory `C:\SD`. If you installed it elsewhere, you must use the `Properties` command of the explorer (in Windows 95) or the PIF editor (in Windows 3.1) to modify the Command Line or Program Filename given in `sdtty.pif`.

Under Windows 95

Click `Start`. Choose `Settings` and then `Taskbar`. Click on the tab called `Start Menu Programs`. Click on `Add`. Type the file name `c:\sd\sdtty.pif`, and then click `Next>`. When it says `Select folder to place shortcut in`, select the desired location, for example, `Desktop`, `Start Menu`, or `Programs`, and then click `Next>`. When it says `Select a name`, leave it as `SDTTY` and click `Finish`. Sdtty will now be in the selected location.

Under Windows 3.1

From the Program Manager, double-click the program group you want to place it in, for example, `Accessories`. In the Toolbar of the Program Manager, click `File`. Then click `new`. Select `Program Item`, and click `OK`. Type `C:\SD\SDTTY.PIF` as the `Program Filename` or `Command Line`. Choose an icon if you wish. Then click `OK`.

Licensing Terms

Sd and Sdtty are "free" programs. In short, this means that it is perfectly legal and ethical to copy and redistribute the program and documentation. For example, you may obtain the program by having a friend copy the disk and manual for you. (Or just copy the disk and print your own manual from the file that is on the disk.)

Sd and Sdtty are sometimes also distributed in source form. When so distributed, they are licensed under the GNU General Public License, which is in the file '`COPYING`'. The terms of this license, for distribution of source files or programs made from modified source files, are very strict. Programs constructed from modified source files may never be duplicated or redistributed by anyone, even in executable form, without obeying the conditions of the license. Read the license carefully.

Note in particular that any modified program based on Sd or Sdtty must obey the licensing terms every time it is redistributed by anyone. This requires redistribution of all of the source files, or otherwise complying with section 3 of the license.

This document is distributed in both formatted and plain-text form, in the files 'sd_doc.ps' and 'sd_doc.txt', respectively. Some other documents are also distributed with the program. Permission is given to distribute verbatim copies of these documents, either by photocopier or by computer, but modification of them is forbidden by copyright law.

How to Contact the Authors

The Internet mailing list sd, hosted at x.org, is for discussion of Sd and announcements of new versions. Send email to sd-request@x.org to be added.

Please send any praise, bug reports, or other comments on Sd to the authors at the addresses given below. Be sure to include the version number.

William Ackerman
wba@apollo.hp.com

Stephen Gildea
gildea@x.org

If you do not have access to electronic mail, write to

Bill Ackerman
100 Parlmont Park
North Billerica MA 01862-2722

Acknowledgements

Thanks to Kathy Godfrey, Sue Curtis, Chris Stacy, and Lynette Bellini for their help, encouragement, suggestions, contributions of calls, feedback, testing, and many other contributions.

The non-windowed interface for `sdtty` (see Chapter 8 [Terminal Interface], page 44) was written by Alan Snyder.

We received helpful feedback and testing from Robert French, Judy Anderson, Bill van Melle, Lois Lew, Ron Nicholson, Dave Decot, and Kristin Jensen.

1 Getting Started

If you are using a PC and have installed `Sdtty` to run under Windows 3.1 or Windows 95, start the program in the usual way by clicking its icon. `Sdtty` is actually a DOS program, so, once it is started, it will not use the mouse.

The first thing it will do (assuming you are not using the session control feature of the initialization file) is ask you for the level. Type the desired level, and press ⟨ENTER⟩.

`Sd` or `Sdtty` can be run from a shell (on Unix or DOS) by typing its name and the level you want to write sequences for, e.g.,

```
sd c2
```

or

```
sdtty c2
```

In addition to calls and concepts, there are program commands for actions such as automatic resolves, exiting the program, and other miscellaneous actions. See Chapter 6 [Resolving and Searching], page 36. In `Sd`, they are found in the menu in the upper left corner. In `Sdtty`, just type the name of the command.

The following chapters cover the operation of `Sd` and `Sdtty` in detail.

2 The Completing Reader for Sdtty

`Sdtty` uses a *completing reader* for interpreting the characters that you type. This means that you only need to type as much of the word or command as is required to be unambiguous. Whenever you type a space, `Sdtty` completes the word you just typed, if necessary and possible. Whenever you type ⟨ENTER⟩ (or ⟨RETURN⟩ on some keyboards), `Sdtty` completes the entire line if necessary and possible.

For example, at mainstream, you can enter a swing thru by typing '`swing thru` ⟨ENTER⟩'. When the space after '`swing`' is typed, no completion is required. When the ⟨ENTER⟩ after '`thru`' is typed, no completion is required either. If you type '`sw thru` ⟨ENTER⟩' instead, `Sdtty` completes the word '`swing`'. It does this by printing '`ing`' on the screen as soon as you type the space after '`sw`'. If you type '`sw th` ⟨ENTER⟩', `Sdtty` will complete both words. It prints '`ing`' as soon as you type the space, and it prints '`ru`' as soon as you type ⟨ENTER⟩. In fact, you could just type '`sw` ⟨ENTER⟩'. `Sdtty` completes the entire line, if necessary and possible, when return is typed. So it will print '`ing thru`' after your '`sw`'. In each case, the full phrase '`swing thru`' will appear on the screen, and will be printed in the final sequence.

Because of this completion, you can get into the habit of just typing '`sw` ⟨ENTER⟩' at mainstream. However, it will only work if what you type is unambiguous. At higher levels, '`sw`' could also mean '`swap around`'. When there is an ambiguity, `Sdtty` will beep at you and complete as much as it can. You can type '`?`' or '`!`' (discussed below) to see what the problem is.

With practice, you can get a reasonable feel for how much abbreviation you can get away with in a given context.

Typing ⟨ESC⟩ (or ⟨ALT MODE⟩ on some keyboards) or ⟨TAB⟩ will complete the entire line, just like ⟨ENTER⟩, but will not actually process the line. If you like what you see, you can then press ⟨ENTER⟩ to execute the call. If the line is ambiguous, the program will print as much of the line as it can. When used effectively, this can save you a lot of typing.

`Sdtty`'s completing reader is completely indifferent to the capitalization of what you type. It capitalizes its output according to its own notions of aesthetics and ambiguity avoidance.

2.1 Question Mark and Exclamation Point

At any time, you can type a question mark or exclamation point. Either of these will make `Sdtty` print out all legal completions of the line that has been entered so far. If what you have typed is unambiguous, that will consist of just one thing. If you have typed nothing, it will consist of every legal thing you could type—typically that is every call, every concept, and every special command.

The difference between question mark and exclamation point is that the question mark actually attempts to execute every call in the current call menu (that is, every call whose name matches the text you have typed so far), and shows only those that can be legally performed. Exclamation point simply prints every call on the menu that matches the text that you have typed so far. Remember that the call menus are only approximate—they often have many calls listed that are in fact not legal in the present setup. This is particularly true when concepts are in place. The program simply has no idea what calls are legal without trying them. Question mark tells it to do so.

The difference between question mark and exclamation point only applies to calls. All concepts and special commands (`resolve`, for example) are always listed whenever either character is typed.

The output from typing a question mark or exclamation point may be quite lengthy. If it fills more than one screenful, `Sdtty` will stop and display '`--More--`' at the bottom of the screen. Type ⟨SPACE⟩ to go on to the next screenful. Type ⟨ENTER⟩ to see just one more line. Type a backspace (or ⟨DEL⟩) to stop. When the output ends, either because `Sdtty` printed everything or because you typed a backspace, the partially entered line will be redisplayed, just as it was before you typed the question mark.

When you are typing calls or concepts and you type ⟨ENTER⟩ while the line that you have typed is ambiguous, `Sdtty` will display something like '`(10 matches, type ! or ? for list)`'. The number that it prints is the number of syntactically legal choices among which it can't decide. This is the number of choices that would have been listed if you had typed '`!`'. It may be more than the number of things that would be listed if you had typed '`?`'. The reason is that the '`?`' operation has to test every call before listing it. It can easily do this faster than you can read the list of choices. When you type ⟨ENTER⟩ while there are multiple syntactic choices available, the computer can't count them fast enough to respond instantaneously. It therefore displays an approximate number, computed by looking at its database without testing each call.

2.2 Minimizing the Amount of Typing You Must Do

The `Sdtty` completion mechanism, especially the ⟨ESC⟩ (or ⟨ALT MODE⟩ or ⟨TAB⟩) key, can save you a lot of typing. For example, typing

```
heads do your part ⟨ENTER⟩
```

is probably more than you want to do. If you type just

```
heads do
```

the line will be almost unambiguous—there are only two possible commands that start this way. By typing a question mark, `Sdtty` will show them to you. They are

```
heads do your part
```

and

```
heads do your part (while the others)
```

What these concepts actually mean will be discussed at length in Section 12.10 [Designating Certain People], page 79. For now, we are just discussing how to type them.

These commands don't differ until the '`t`' in '`part`'. If you type ⟨ENTER⟩ after '`part`', `Sdtty` will know that you want the first of these commands. If you type a space and then ⟨ENTER⟩, `Sdtty` will know that you want the second. Does this mean that you must type the whole command up to '`part`', followed by a space or ⟨ENTER⟩? No. Just type '`heads do`', and then press ⟨ESC⟩. `Sdtty` will display '`heads do your part`', since that is the text that is common to both commands. At that point, you may press ⟨ENTER⟩ to get the '`heads do your part`' concept. Or you could type space and then ⟨ENTER⟩ to get the '`heads do your part (while the others)`' concept. You could, of course, type the complete phrase '`(while the others)`' after the space, but there is no need to. Once you type the space, `Sdtty` knows exactly which command you want.

What would happen if you tried to shorten the command still further by typing just '`heads d`' instead of '`heads do`'? Below C2 this would work just fine. At C2 and above, an ambiguity would exist because of the '`heads disconnected`' concept. Typing a question mark after '`heads d`' will show you the ambiguity. In general, typing a question mark will help guide you in minimizing the amount of typing you must do. With practice, you will be able to enter your favorite calls and concepts with minimal typing.

Here is another example. To enter '`criss cross the shadow`' or '`criss cross your neighbor`', it is not necessary to type '`criss cross t`' or '`criss cross y`'. You can just type '`cri`' followed by ⟨ESC⟩. The program will print everything up to the point of ambiguity, which, at C2, is '`criss cross `'. At that point, you just need to type '`t`' or '`y`', followed by ⟨ENTER⟩. Above C2, there are more possible calls, but typing '`cri`' followed by ⟨ESC⟩ is still a good way to proceed.

2.3 Ignoring Blanks and Hyphens when Typing

`Sdtty` has an option whereby it can recognize calls and concepts that have been typed in without the spaces that normally separate words. Some users prefer abbreviated spellings like `inroll circulate`—when this switch is given, you can spell it that way. You can even type `tandemreverserandomstablespinthetop` if you wish. This also allows one to ignore hyphens, as in `tandembased triangles` or to type blanks instead of hyphens, as in `tandem based triangles`. In all cases the correct spelling will be printed in the transcript: `tandem-based triangles` or `in roll circulate`.

The `toggle ignoreblanks` command toggles (turns on or off) the state of this option. See Section 7.7 [Changing Modes], page 43. It can also be turned on when the program begins, either by giving a command-line option (See Section 3.1 [Command-Line Options], page 14) or through the use of the initialization file (See Section 4.2 [Option Control], page 21).

3 Invoking Sd via Command Line (DOS or UNIX only)

sd level [Xt options ...] [Sd options ...]
sd level -write_list filename
sd level -write_full_list filename

The program is normally invoked with a single argument—the level. This is one of the following: m (mainstream), p (plus), a1, a2, c1, c2, c3a, c3, c3x, c4a, or c4. The level can also be determined by the selection made when using the "session" feature. If the level is not given, or the program was started by a mouse click from Windows, the program will ask you for it. However it is specified, the level determines the calls and concepts that will be made available, according to our best guess of what the levels mean. Various optional arguments are permitted to control the window system, customize the list of calls used, and set other options. These will be described later.

The call definitions will be read in from the encoded database file 'sd_calls.dat'. The program will then ponder the database for a few seconds while it determines what calls to put on what menus. Depending on the speed of your computer and the level you selected, this could take from a few seconds to over a minute.

3.1 Command-Line Options (DOS or UNIX only)

There are a number of option keywords that can be given to Sd or Sdtty when the program is started. These control various aspects of the program's operation. These keywords are placed on the command line with which the program was invoked from the Unix shell or the DOS prompt. Each keyword has a hyphen in front of it.

Examples:

```
sd -singlespace plus
sdtty -retain_after_error -active_phantoms c3
```

These option keywords can also be placed, without the preceding hyphens, in the "options" section of the initialization file. See Section 4.2 [Option Control], page 21. When this is done, the options will always be in force, and do not need to be specified each time the program is run.

The following options are used by both Sd and Sdtty. Note that some of them use the underscore character, whereas the corresponding "toggle" commands use blanks between words.

`-db` *filename*
: location of the calls database. The default is '`sd_calls.dat`' in the current directory.

`-sequence` *filename*
: base name of the file to write sequences to. The calling level will be appended to this name. The default is '`sequence`', so a complete file name might be '`sequence.C1`'.

`-singlespace`
: This makes the program write out the output file with single spacing. This may be useful for those who use special fonts or special printer arrangements.

`-ignoreblanks`
: This makes the program recognize calls and concepts that have been typed in without the spaces that normally separate words. In particular, it makes the program behave as though you had given the `toggle ignoreblanks` command. See Section 2.3 [Ignoring Blanks], page 13.

`-retain_after_error`
: This makes the program automatically retain all concepts that had been typed prior to the current call, whenever an error occurs. The usual behavior is to discard all typed concepts when an error occurs. In particular, it makes the program behave as though you had given the `toggle retain after error` command. See Section 5.5 [Retaining Concepts After an Error], page 34.

`-discard_after_error`
: This makes the program discard all concepts that had been typed prior to the current call, whenever an error occurs. Since this is the usual behavior, you never need this option.

`-active_phantoms`
: This turns on the "active phantoms" behavior, as though the command `toggle active phantoms` had been given.

There are X resources associated with both options: `Sd.sequenceFile` is equivalent to the `-sequence` switch, and `Sd.databaseFile` is equivalent to `-db`.

If you almost always will be passing the the same value for a command-line switch, you may find it more convenient to set the corresponding resource.

With the X Window System interface, Sd accepts all Xt command-line options. The following are some of the more useful options for use with Sd.

`-rv`
: reverse video

`-bg` *color* background color

`-fg` *color* foreground color

`-bd` *color* border color

`-font` *font*
: font used everywhere

`-geometry` *geom-spec*
: geometry in pixels

`-title` *string*
: title for window manager and icon use

`-name` *string*
: name to look up resources under. default: program name.

`-xrm` *resource-line*
: an X resource manager string.

These options are specific to the terminal-based version, `Sdtty`.

`-no_cursor`
: dumb terminal mode: do not use cursor motion commands to keep the screen up-to-date efficiently. Use this if the cursor motion and screen manipulation operations of your terminal or operating system are not available or are not working satisfactorily.

`-lines` *nlines*
: the number of lines on the screen. Default is 25.

`-no_line_delete`
: do not use the insert line or delete line cursor control commands. Use this if these commands are not working satisfactorily on your terminal.

`-no_graphics`
: Suppress the use of special PC graphics characters when drawing pictures on the screen. These characters are not standard on all PC's. Pictures drawn in the output file will never use the special graphics characters.

3.1.1 Abridgement

The program recognizes three flags which allow use of an abridged list of calls:

`-write_list` *filename*
: write out the call list for the indicated level and exit

`-write_full_list` *filename*
: write out the call list for the indicated level and all lower levels and then exit the program.

`-abridge` *filename*
: read in the file, strike all the calls contained therein off the menus, and proceed.

The first two are used to prepare a call list. The call list for the indicated level, exactly as the calls appear in the menu, will be written to the named file. If `-write_list` is used, only the calls exactly on that level will be written. If `-write_full_list` is used, the lower level calls will be written as well, so the file will look exactly like the main call menu. After performing either of these operations, the program exits.

The third special flag is used to read in a list of calls to be avoided. Any call listed in the file, in precisely the same format as it was written out, will be removed from the internal database prior to running the program. Every sequence written under control of such a file will say '`(abridged)`' on its header line.

To write sequences for a group that is learning C2, for example, run the program with

```
sd -write_list my_group c2
```

Then delete from the file '`my_group`' those calls that the group has learned, i.e., those calls not to be avoided. When writing sequences, use the command

```
sd -abridge my_group c2
```

As the group learns new C2 calls, delete the corresponding lines from the file '`my_group`'. That file always contains the calls that they don't yet know. When the file goes to zero, they know the whole list.

Be aware that the abridgement mechanism works only for calls, not for concepts. You must keep track of what concepts not to use.

The lines in the abridgement file must always be in exactly the same format as the strings that are written out by the `-write_list` or `-write_full_list` flags. The program has no tolerance for creative capitalization, stray blanks, or other variations. Any line in the file that does not match a call in the menu is simply ignored. The order of the lines is not important.

4 The Initialization File

When Sd or Sdtty starts, it looks for a file 'sd.ini'. This file, if present, contains information about dances you might be working on, and about possible individual preferences you might have for the way the program runs. Such a file might look like this:

```
[Options]
discard_after_error
singlespace

[Sessions]
sequence.C3       C3          12      NACC, June 1995
sequence.C4       C4          31      NACC, June 1995
workshop          C1          75      My wednesday group
+                 A2          9       NESRDC/1995
*                 A1          2       Lake Shore Farm Weekend
```

4.1 Session Control

Each line after the line [Sessions], up until a blank line or the end of the file, describes a possible session of using the program, showing the output filename, level, next sequence number, and title.

This example file shows 5 possible things you could work on. You can select any one of them during a session. When the program is started, it will display the 5 lines and ask you which one you want to use, like this:

```
Do you want to use one of the following sessions?

  0     (no session)
  1  sequence.C3       C3          12      NACC, June 1995
  2  sequence.C4       C4          31      NACC, June 1995
  3  workshop          C1          75      My wednesday group
  4  +                 A2          9       NESRDC/1995
  5  *                 A1          2       Lake Shore Farm Weekend
  6     (create a new session)

Enter the number of the desired session:
```

If you enter "0", or just press ⟨ENTER⟩, the program will ignore the sessions and proceed normally. If you enter the number of one of the lines of the file, the program will operate at that level, and use the named output file and title. It will also print a sequence number on each card, starting with the number shown. For example, if you entered "3" after the file above is displayed, the program will operate at C1. The output file will be "workshop" instead of the default "sequence.C1". Cards will be serialized starting at 75. Each card will have a title saying "My Wednesday group".

If the given file name is "`+`", the program will generate a file name for you, based on today's date. This may be useful for managing files, so that you know when it is safe to delete a file that you have printed. The program will of course tell you what the name of that file is. All runs of the program using that session on the same day will append their sequences to the same file. The file name will be something like like'`9mar96.C3`'.

If the given file name is "*", the program will generate a file name in the same way, except that it will always generate a new file for each run of the program. If the program is started several times in one day, the files will be different. The generated file names will be something like '`9mar96a.C3`'. Note the '`a`' after the year.

At the end of the session, the file will be updated with the next index number. For example, if you wrote 12 sequences, they would be numbered 75 through 86, and the file would be rewritten as

```
workshop          C1           87     My wednesday group
```

so that the next session for that group would start with sequence number 87.

If, at any time during the session, you changed the output file (with the `Change Output File` operation) or the title (with the `Change Title` operation), the initialization file will be updated at the end of the session to show the effect of the change.

When the program starts, you can tell it to create a new entry in the file by entering the number corresponding to `(create a new session)`. In the above example, type "6". The program will ask you for the level and title, and will set the output file to the standard "sequence.C1" or whatever. You may use the `Change Output File` command and the `Change Title` command to change these later if you wish. At the end of the session, item number 6, containing the appropriate data, will be written to the updated initialization file.

If you no longer want some entry in the initialization file, you can delete it. Type the negative of its line number. In the above example, if you entered "-4", the line

```
+                 A2           9      NESRDC/1995
```

would be deleted. After doing this, the program terminates immediately. Run it again to use the new initialization file.

Whenever the program updates the initialization file 'sd.ini' at the end of a session, it saves the old contents in 'sd2.ini'. By copying that back to 'sd.ini', you can restore it.

You can also edit the file with an editor. Simple editors, such as 'notepad' are preferable to sophisticated word processors when editing this file, since word processors often insert specialized control information into the file.

4.2 Option Control

Each line after the line `[Options]`, up until a blank line or the end of the file, contains a command line switch. When the program is started, it will behave as though those switches had been given on the command line. This makes it possible to encode your personal preferences so that the program will always use them. Do *not* place a hyphen in front of an option in the initialization file.

For example, the line

```
singlespace
```

would be used if you like your output single-spaced.

The line

```
no_graphics
```

would be used if your computer does not have the special graphics characters.

The line

```
ignoreblanks
```

would be used if you wish to type calls and concepts without the normal blanks separating them, or without hyphens. See Section 2.3 [Ignoring Blanks], page 13.

The line

```
retain_after_error
```

would be used if you wish to have the program retain the concepts that you had typed when an error occurs. See Section 5.5 [Retaining Concepts After an Error], page 34.

You can edit the file with an editor such as '`notepad`'.

4.3 Sample Initialization Files

When `Sd` or `Sdtty` is installed, no initialization file exists, but the files '`sample1.ini`' and '`sample2.ini`' are provided for you in the working directory. The '`sample1.ini`' file shows an example of the "session" mechanism only, and '`sample2.ini`' show both mechanisms. The '`sample2.ini`' file is the same as the example shown previously. You can copy either file to '`sd.ini`' and edit it if you wish.

5 Calling

This chapter describes how to use the program in its normal mode to generate a sequence of calls of your choosing.

5.1 The Startup Menu

The startup menu is displayed when the program starts and after each sequence has been completed and written to a file. In addition to the program exit choice, this permits you to select the starting operation. The selections here, and almost everywhere else in the program, are made in `Sd` by moving the mouse until the cursor is over the chosen item, and pressing Mouse Button 1 (typically the left button). In `Sdtty`, selections are made by typing them. To see the “menu”, that is, the list of things you could type, type a question mark.

5.1.1 Heads/Sides Start

If you click or type `heads start` or `sides start`, the sequence will begin with ‘`heads`’ or ‘`sides`’ and the first call. Normally this means that the designated people will move into the center and do the first call while the others wait. Subsequent calls will be directed to everyone. For example, one might select `heads start`, and then `star thru`, and then `double pass thru`, and so on. You can also select `heads start` or `sides start` followed by a call such as *split square thru* or *split dixie style to a wave*. Use the `split` concept to get these.

When using `Sdtty`, you must press ⟨ENTER⟩ after typing `heads start` or `sides start`.

If you want to have the heads go into the middle and do several calls before the sides join in, as in *head ladies chain ; heads square thru 4*, or *heads touch 1/4 ; walk and dodge*, you must select `centers` before all calls after the initial one. Typically, you won’t need to read the card that way, but can rely on precise phrasing and inflection of your voice to express what you want.

5.1.2 1P2P Lines

The `heads 1P2P` and `sides 1P2P` starting actions refer to caller jargon for a common starting maneuver, typified by <*ANYONE*> *lead right and circle up 4*. You can read the card that way, or as *wheel thru and circle up 4*, or as *step right*, or as *bring us together*, or whatever improvisation you like. The promenade distance printed at the end of the sequence will assume that you said *step right* or *lead right* (or *wheel thru*) *and circle up 4*.

Whether you consider 1P2P openings to be overused is up to you.

5.1.3 Just As They Are

The `just as they are` selection starts the sequence on squared set spots, without having the heads or sides begin. This is generally useful only if the *all 4 couples* concept is to be used.

5.2 Entering Calls

The usual thing you do when writing a sequence is to select calls from the call menu, perhaps preceded by one or more concepts. (Concepts are discussed in the next section.) In `Sd`, a call is selected by clicking on the desired menu item. In `Sdtty`, a call is selected by typing its name, followed by ⟨ENTER⟩. The completing reader (Chapter 2 [The Completing Reader for Sdtty], page 10) lets you abbreviate calls.

You should be aware that a number of *modifier* words like `cross`, `left`, and `grand` are treated as *concepts* by `Sd` and `Sdtty`. While concepts don't officially exist below the Advanced program, `Sd` and `Sdtty` consider words such as these to be concepts. When using `Sdtty`, this generally makes no difference. Because of the way the program operates, you can just type the call, with whatever concepts or modifier words it has, in the natural way. For example, you could type:

> *`grand swing thru`* ⟨ENTER⟩

or:

> *`centers cross run`* ⟨ENTER⟩

In a few cases, either because a concept has a complex structure (e.g., `checkpoint`), or because an utterance is ambiguous (e.g., doing 1/2 of a *stable swing thru* vs. doing a *1/2 stable swing* *thru*), it is necessary to type concepts by themselves, pressing ⟨ENTER⟩ after each one. This will be discussed further in Section 5.3 [Entering Concepts], page 30.

When using `Sd`, all concepts, modifier words, and calls are listed separately in the menus. For example, to get a *grand swing thru*, click on `grand` in the concept menu, and then `swing thru` in the call menu. There is no item `grand swing thru` in the call menu. The menus would be too unwieldy if such things were listed separately. To do 1/2 of a *stable swing thru*, click on the `1/2` concept (actually the `<N>/<N>` concept), followed by the `stable` concept. To do a *1/2 stable swing thru*, click on the `1/2 stable` (that is, `<N>/4 stable`) concept.

5.2.1 Call Variations

There are a number of issues that make call entry less than completely straightforward. These have to do with variations that calls have—directions (as in `quarter left`), numbers (as in `square thru 3` or `three quarter thru`), person designators (as in `sides kickoff`), subcalls (as in `clover and quarter thru` or `vertical tag your neighbor` or `shuttle rally` or `in roll motivate`), and modifiers (as in `strut left` or `trans cross chain reaction`).

To list all possible variations in the menus would be unwieldy for a variety of reasons. Because of this, various methods are used to shorten the menus and to allow the user to specify the desired variation. These methods will be discussed in detail presently, but in most cases what happens is the following:

In `Sd`, the menus list only the base calls with special keywords as in `quarter <DIRECTION>` or `square thru <N>`. When you click on such a menu item, a popup menu appears listing the possible choices.

In `Sdtty`, the *menu* (that is, what you see if you type a question mark) also has keywords. You can type the item just as shown, e.g., `quarter <DIRECTION>` or `square thru <N>` (you would actually type the angle brackets) and then answer the question that `Sdtty` asks. But there is an easier way—you can type what you want directly. That is, just type `quarter left` or `square thru 3`. If you want to see the choices, type `quarter <DIRECTION>` or whatever, and then type a question mark.

Directions

Calls that take a direction have the keyword `<DIRECTION>` in them in the menus, like `pass <DIRECTION>` or `spin a windmill, outsides <DIRECTION>`.

When using `Sd`, after you click on such a call, a popup will appear listing all possible directions. Click on the one you want. If you decide you don't want that call, just move the mouse away from the popup.

When using `Sdtty`, you can either type `pass <DIRECTION>` literally, and then answer the question that the program will ask, or just type the actual call, for example, `pass in`.

The menu lists all possible directions, including some that may not be possible in the given context.

Numbers

Calls that take a number have the keyword `<N>`, `<N/4>`, or `<Nth>` in them, like `eight chain <N>`, `invert the column <N/4>`, or `square thru, but on the <Nth> hand <ANYTHING>`.

When using `Sd`, a popup will appear listing the numbers from 1 to 8. Click on the one you want. If the keyword was `<N/4>`, click on the value of `N`. The program will turn it into `N/4` (or, if `N`=2, into `1/2`). If you decide you don't want that call, just move the mouse away from the popup.

When using `Sdtty`, you can either type the keyword literally, and then answer the question, or just type the number just as you want it to appear in the call. That is, you can type `invert the column <N/4>` and then `3`, or `invert the column 3/4`. You can type `1/2` or `2/4`—the program will print it as `1/2` in either case. For `square thru but on the <Nth> hand`, you can type the number in `Nth` form, as in `square thru but on the 3rd hand`. Of course, if you just type `3`, the completing reader will finish it for you.

Not all values of `N` will be meaningful for all calls. For example, you can `eight chain 5`, but you can't `invert the column 5/4`.

Some calls have the words `quarter`, `half`, or `three quarter` in their names. Examples are `quarter thru`, `quarter the deucey`, `quarter mix`, `quarter the alter`, and `quarter chain and circulate in`. These are listed in the Callerlab lists with the words spelled out, though individual preferences vary. Some people prefer to write `1/4 thru` and `3/4 thru`. `Sd` and `Sdtty` list the call as `<N/4> thru` in the menu. In `Sd`, you click on that and then click on a number (only 1 and 3 are legal, of course). In `Sdtty`, you can type `<N/4> thru` and then type 1 or 3, or you can type the call directly, either in numbers or in words. That is, you can type either `1/4 thru` or `quarter thru`. No matter how you enter one of these calls, it will always be printed out in words, as in `three quarter thru` or `half chain and circulate in`.

Person designators

These work like the others in a straightforward way. The calls appear in the menu with the `<ANYONE>` keyword, as in `patch the <ANYONE>`. In `Sd`, you click on that and then make a selection from a popup menu. In `Sdtty`, you can either type the `<ANYONE>` literally and then answer the question, or you can type in the desired call directly, as in `patch the sides`.

The gender designators are `boys` and `girls`. This is the way they appear in the Callerlab mainstream list. The author does not take a position on whether the adult terms would be more reasonable words to use in any given context. You must use your own judgement in deciding what to say.

Not all designators are legal in all cases; for example, you can't call `leads run` from a grand wave.

The program specifically refuses to recognize the meaning of the designators `centers` and `ends` while in a 1x8 setup (e.g., a grand wave). This is because these terms can be ambiguous in such a setup. If you want the centers of each 1x4, you must use the `EACH 1X4`, `EACH LINE`, or `EACH WAVE` concept in order to make the `centers` or `ends` designators work. Of course, in many cases, other designators, such as `boys` or `girls` can identify the same people unambiguously. If you want to designate the 4 people in the center of the set, use `center 4`. `Outer pairs` specifies the others.

Tagging Calls

Calls that use tagging calls have the keyword `<ATC>` (for Any Tagging Call) in them, as in `<ATC> your neighbor` or `<ATC> chain thru and scatter reaction`. Specify these in the usual way. In `Sdtty` you can type the `<ATC>` directly or you can enter the complete call.

At C3 and above, the tagging calls include the calls `revert <ATC>` and `reflected <ATC>` that make use of another tagging call. In `Sd`, you will be presented with another popup. In `Sdtty`, you can type in the entire call directly, as in `revert cross flip chain thru reactivate`.

<ANYTHING> Calls

Calls that use the "<ANYTHING>" mechanism, such as `in roll motivate`, have the keyword `<ANYTHING>` in them, as in `<ANYTHING> motivate`. In `Sdtty`, you can type in the entire call directly. In `Sd`, click on the call with the word `<ANYTHING>` in it. A popup will appear, asking for the replacement for the circulate.

Only bona-fide circulate-type calls, and other well-recognized similar things such as `split counter percolate` can be obtained by this method. You can get more general substitutions by using the `simple modifications` command. Type (or click) that command, and then `motivate`. You will be asked for a substitute call. You can enter anything at this point, such as `swing the fractions`. The result will appear in the transcript with brackets, as in `[swing the fractions] motivate`.

Subcalls

Mandatory subcalls appear in the menu as the keyword `<ANYTHING>`, as in `clover and <ANYTHING>`. The handling of these in `Sdtty` is not as nice as for the keywords discussed previously. It is not possible to type in the call just the way you want it to appear. *You must literally type the <ANYTHING>*. That is, you type `clover and <ANYTHING>` or `slant <ANYTHING> and <ANYTHING>`. Of course, because of the completing reader, you rarely need to type that much. Just `clover a` is enough, for example. Furthermore, function key ⟨F8⟩ is equivalent to typing `<ANYTHING>`. In `Sd`, click on the menu item in the usual way. Either way, the program will prompt you for a subcall. This can be any legal call, including concepts and modifiers. Whatever you enter will appear in brackets, as in `clover and [TANDEM shazam]`. These mandatory subcalls, with the keyword `<ANYTHING>` in the menu, are just one of many types of modifications. See Section 5.4 [Call Modifications], page 33.

Modifiers such as cross, left, split, grand, single, magic, and interlocked

As discussed above, the program considers these to be actual concepts, even though they aren't really.

Some calls use these concepts in ways that have a tricky word order.

Whether the word order is tricky or not, you can always enter the concept followed by the call as separate items. For example, in `Sd`, you can click on `left` followed by `chase right`. In `Sdtty`, you can type in `left` and then `chase right`, on two separate lines.

In addition, `Sdtty` allows more natural text entry. You can type in `grand swing thru` or `chase left` directly.

Other examples of this phenomenon are

```
switch to an interlocked diamond
unwrap the magic diamonds
hang a left
scoot and cross ramble
trans cross reactivate
revert cross flip chain thru cross nuclear reaction
```

No matter how you entered it, the calls will be printed out with the words in the correct place.

The foregoing only applies to calls in which the word is an actual concept or modifier. Some calls happen to have the word `cross` as part of their names. Examples are: `cross and wheel`, `cross and turn`, `cross your neighbor`, and `crossfire`. Such calls appear in the menu just as they are spelled.

5.2.2 Call Menus

In `Sd`, the call menu is the long menu near the center of the screen. There are actually 17 menus that can appear here, each tailored to the current setup. For example, if the current setup is right-hand waves, a menu will appear that contains only those calls that the program believes might be legal from right-hand waves. This determination is approximate but conservative—appearance on the menu does not necessarily mean it is legal, though absence from the menu means that the program is fairly certain that it can't be legal. There is a *universal* menu that appears when the setup is not one of the common ones, or when anything complex is going on. This contains every call that is in the database for the chosen level.

The call menus are alphabetized in a way that ignores blanks and hyphens. Special keywords such as `<N>`, `<ANYONE>`, `<DIRECTION>`, or `<ANYTHING>` are listed before letters. Hence, `<ANYONE> run` and `<ATC> your neighbor` will be found near the beginning of the menu.

5.2.3 Scrolling (Sd only)

The call menu can be scrolled with the scroll bar to its left. Move the mouse into the scroll bar area. Clicking the left mouse button will scroll the window down over the menu by a percentage of the window's size that is equal to the percentage that the cursor is down from the top of the scroll bar. Clicking the right mouse button will scroll the window up by the same amount. Clicking the middle mouse button will move the window to a percentage down the menu that is equal to the percentage that the cursor is down from the top of the scroll bar. If the mouse is moved while the middle mouse button is held down, the menu will scroll in real time.

5.3 Entering Concepts

A call may be preceded by concepts, which modify the action of the call. These may be nested (*stacked*) to any reasonable depth.

In `Sdtty`, just type the concept you want. The completing reader will operate in the usual way. Nearly all concepts may be entered in a natural way, on the same line as the call that they affect. As discussed previously, this means that you can type things like

`grand swing thru` ⟨ENTER⟩

or

`boys 1/2 stable split the difference` ⟨ENTER⟩

or

`random tandem swing thru` ⟨ENTER⟩

Whether you typed concepts separately or together, the result is the same, and the output file will look the same. For example, the following are all legal and equivalent.

reverse random tandem swing thru ⟨ENTER⟩

reverse random ⟨ENTER⟩
tandem swing thru ⟨ENTER⟩

reverse ⟨ENTER⟩
random ⟨ENTER⟩
tandem ⟨ENTER⟩
swing thru ⟨ENTER⟩

In the output file, they will all be shown as `reverse random tandem swing thru`.

Occasionally some ambiguous situations can arise. What if you wanted the boys to do 1/2 of a *stable split the difference*? The phrase `boys 1/2 stable split the difference` is ambiguous. Whenever `Sdtty` is confronted with an ambiguity, it chooses the option that nests concepts least deeply. A single application of the *1/2 stable* concept is less deep (simpler) than an application of the *1/2* concept followed by an application of the *stable* concept, so it chooses the former. If you really want to tell the program to do the latter, you must make clear that you mean the *1/2* concept by itself. The way to do this is to enter each concept separately, pressing ⟨ENTER⟩ after each one. That is, type

boys ⟨ENTER⟩
1/2 ⟨ENTER⟩
stable ⟨ENTER⟩
split the difference ⟨ENTER⟩

When `Sdtty` resolves an ambiguity, it shows how it did so through the judicious use of commas. For example, the preceding operation will be printed as `boys, 1/2, stable, split the difference`, as opposed to `boys, 1/2 stable, split the difference`. Do not type the commas in.

Whenever you run into trouble typing concepts, try typing each one separately, pressing ⟨ENTER⟩ after each one.

Some concepts require two calls (e.g., `checkpoint` and `interlace`). *You must enter such concepts by themselves.* That is, you must press ⟨ENTER⟩ after typing any of these concepts. You can't type

```
checkpoint ah so by recycle ⟨ENTER⟩
```

or

```
boys trade (while the others) u-turn back ⟨ENTER⟩
```

You must type

```
checkpoint ⟨ENTER⟩
ah so ⟨ENTER⟩
recycle ⟨ENTER⟩
```

or

```
boys (while the others) ⟨ENTER⟩
trade ⟨ENTER⟩
u-turn back ⟨ENTER⟩
```

After choosing such a concept, enter the first call, preceded by whatever concepts apply to it. The program will then prompt you for the second call. A complex tree of concepts and calls can thus be constructed.

In `Sd`, click on the concept from a menu. In a perhaps hopeless attempt to keep the concept menus from becoming too unwieldy, concepts have been organized into a general concept menu and several submenus organized by topics. The main concept menu is in the lower left corner. It supposedly contains all the extremely common concepts. Buttons for the other concepts submenus appear above this menu. Clicking on one of these buttons brings up the submenu. If you don't see what you want on a concept submenu, move the mouse away. Some concepts appear in more than one submenu—it doesn't matter which one you use. With practice, you will know which submenu to use. All available concepts, organized by submenu, are listed in Chapter 12 [Concept Notes], page 56.

When you click on any concept, the universal call menu replaces whatever special call menu may have been presented, since the set of legal calls becomes highly unpredictable.

Some concepts require a numeric designator (e.g., `interrupt after the 3rd part`) or a people designator (e.g., `girls are stable`). The handling is the same as for calls that require these.

The program recognizes the level at which concepts are legal, but lets you override this if you wish. The `toggle concept levels` command toggles (turns on or off) the state of off-level concept permission. See Section 7.7 [Changing Modes], page 43 and Chapter 10 [Using Off-Level Concepts], page 49.

5.4 Call Modifications

Some calls can be modified in a natural way, such as '`catch fan the top 3`', '`busy 1/2 tag`', and '`vertical tag your neighbor`', or in unnatural ways, such as '`trade the diamond but replace the diamond circulate with explode the diamond`'. To form such a call, you must first enable call modifications by selecting `simple modifications` or `allow modifications` before choosing the call.

There are two levels of this feature. If you click on `simple modifications`, you get the simple version, which only prompts you for *natural* modifications. Natural modifications have been somewhat arbitrarily defined as those for which there is an accepted way of fitting the words in without using the phrase "but replace the <whatever> with <whatever>." If you click on `allow modifications`, you allow a potentially large number of modification possibilities, sometimes several in the same call.

In most cases, `simple modifications` is probably the right thing.

The special "circulate" modifications exemplified by '`in roll motivate`' can be entered by this method. Select `simple modifications`, select `motivate`, and replace the circulate. The result will show the replacement call in brackets and spelled out in full, as in `[in roll circulate] motivate`. However, there is a better way to get these particular types of modifications—select `<ANYTHING> motivate` and then select the replacement for the circulate. The result will look natural, for example, `in roll motivate`. In `Sdtty` you can type in such a call directly.

When either level of modifications is selected, the line at the top of the text transcript area indicates this. Also, the universal call menu is chosen whenever modifications are enabled, since the possibilities are unpredictable.

When you click on a call for which modifications are possible, a popup will appear saying something like `The box circulate can be replaced. Do you want to replace it?` If you want to replace the designated call, click in the active area of the popup; otherwise, move the mouse away. If you indicate that you want to replace the call, you will then be prompted for the replacement call. Enter it, along with any concepts. In some cases you may be asked repeatedly about various modifications. For example, in *motivate*, you can replace the initial *circulate*, and you can turn the star a different amount. You may select any or all of these modifications.

Note: When complex modifications are involved, the program may ask for them in an order that seems unnatural. This is because it asks in the order that it executes the call internally, which may not be the same as the order in which the words are spoken. Do not be alarmed. Observe the popup titles and prompts carefully. The final result should come out in the correct order.

For both these optional modifications and the mandatory subcalls discussed earlier (e.g., `clover and <anything>`), the program attempts to show unambiguously how everything is structured, by putting subcalls and their accompanying concepts in square brackets. It also attempts to put natural modifications in their natural place in the phrase. In complex cases, like '`[CHECKPOINT LEFT catch [SINGLE CONCENTRIC snake] 3 BY [2/3 recycle] the difference] cover up`' things may be quite difficult. How (and whether) you choose to read such a card is up to you.

5.5 Retaining Concepts After an Error

When an error occurs in the execution of a call, `Sd` has an option whereby it tries to leave in place any concepts that were associated with that call. That is, after printing the error message, the program will behave as though you had once again typed those concepts. The assumption here is that only the call was mistaken, and you really wanted the concepts and do not wish to have to enter them again. If you don't want those concepts, you can remove them one by one with the `undo` command. Or you can use the command `discard entered concepts` to get rid of all of them at once.

The `toggle retain after error` command toggles (turns on or off) the state of this option. See Section 7.7 [Changing Modes], page 43. It can also be turned on when the program begins, either by giving a command-line option (See Section 3.1 [Command-Line Options], page 14) or through the use of the initialization file (See Section 4.2 [Option Control], page 21).

5.6 Asymmetric Selectors

The people selector popup contains various asymmetric selectors such as `near line` and `far box`. You can use these selectors with the `<ANYONE> do your part` concept to call *near column pass thru*, etc. The resultant setup must be reasonable—shape-changers on one side of the set may lead to problems. Most calls have not been checked for robustness in the presence of asymmetrical setups. Use this with care! The resolver checks all eight people, so it should work. If you have not restored symmetry, resolves may be extraordinarily difficult to find.

6 Resolving and Searching

This chapter describes how to have the program find calls for you to accomplish various goals.

6.1 Resolving

Whenever the setup is in a resolved state, whether intentionally or accidentally, the program indicates that fact at the bottom of the transcript area. The program looks for *right and left grand*, *left allemande*, *promenade*, *single file promenade*, and *circle left/right* getouts from a variety of setups. If the sequence needs an *extend*, *slip the clutch*, *circulate*, *pass thru*, *trade by*, *cross by*, or *dixie grand* first, the program reports that too.

Whenever the '`resolve`' message appears, you can end the sequence and write it to a file by selecting `write this sequence`. In `Sd`, the button for this is in the group in the upper left portion of the screen. In `Sdtty`, simply type the command, with the usual completion mechanism.

There are at least three ways to resolve a sequence: you can wander into a resolve by accident, you can *sight resolve* the square, entering the calls that you want, or you can use the `resolve` command. This command adds the necessary calls to the sequence, exactly as if you had entered them.

When you select `resolve`, the program goes into a special mode in which it searches for resolutions, saves them, and lets you look through them and pick one that you like. A resolution is a sequence of up to three calls that leads to a resolved state. While the program is in resolve mode, the call menu is replaced by a special menu of options, along with information about the current resolution. That information tells how many resolutions are currently stored and which one is currently shown. When you select `resolve`, the program finds the first resolution and displays it, showing information '`1 out of 1`', which means that it has one resolution stored, and resolution number one is currently displayed. The transcript area shows the effect of that resolution.

You can select `find another` to search for another resolution and add it to its stored list. When the list has more than one resolution in it, selecting `previous` and `next` will move around in its list of previously found resolutions and show whichever one you want. In this way, you can search for a better resolution than the one you already have, but go back to the earlier one if no better one is forthcoming.

Selecting `accept current choice` will leave resolve mode, causing the current resolution to be added to the sequence exactly as though you had entered those calls manually.

Selecting `abort the search`, `exit the search`, `quit the search`, or `undo the search` will throw away all of the saved resolutions and leave resolve mode, but will not destroy the sequence. The sequence will be left just as it was before `resolve` was selected.

Selecting anything else, such as `write this sequence`, is equivalent to `accept` followed by whatever that action is. So, for example, you can select `write this sequence` as soon as you see a resolution that you like.

When using `Sdtty`, these special commands are typed into the completing reader in the usual way. To make it more convenient to enter them, the command names have been chosen to be unambiguous from just one or two letters. So, for example, `p` means `previous`, `n` means `next`, and `f` means `find another`.

If one or more concepts have been entered when you select `resolve`, the program will search only for resolutions whose first call starts with those concepts. So, for example, selecting `once removed` and then `resolve` might get you this resolution:

```
ONCE REMOVED reverse the pass
linear flow
right and left grand  (7/8 promenade)
```

The program searches for resolutions by using a random number generator to generate up to 5000 random sequences, occasionally inserting concepts. It biases the search in favor of short sequences (one call) rather than long ones (three calls) and against resolutions that require *all 8 circulate*, *pass thru*, or *trade by* at the end. If, after 5000 attempts, no resolution is found, the `find another` operation fails. (This tends to happen if you try to resolve out of an hourglass at Mainstream.) You can select `find another` again to make another 5000 attempts if you wish.

Remember that the `resolve` *operation by itself does not write the sequence to a file.* You will not be able to print a sequence until it has been written to a file. You must give the `write this sequence` command to write out the sequence.

6.2 Normalizing or Standardizing the Setup

These commands invoke an operation very similar to the resolver, except that they search for sequences of up to three calls that make the setup "nicer".

The `normalize` command reduces various matrices, such as 4x4, into an 8-person setup. It uses a variety of phantom-like concepts to do its work.

Be aware that the phantom-like concepts that this command uses are not the only ways to get out of large matrix setups. There are a number of calls, like *press*, *truck*, *loop*, *squeeze*, *Z axle*, and *finish a long trip* that are also useful. The `normalize` operation does not search for such things. It is only intended to search for things that might not be obvious.

The `standardize` command turns the setup into an 8-person setup in which people are facing in "reasonable" directions. That is, it gets out of T-bone setups. It uses both plain calls and calls with concepts.

If you have selected `toggle concept levels`, these commands may make use of concepts that are not legal at the chosen level. This is sometimes useful in emergencies.

6.3 Letting the program pick a random call

There are five commands in this family: `pick random call`, `pick simple call`, `pick concept call`, `pick level call`, and `pick 8 person level call`. They search for any legal single call. While this may sound like a fairly pointless operation, remember that you can use this while one or more concepts are already entered, in which case it will search for legal calls that involve those concepts. Hence, this operation is useful for finding clever uses of difficult concepts such as *checkpoint*, *interlace*, or *on your own*.

The command `pick random call` attempts to pick completely random things—sometimes calls, and sometimes calls with concepts. `Pick simple call` never uses concepts. `Pick concept call` always uses at least one concept with the call that it picks. `Pick level call` picks only calls that are at or near the specified calling level. (By "near" we mean that, if the calling level is C4, it will pick C4A or C4 calls. If the level is A2, it will pick A1 or A2 calls.) `Pick 8 person level call` is similar, but it picks only calls that involve all 8 people.

6.4 Creating a Specific Setup

These commands pick some call or calls that go into a specified setup. The commands are:

```
create any lines
create waves
create 2fl
create lines in
create lines out
create inverted lines
create 3x1 lines
create any columns
create columns
create magic columns
create dpt
create cdpt
create trade by
create 8 chain
create any 1/4 tag
create 1/4 tag
create 3/4 tag
create 1/4 line
create 3/4 line
create diamonds
create any tidal setup
create tidal wave
```

"Any lines" and "any columns" mean any 2x4 setup in which people are all in generalized lines or columns, respectively. "Any tidal setup" means any 1x8, no matter how people are facing. All the others are independent of handedness—for example, a two-faced line can be of either handedness.

6.5 The `Reconcile` operation

Selecting `reconcile` invokes an operation like resolving, but it puts the generated calls someplace other than at the end of the sequence. This is useful if you have a clever getout at the end of the sequence, and you want it to be the resolve, but people don't have their partners and corners. This lets you retroactively modify the sequence so that the clever getout will work.

Note first that this operation may only be invoked when the setup is left-handed two-faced lines, right-handed waves, or left-handed waves, in which case it assumes you want a *promenade*, *right and left grand*, or *left allemande* respectively. The program must know which getout type you want. If the setup is an 8-chain, it can't tell. In that case, either do a `touch` or a `left touch`, to tell the program that you really want a *right and left grand* or *allemande left*, respectively. Then, after the reconcile operation is complete, you can erase that extra call.

The reconcile operation is very similar in behavior to resolve, except that the program needs to know the insertion point. Rather than searching for the first resolve as soon as you enter the mode, the reconcile operation lets you set the insertion point before searching. Select `raise reconcile point` or `lower reconcile point` to set it. A dotted line will be displayed showing where in the sequence the generated calls will be placed. When this is in the right place, select `find another` to find the first reconcile. You can change the insertion point at any time.

Reconciles are extremely difficult to find—much harder than resolves. To avoid frustration, make the insertion point be at a place where the setup is very simple and a large number of calls are legal. We recommend making the insertion point be at a place where the setup is in waves. Remember that, if the insertion point is at an hourglass, the program has to find a random sequence of up to three calls that goes from an hourglass to another hourglass while miraculously performing the required permutation.

Avoid using reconcile when a gender-dependent or head/side-dependent call lies between the insertion point and the end of the sequence. The program checks every reconcile by re-executing all calls from the insertion point to the end and verifying that everything is exactly as it was except for the permutation of the people. For example, if the call '`heads kickoff`' occurs after the insertion point, and a potential reconcile changes heads and sides, it will not be offered. For gender-dependent calls the situation is a little better: if, at the end of the sequence, the boys are in the center and the girls on the end before doing the reconcile, you know that any inserted sequence will have to be gender-preserving anyway, so calls like '`star thru`' and '`boys kickoff`' will be okay.

7 Miscellaneous Commands

This section describes commands that are not concerned with generating sequences.

7.1 Saving the Sequence

At any time when the sequence is resolved (for example, after a successful use of the `resolve` operation), you can select `write this sequence`. (This command used to be called `end this sequence`.) This will append the current sequence to the current output file, and then go back to the startup menu. You will be given an opportunity to enter a line of text to be used as a subtitle, for example "very hard interlace" or "stupid biggie." The written sequence will also be annotated with the level, the current date and time, and the version numbers of the program and database.

Until you have done the `write this sequence` operation, the sequence is not written to disk and you cannot start writing a new sequence. Just resolving isn't enough.

7.2 Changing the Output File

You can change the name of the file to which the program writes its output by selecting the `Change output file` command. A popup will appear, prompting you for a new file name, or `Sdtty` will ask you to type in the new name. This action will become effective the next time a sequence is written out. Sequences previously written will stay under their old name.

When the initialization file is used, any change in the output file name will be written back to that file at the end of the session.

7.3 Changing the Title

You can change the title that will be printed at the top of each sequence by selecting the `Change Title` command. A popup will appear, prompting you for a new title, or `Sdtty` will ask you to type in the new title. This action will become effective the next time a sequence is written out. Sequences previously written will keep their old title. You can remove the title by just pressing ⟨ENTER⟩ when asked for the new title.

When the program starts, there is no title, unless the initialization file was used.

When the initialization file is used, any change in the title will be written back to that file at the end of the session.

7.4 `Abort`, `Exit`, and `Undo`

If you select the `exit` command, the program will exit. If a sequence is in progress, the program will ask for confirmation first. (Remember that a sequence is not finished until the `write this sequence` command has been given.) In `Sd` with the X Window System, using your window manager to send a `Delete Window` message to the program is equivalent to clicking on the `exit` button. This is what normally happens when you double-click the special control button in the upper left corner of the `Sd` window.

If you select the `abort` command, the program will abort the sequence but not exit. It will go back to the startup menu to allow you to start another sequence. It will ask for confirmation first.

If you select the `undo` command while no call is partially entered, that is, while no concepts have been entered, the program will erase the last complete call, with all of its concepts. If one or more concepts have been entered, only the last concept will be erased.

7.5 Inserting Comments

You can insert arbitrary text strings into the sequence transcript in the form of *comments*. This is useful for things that you may want to say to help the dancers, such as "side boys be careful—go to your left." Also, if you want to call something that the program can't do, so you have to trick the program into doing it by means of several other calls, it is useful to enter a comment saying what you originally wanted. This can guide you when editing the transcript file.

To enter a comment, select `insert a comment`. In `Sd`, a popup will appear into which you may enter the text. In `Sdtty`, simply type the comment text on the next line. The comment will be enclosed in curly braces in the transcript.

A comment may be entered in front of any concept or call, and goes into the transcript exactly where it was entered. Because of the way the program assembles mouse-clicks into transcript lines, it is not possible to enter a comment at the end of a line.

7.6 Keeping Pictures

The program shows pictures on the screen for the current position and the position just before the last call. Normally, those pictures will not appear in the sequence written to a file. If you want the current position to have its picture written in a file (say, because the sequence is very difficult and you believe it may be necessary to say something helpful then like "check a parallelogram with boys in the center box") select the command `keep picture`.

7.7 Changing Modes

There are several commands that change some aspect of the way `Sd` or `Sdtty` runs. They are:

```
toggle concept levels
toggle active phantoms
toggle ignoreblanks
toggle retain after error
```

Normally, all of these are off. You may want to turn one or more of them on for your own personal preference, or for some special reason. These commands *toggle* their respective modes. That is, they turn the mode on if it was off, and off if it was on. The current state of the `concept levels` and `active phantoms` modes is always displayed in the input prompt in `Sdtty`, or the mode line in the upper right corner in `Sd`.

The `toggle concept levels` command changes the legality of concepts that are not on the specified calling level. See Section 5.3 [Entering Concepts], page 30.

The `toggle active phantoms` command changes the persistence of phantoms during certain calls. See Section 12.8 [Assume Waves], page 76.

The `toggle ignoreblanks` command changes the option whereby the parser lets type calls and concepts without the normal blanks separating them, or without hyphens. See Section 2.3 [Ignoring Blanks], page 13.

The `toggle retain after error` command changes the option whereby program retains the concepts that you had typed when an error occurs. See Section 5.5 [Retaining Concepts After an Error], page 34.

8 Terminal Interface

8.1 Sdtty

The program `Sdtty` uses the completing reader for call, concept, and command entry.

When typing, press *SPACE* to complete the current word. Type ⟨TAB⟩ or ⟨ESC⟩ to complete as much as possible. Type *C-u* (that is, Control U) to clear a partially-typed line.

`Sdtty` has a special command `refresh display` that has no counterpart in the window-based version. It causes a complete clean transcript of the current sequence to be printed. This can be used if the display got messed up due to such things as unreliable terminal or modem behavior.

`Sdtty` tries to keep the screen up-to-date efficiently, maintaining a clean transcript of the current sequence on the screen at all times, by using whatever Operating System facilities it thinks are appropriate. If, for any reason, this doesn't work properly for you, you can give the `-no_cursor` switch at program startup to disable that. This will cause `Sdtty` to treat the computer's display as if it were a dumb typewriter. When started in this way, the part of the transcript that has changed with each command is simply redisplayed. This will inevitably lead to a choppy appearance on the screen or printing device. The `refresh display` command cleans that up.

8.2 Function Keys in Sdtty

On the PC, `Sdtty` recognizes a number of function keys. These perform the following commands:

```
key     normal                    shift                    control

F1   heads start              sides start              just as they are
F2   two calls in succession  twice
F3   pick random call         pick concept call        pick simple call
F4   resolve                  reconcile                normalize
F5   refresh display          keep picture             insert a comment
F6   simple modifications     allow modifications
F7   toggle concept levels    toggle active phantoms
F8   <anything>               <concept>
F9   undo last call           abort the search
F10  write this sequence      change output file
F11  pick level call          pick 8 person level call
F12  find another             accept current choice    previous
```

All of these except '<anything>' and '<concept>' execute the command directly, that is, pressing the function key is equivalent to typing the indicated text and then pressing ⟨ENTER⟩. The keys for '<anything>' and '<concept>' are simply equivalent to entering that text without pressing ⟨ENTER⟩.

8.3 Sdtty on the PC

Sdtty uses the DPMI memory manager. If necessary, it uses its own copy of DPMI. It can be started under either standalone DOS or the 'MS-DOS Prompt' under Windows. It can also be started directly as a Windows application, even though it does not use the mouse while running. See [Making Sdtty Start Under Windows], page 6.

You need a 386 or higher processor and 2Mb of memory. A hard disk is recommended. There should be at least 175 kilobytes of memory free in the first 640Kb. The authors have not actually tried it on a system this small, and we would be interested in your experiences.

Sd and Sdtty are believed to be insensitive to slight inaccuracies in floating-point division.

To deal with different types of display hardware with various screen sizes, the -lines *n* switch can be given at program startup, as in sdtty -lines 36 a2. The default size is 25.

8.4 Sdtty on Unix

The Unix version normally uses the *curses* display access mechanism. It should make effective use of the screen-editing features of the terminal (VT-100 or emulation of same, or whatever) to keep the screen updated. However, this requires that all of the system facilities for dealing with terminals (`TERM` environment variable, `terminfo` database, `stty` behavior, etc.) be working properly. If, for any reason, this doesn't work for you, you can turn the *curses* mechanism off completely by using the `-no_cursor` command-line switch at program startup. This should even make it possible to use the program from a printing device.

When using *curses*, it normally tries to use the insert/delete line capabilities of your terminal device to update the screen more efficiently. For certain terminal or modem configurations, this may be counterproductive. If so, you can turn this feature off by giving the command-line switch `-no_line_delete` at program startup.

9 Linguistic Idiosyncrasies

In a number of cases, the program behaves in a way that could be considered idiosyncratic, peculiar, or simply wrong. Some of these cases are admittedly bugs or shortcomings of the program. Others are inevitable consequences of what the program is trying to do.

The syntax, semantics, and general structure of the square dance calling language is fairly regular and precise—much more so than, for example, the English language. That is, the correspondence between dancer actions and verbal phrases is fairly regular. However, there are exceptions. A very simple example of this is the fact that callers generally say '`quarter top`' and '`half the top`', using the word "the" in one case but not the other.

The program handles these exceptions by calling in an occasionally stilted but unambiguous computer-ese dialect. This dialect is intended to be very close to the words you would use when calling, but it is sometimes different. The rationale for this is that it is too hard to make the program always use the words that a caller would use. When writing sequences, be aware that the dancers do not necessarily know this computer-ese dialect. Use the words that you think are correct and natural. Your judgement is much better than any program's judgement can ever be.

The program sometimes requires you to say various *helper* words, such as `left` in cases where their necessity is is not universally agreed upon. For example, from a left-handed quarter-tag, `Sd` will not accept `scoot and plenty`—you are required to enter `left scoot and plenty`, even though the Callerlab definition does not require the word `left` in this case. What helper words you use or don't use when calling is, of course, up to you.

There are some innocent-looking things that the program can generate for which appropriate words do not exist. You must not write sequences containing such things, because you won't be able to call them. For example, from boy-boy-girl-girl waves, it is perfectly straightforward for a caller to say '`boys hinge`'. From boy-girl-boy-girl waves, '`boys hinge`' is meaningless, but the program accepts '`boys do your part, hinge`'. It doesn't know that this is an unacceptable thing to call. Don't do it.

In general, if you don't think you can clearly express to the dancers what you want, don't call it.

A number of concepts are particularly vulnerable to this phenomenon. Examples are `precede it by`, `follow it by`, the fractional concepts like `1/2`, `1-1/2`, and `3 times`, and the calls that designate certain people.

Callers often use phrases such as '`finish` ...' or '`like a` ...' in imprecise ways. Sd uses a very precise definition for these phrases. See Section 12.5 [Miscellaneous Concepts], page 65.

10 Using Off-Level Concepts

Because the program is so literal-minded, it sometimes doesn't understand how to do things that are in fact very easy for callers to express and for dancers to do. In some cases, the necessary sophistication lies in some concept that is legal only at high challenge levels.

When such a situation arises, you can tell the program to permit the use of concepts that would not normally be legal. The command to do this is `toggle concept levels`. See Section 7.7 [Changing Modes], page 43.

If you have a line of 6 (or 8) people, and you want them to do a '`1/2 tag`', use the `3x3` (or `4x4`) concept. Of course, if you were calling mainstream, you wouldn't say '`3x3 1/2 tag`'. You would presumably say something like '`line of 6 in the center, 1/2 tag`'.

If you have a column of 6, and you want them to '`walk and dodge`' in that setup, that is, the lead two dodge while the other 4 walk forward, use the `2X1` concept. Use your own words when calling this.

If you want people to do something (e.g., '`diamond circulate`') around the outside, you may need to use the `disconnected` concept. For example, if you had waves with the boys on the ends, and you had just done a '`1/2, acey deucey`', the boys have a big diamond around the outside. You can't just do a `BOYS diamond circulate`, because the `<ANYONE>` concept is extremely fussy. See Section 12.10 [Designating Certain People], page 79. The way to do this is with `BOYS DISCONNECTED diamond circulate`. This concept is only legitimate at C2, so you will need `toggle concept levels` below that. You would presumably say something like '`boys diamond circulate around the outside`'.

Whenever you use an off-level concept, a warning will be printed in the transcript. You should read such a card using whatever words are appropriate to get the dancers through the action that you intend.

11 Call Notes

11.1 Sweeping Direction

'`Sweep 1/4`', '`with the flow`', and '`by golly`' are somewhat unsophisticated in the way they calculate the sweeping direction. The program infers the sweeping direction from the roll direction of the preceding call. This is known to work for the common calls such as *recycle*, but may do something tasteless if used with an unusual call.

11.2 C1 *Single Rotate*

'`Single rotate while the others`' is intended to be called to the heads or sides from a squared set. It performs the common C1 usage of this call. It is nonsensical in other contexts.

11.3 Spread

The term *spread* can mean different things and so is three calls in this program.

Use the call '`spread`' after calls such as *follow your neighbor*.

Use the call '`... and spread`' after *wheel and deal*, or after starting a sequence with something like '`heads star thru`'. You can also use the call '`wheel and spread`'.

Use the call '`<ANYONE> spread`' from columns, to get the designated people to slide away from their partner into a 4x4 matrix.

11.4 <ANYTHING> and circle

Callers use the `<ANYTHING> and circle` construction with either an 8-person call to be done by everyone, or a 4-person call to be done just by the centers, and dancers generally have no confusion about it. The program is not so clever. The `<ANYTHING> and circle` call has all 8 people do the call, so, for example, it could be used with `ping-pong circulate` or `scoot and plenty`. If you want just the centers to do the <ANYTHING> call, you can use the `centers` concept. For example, you could select `<ANYTHING> and circle 1/4`, then `centers`, then `switch the wave`. The result will come out as `[CENTERS switch the wave] and circle 1/4`. You presumably would not say the word '`centers`' when calling this.

There is another way to get this effect is to select `simple modifications` followed by `swing and circle`. When this method is used, the modification call will be done by the centers only. So you could select `simple modifications`, then `swing and circle 1/4`, and then give the modification call `switch the wave`.

If all else fails, there is a call `and circle 1/4` that just does the final circling part.

11.5 Centers Back Away

This call is intended for use in sequences that begin with heads or sides doing a '`ladies chain`' or '`right and left thru`' or something similar. After they do such a call, they will remain in the center. To have the same people do the next call (for example, '`pass the ocean`' or '`star thru`'), just use the '`centers`' concept. If you want the *inactive* dancers to come in and do the next call, use the call '`centers back away, others come in and <ANYTHING>`'. So, for example, you might enter:

```
heads start
ladies chain
centers back away, others come in and <ANYTHING>
star thru
```

You might read this as "head ladies chain, then the sides move in and star thru".

11.6 <ANYONE> Promenade

These calls are intended for use in sequences that begin with heads or sides promenading halfway around the set. The calls are

```
promenade halfway, come in to the middle and <ANYTHING>
```

and

```
promenade halfway, while the others <ANYTHING>
```

In each case, start the sequence with the `heads start` or `sides start` command, and then issue one of the calls listed above. Ignore the fact that the displayed setup will look strange before the promenade. You will be prompted for the '`<ANYTHING>`' call, which will be done by the appropriate people in the center of the set.

11.7 Ends Divide

This is a call that should be used when you want the outsides to move along until they are facing each other. For example, after '`heads swing thru`', you might say '`ends divide`', and then have them '`star thru`'. Note that '`ends divide`' is the name of the call. It is not an application of the '`ends`' concept. You must type it on one line, and you may not substitute another designator, such as '`sides`', for the word '`ends`'. So, for example, you might enter:

```
heads start
swing thru
ends divide
ends touch 1/4
6x2 acey deucey
```

You might read this as "heads swing thru, while the sides divide and touch 1/4".

11.8 Little, Little More, and Plenty

These are problematical calls when the outsides are in line-like, rather than column-like, orientation. This situation arises, for example, when the starting setup is twin diamonds. The Callerlab C1 and C2 definitions state that in this case the outsides counter rotate just as they are, without first quartering in some direction. However, many high-level callers have adopted a definition that says that the outsides *always* quarter right (or take whatever direction is given) before counter rotating. Sd uses this formulation. To handle all the possibilities, there are 3 varieties of the call '`little`'—just '`little`', '`little, ends face <DIRECTION>`', and '`little, ends go as you are`'. Use the last one when the dancers are in diamonds and you want the outsides to counter rotate directly. Whether or not you say "ends go as you are" is up to you. The calls '`little more`' and '`plenty`' have analogous behavior.

11.9 Flip Back

This call is a special case of the '`<ATC> back to a wave`' class of calls, and is a recognized C1 call. You should use this name rather than the potentially misleading '`flip back to a wave`'.

11.10 Face <DIRECTION>

The call '`face <DIRECTION>`' does not appear to be a genuine Callerlab-sanctioned call. It is intended to be used, for example, after '`tag the line`', to give the direction that you wish to have the dancers face. So, for example, you could enter the calls '`tag the line`' followed by '`face in`' to get the action that is commonly expressed as '`tag the line, in`'. Note that the A2 designators '`zig-zag`', etc., are among the permissible directions.

11.11 Grand Chain 8

In order to handle all of the nuances of '`grand chain 8`', it was necessary to make a separate call '`grand chain 8 (centers and ends)`'. Use this call when you want just the centers or ends to do their part, or you are using some concept like '`central`'. In other circumstances, just '`grand chain 8`' should work.

11.12 Snag Circulate

The '`snag`' concept, meaning to have the centers do just half of the call, does not work with the call '`circulate`', because that call is not defined in the database as having centers' and ends' parts. Hence, the separate call '`snag circulate`' has been provided.

11.13 Funny Square Thru

Because of the peculiarity of this call, this call was not implemented with the '`funny`' concept. Instead, it is a call by itself. This means that, in `Sdtty`, you type `funny square thru <n>` on one line, rather than pressing ⟨ENTER⟩ after the word `funny`. In `Sd`, the call menu will have `funny square thru <n>`.

11.14 Rotate from Columns

Remember that *rotate* and *single rotate* from columns is legal only at C3, but from lines it is legal at C2. Since the program has no knowledge of different starting setups implying different levels, the call is in the database at C2. Don't use it from columns at that level.

11.15 Recycle

There are two *Recycle* calls in the database: '`recycle`' can be done from waves, facing couples, right-handed boxes, or a couple facing a right-handed mini-wave. It can take the '`cross`', '`single`', or '`diamond`' modifiers. It is the call that dancers will do when told to *recycle*, and is the call that is used in *cover up* and *counterpoint*.

Unfortunately, there is no mechanism for the program to recognize a call as being fractionalizable in some setups but not others, and the facing-couples version of recycle is not fractionalizable. To permit fractional use of recycle, as with '`interlace`' or '`random`', we have provided a call '`recycle (by parts)`', which can be done only from waves. Use this call (but don't read the "by parts" phrase) when fractions are required.

11.16 Going Through Impossible Situations

Under certain circumstances, people who collide in the same spot during a call will take right hands with each other, according to well-known rules governing this type of occurrence. Other than that, the program does not allow "impossible" intermediate situations in which multiple people occupy the same spot.

In some cases, you may want to call something that sends people through a momentary impossible situation, for example calling something like '`everyone do your part, split trade circulate twice`' when people are in normal waves. The first '`split trade circulate`' is impossible, but if you tell the dancers to do their own part and not worry about the collision after the first one, they can do it.

The program doesn't like to do this. Just calling `twice split trade circulate` won't work, because the program will have multiple dancers on the same spot after the first one. You can trick the program into doing it anyway, by using the `<ANYONE> do your part` concept. Enter something like `heads do your part, twice split trade circulate while the others twice split trade circulate`. Exactly who has to be designated depends on the the setup.

You can also use this method to get everyone to do their part of calls that are illegal in the existing setup. For example, flip back is illegal from facing lines. If you really want people to do it anyway, with everyone doing their part, you can enter `beaus do your part, flip back while the others flip back`.

12 Concept Notes

Most concepts have clear and precise meanings, and do not need any further discussion here. However, for a significant number of concepts, the distinction between precisely defined, universally understood terminology, and "common sense" description of what you want the dancers to do, becomes rather blurred. When using computers to write choreography, a lot of questions of the form, "How do I get the program to do this?" arise. For that reason, a complete list of available concepts will be given here. For many of them, their meaning is clear, and nothing further will be said. For the troublesome ones, we will attempt to describe just what the program understands them to mean.

For many of the less straightforward cases, the words that the program uses are not necessarily the words that you would use to get the dancers to do the equivalent thing. You must always use your own judgement. No claim is made that the program's notion of what the words literally mean is the universally accepted definition of the concept.

In the menu-based program (Sd), most of these concepts are on special menus that are brought up by clicking on one of the concept icons on the left side of the screen. There are far too many concepts to make them all visible on the screen at the same time. In `Sdtty`, just type the name of the concept without worrying about what menu it would be on.

Be aware that those concepts that take two calls, such as `checkpoint`, `two calls in succession`, or `<ANYONE> do your part (while the others)` must be entered by themselves. That is, you must press ⟨ENTER⟩ after typing any of these concepts. You can't type

`checkpoint ah so by recycle` ⟨ENTER⟩

or

`boys trade (while the others) u-turn back` ⟨ENTER⟩

You must type

```
checkpoint ⟨ENTER⟩
ah so ⟨ENTER⟩
recycle ⟨ENTER⟩
```

or

```
boys (while the others) ⟨ENTER⟩
trade ⟨ENTER⟩
u-turn back ⟨ENTER⟩
```

The concepts to watch out for are labeled "[must be entered by itself]" in the following lists.

Just what constitutes a "concept" and what constitutes a mere "variation" of a call is not always clearly delineated. We list below only those things that the program considers to be "concepts." There are many other variations of calls, such as *square chain thru to a wave* that are not listed here. In general, variations of this sort can be obtained simply by clicking on them or typing them. They should be visible on the call menu, and, if you type a question mark to `Sdtty` while typing the call name, these variations should be displayed.

In the concept names, "C/L/W" has the usual meaning of "columns, lines, or waves." Extending that terminology, "B" means boxes, "D" means diamonds, "1" means 1/4 tags, and "3" means 3/4 tags. "DS" means "diamond spots", that is, diamonds or general 1/4 tags in which no assumption is made about anyone's facing direction. "GT" means "general 1/4 tags", in which the ends could individually be facing either in or out.

In all concepts that can specify either lines or waves, the "waves" version has an implicit "assume waves" operation in it. Under normal circumstances, this means that the live people can make use of the assumption of waves in order to decide how to start the call. For example, in normal columns, `split phantom waves in roll circulate`, which is equivalent to `split phantom lines, assume waves, in roll circulate`, is legal, since the phantom in-roller's location can be deduced from the wave assumption.

The call `split phantom lines in roll circulate` is illegal from columns because the facing direction of the phantom in-roller is unknown.

If "active phantoms" are used, either because the `with active phantoms` concept is used after the phantom wave concept or because the `toggle active phantoms` command was given, the phantoms will work throughout the entire call, based on the initial facing direction in waves.

The concepts with the word `diamonds` have an implicit `assume general diamonds` operation in them. This means that everyone must think they are individually in some kind of diamond, with their right or left hand toward the center. You may give an explicit `assume` concept to make the setup more specific if you wish, as in `split phantom diamonds` followed by `assume interlocked diamonds`.

The concepts with the words `1/4 tags` or `3/4 tags` require some kind of line in the middle, and the ends as a couple facing in or out, respectively. Hence, for example, the `split phantom 1/4 tags` concept includes 1/4 tags, 1/4 lines, or any kind of line in the center, as long as the ends are facing in. You may give an explicit `assume` concept to make the setup more specific if you wish, as in `split phantom 1/4 tags` followed by `assume right 1/4 lines`.

The concepts with the words `general 1/4 tags` have an implicit `assume general 1/4 tags` operation. This means that the centers must think that they are in some kind of line, and the ends must be in some kind of couple or miniwave. The word *general* means that the in-or-out facing direction of the outsides is unimportant, but they must be facing in or out, not sideways as in diamonds. There are no `general 3/4 tags` concepts, because the `general 1/4 tags` setup does not distinguish between ends facing in or out.

The concepts with the words `diamond spots` make no assumptions at all about facing direction. The people can be in any kind of general diamond or general 1/4 tag. Use these when people's facing directions are bizarre.

Precisely what an assumption means is determined by whether the "active phantoms" mode is turned on, and is discussed in Section 12.8 [Assume Waves], page 76.

12.1 Phantom Concepts

These involve picking out virtual setups, which may include phantoms, from a real formation that is larger than eight people.

The menu for these is brought up by clicking on the "phantom line/col concepts" or "phantom box/dmd/qtag concepts" icon.

In all cases outboard phantoms are added as required at the start of the call, and removed where possible at the conclusion of the call.

`split phantom C/L/W/B/D/1/3/DS/GT`

`interlocked phantom C/L/W/B/D/1/3/DS/GT`

`phantom C/L/W/B/D/1/3/DS/GT`
: These are straightforward.

`12 matrix split phantom C/L`

`12 matrix interlocked phantom C/L`

`12 matrix phantom C/L`
: These are done from a 3x4. The setup is separated into parallel 2x3's, and the call is done in each of those setups.

`divided C/L/W`
: These are done from a 2x8. The setup is separated into end-to-end 2x4's, and the call is done in each of those setups. They are equivalent to, and generally replaced by, *split phantom boxes*. If people are T-boned, you can use `standard` to permit the concept to be used.

`12 matrix divided C/L/W`
: These are similar to *divided C/L/W*, except that they are done from a 2x6. The setup is separated into end-to-end 2x3's, and the call is done in each of those setups. There is no *split phantom boxes* equivalent for these.

`twin phantom tidal C/L/W`
: These are done from a 2x8, and form two parallel 1x8's with phantoms, in each of which the call is done independently.

  The program uses the term "tidal" to mean a 1x8. You may prefer to use the word `grand`.

`twin phantom C/L/W of 6`
: These are done from a 2x6, and form two parallel 1x6's with phantoms, in each of which the call is done independently.

`triple C/L/W/B/D/1/3/DS/GT`

`quadruple C/L/W/B/D/1/3/DS/GT`
: These are straightforward.

`center triple C/L/W/B (singular)`

`outside triple C/L/W/B`
: These are done from a setup that can be construed as having a center 4-person setup with two other 4-person setups on the outside. The center setup can be different from the outside setups, and reasonable independent shape-changers are allowed.

A suitably populated 4x4 matrix can be construed as a center triple box surrounded by outside triple lines or columns.

`center phantom C/L/W/B`

`outside phantom C/L/W/B`

These are done from a setup that can be construed as having a center 8-person setup with two other 4-person setups on the outside. The center setup can be different from the outside setups, and reasonable independent shape-changers are allowed.

A common case of this is a 4x4. The center phantom lines or columns can do an 8-person call. The outer phantom lines or columns can do independent 4-person calls.

`triple C/L/W/B/D/1 working together`

`triple C/L/W/B working apart`

`triple L/W/B working forward/backward`

`triple C/B working right/left`

`triple C/L/W/B working clockwise/counterclockwise`

`quadruple C/L/W/B/D/1 working together`

`quadruple C/L/W/B working apart`

`quadruple L/W/B working forward/backward`

`quadruple C/B working right/left`

`quadruple C/L/W/B working clockwise/counterclockwise`

`quadruple C/L/W/B/D working toward the center`

These are straightforward.

`triple diagonal C/L/W`

`triple diagonal L/W working forward/backward`

`triple diagonal columns working right/left`

These are done from a 6x6 matrix. For example, in "blocks", the main diagonal and the two adjacent diagonal lines of 4, each of which has only centers occupying it, comprise the triple diagonal lines. These concepts are quite obscure.

`triple twin C/L/W`

This is done from a 4x6 setup. Three parallel 2x4 setups are formed, and the call is done independently in each of them.

`concentric triple boxes`

`concentric quadruple boxes`

These are done from a 2x6 or 2x8 respectively. The concentric boxes work independently.

`twin phantom D/1/3/DS/GT`

These are two sets of parallel diamonds or 1/4 tags (that is, setups in which you could call `6x2 acey deucey` or `scoot and plenty`) that are next to each other along the opposite orientation to the orientation given by split phantom diamonds. That is, there is a 2x2 matrix of diamonds or 1/4 tags. The diamonds in each parallel pair work with each other, but the pairs work independently.

12.2 Tandem or Couples Concepts

The menu for these is brought up by clicking on the "couples/tandem concepts" icon.

as couples	couples twosome
tandem	tandem twosome
siamese	siamese twosome
couples of three	couples threesome
tandems of three	tandem threesome
couples of four	couples foursome
tandems of four	tandem foursome
boxes work solid	boxsome
diamonds work solid	diamondsome
skew	skewsome
<ANYONE> are as couples	<ANYONE> are couples twosome
<ANYONE> are tandem	<ANYONE> are tandem twosome
<ANYONE> are couples of three	<ANYONE> are couples threesome
<ANYONE> are tandems of three	<ANYONE> are tandem threesome
inside triangles work solid	inside triangles work threesome
outside triangles work solid	outside triangles work threesome
in point triangles work solid	in point triangles work threesome
out point triangles work solid	out point triangles work threesome
wave-based triangles work solid	wave-based triangles work threesome
tandem-based triangles work solid	tandem-based triangles work threesome
<ANYONE>-based triangles work solid	<ANYONE>-based triangles work threesome

These are straightforward.

Any of the above twosome/threesome/whateversome may also be "<N/4> whateversome" or "whateversome <N/4> solid"

Any of the above may be used with the phantom concept in front of it. There are no concepts like phantom tandem listed separately—you must use the phantom concept followed by the concept that you want. Of course, when using `Sdtty`, this issue doesn't make any difference—you can just type

`phantom tandem swing thru` ⟨ENTER⟩

The *phantom tandem/as couples* concepts expand the people into a 4x4 matrix whenever possible. When that is not possible, a 2x8 matrix is formed. You can use the `2x8 matrix` concept in order to force this. For example, from normal columns, `phantom tandem` puts all the real people in tandem with each other, and makes them the centers of *lines*. You could call a `ferris wheel`, but not a `checkmate the column`, from this setup. If you wanted the pairs of real people to be centers of columns, so that you could call `checkmate the column`, use `2x8 matrix phantom tandem checkmate the columns`.

`gruesome twosome`
: This is done from a 2x8. It has an implicit "assume waves" in it.

12.3 Distorted Setup Concepts

The menu for these is brought up by clicking on the "distorted setup concepts" icon.

`stagger`

`big block (any kind of lines)`

`big block waves (this means ''assume waves'')`
: These encompass any kind of distorted lines or columns.

`O`

`butterfly`
: These are straightforward.

`stairstep C/L/W`
: A "stairstep" is a 4x4 matrix occupied so that everyone has exactly one live person on their left or right. It could typically be obtained by doing a `split phantom columns split circulate` from normal columns. You may wish to use the `distorted C/L/W` concept instead.

`ladder C/L/W`
: A "ladder" is a 4x4 matrix occupied so that everyone has exactly one live person in front or in back. It could typically be obtained by doing a `split phantom waves split circulate` from normal waves. You may wish to use the `distorted C/L/W` concept instead.

`offset C/L/W`
: An "offset" C/L/W is a pair of lines or columns that has been sheared in the middle by either 50% or 100%, making a 3x4 or 4x4 matrix, respectively. A 4x4 matrix offset line or column is sometimes referred to as "clumps". All of the live people work together accross the shear line. If a shape-changer is called, the setup may become a 2x8 matrix, that is, a 100% offset parallelogram.

`Z C/L/W`
: This is a pair of lines or columns that have been distorted by having the centers or the ends move one full matrix spot, to make a 4x4 matrix, or by having centers and adjacent ends move 1/2 matrix spot in opposite directions, to make a 3x4 matrix. The people move together in pairs. The 4x4 matrix version can be formed by having either the centers or the ends press ahead from waves. The 3x4 matrix version can be formed by having everyone 1/2 press ahead from waves.

`distorted C/L/W`
: This encompasses any kind of distortion of a pair of lines or columns into a 3x4 or 4x4 matrix.

`diagonal C/L/W (singular)`

`<ANYONE> in your diagonal C/L/W (singular)`
: These require a "blocks" setup. They specify the "long diagonal", recognized as either a line or column. You may specify who the people are (heads, boys, etc.) or not, as you choose.

`diagonal C/L/W (plural)`
: These are done from a 4x6. The setup has 8 real people, doing an 8-person call. The setup could be formed from stairsteps by having the ends press ahead or back such that each line is diagonal and straight. These concepts are quite obscure.

`diagonal C/L/W of 3 (plural)`
: These are done from "blocks". The two lines or columns work together in a 2x3 matrix, doing a 6-person call.

`distorted tidal C/L/W (singular)`
: These are done from a 2x8 matrix in which the real people form some kind of distorted 1x8.

`<ANYONE> in your distorted C/L/W/B/D (singular)`
: These are done from a variety of setups, in which the designated people form a recognizable distorted setup. Reasonable shape-changers are allowed.

`parallelogram`
: This includes 100% offset parallelograms in 2x8 matrices.

`phantom big block L/W`

`phantom stagger columns`

`phantom stairstep C/L/W`

`phantom ladder C/L/W`

`phantom offset C/L/W`

`phantom butterfly or O`

These are the phantom versions of the indicated concepts, in which each dancer works in the spots of indicated setup, with real people or phantoms as required.

12.4 4-person distorted concepts

The menu for these is brought up by clicking on the “4-person distorted setup concepts” icon.

`split` This divides a 2x4 into two 2x2’s. It also has all the other meanings of the word “split”. The distinction between this as a concept and as a word appearing in a call name is extremely hazy.

`once removed`

`once removed diamonds`

The `once removed` concept encompasses diamonds. so you can say either `once removed` or `once removed diamonds`.

`magic`

`diagonal`

`trapezoid`

These are straightforward.

`overlapped diamonds`

This is done from a 1x4.

`overlapped lines`

`overlapped waves`

These are done from a diamond.

`interlocked parallelogram`

This is either done from a 2x4 or a 3x4. In the latter case, the spots occupied by real people are used to form two interlocked parallelograms sharing the center line or column.

`interlocked boxes`

This is done from 3x4 matrix “Z” lines or columns. The spots occupied by real people are used to form two elongated boxes sharing the center line or column. It is equivalent to `Z C/L/W once removed`.

`twin parallelograms`
: This is done from 3x4 matrix offset lines or columns. The people that would form each distorted line or column instead form a distorted box.

`each Z`

`interlocked Z's`
: These are done from 3x4 or 4x4 "Z" lines or columns, and are equivalent to `Z C/L/W split` and `Z C/L/W once removed`, respectively.

`jay`

`back-to-front jay`

`back-to-back jay`
: These include the cases of a 2-faced line in the center. In that case the shapes of the resultant 4-person boxes are different, and the concepts are equivalent to the "parallelogram" concepts below.

`facing parallelogram`

`back-to-front parallelogram`

`back-to-back parallelogram`
: These are other names for the `jay` concepts, except that the center line is required to be a 2-faced line.

`blocks`

`in your blocks`
: These are two names for the same thing.

`4 phantom interlocked blocks`
: These are straightforward.

`triangular boxes`
: This may only be done from a "blocks" setup.

`4 phantom triangular boxes`
: This may be done from any 4x4 matrix.

`distorted blocks`
: This may be done from a 4x4 matrix that comprises blocks with some people out of their usual place in an unambiguous way. People work to the same spots.

12.5 Miscellaneous Concepts

The menu for these is brought up by clicking on the "miscellaneous concepts" icon. Many also appear in the special concept menu that is always visible.

`left`

`reverse`

`cross`

`single`

`grand`

`mirror`

`interlocked`
: These are straightforward.

`single file`
: You must use this concept, instead of the words “on a double track” or whatever, when doing calls like *dixie style* from a single file starting double pass thru.

`triangle`
: This turns a triangle into a box by having the apex step backward. It is intended for things like `triangle peel and trail`.

`diamond`
: This is for things like `diamond quarter thru` and `diamond single wheel`. You should not need to select `diamond` except when you actually want this concept. There are cases in which it may seem that the word *diamond* ought to be added after selecting `interlocked` and/or `magic`. The program will insert the extra word *diamond* for you. So, for example, selecting `magic` and then `alter the diamond` will get ‘`magic diamond, alter the diamond`’, and selecting `magic`, `interlocked`, `diamond`, `as couples`, and `quarter right` will get ‘`magic interlocked diamond, diamond as couples quarter right`’.

`12 matrix`

`16 matrix`
: See Section 12.11 [12 Matrix and 16 Matrix], page 84.

`phantom`
: This is both the C1 concept, and the concept required in front of a tandem or as-couples concept to turn it into ‘`phantom tandem`’ or ‘`phantom as couples`’.

`funny`
: This can handle ‘`circulate`’-like calls and ‘`square thru`’, but can’t handle things like ‘`funny slide thru 3 times`’

`matrix`
: This is used in front of concepts like ‘`split phantom waves`’ to indicate that the normal “glueing” rule is replaced by a rule that makes each setup stay centered on its original center, possibly resulting in overlap among the setups.

`assume <some setup>`

`assume normal casts`

`with active phantoms`
: See Section 12.8 [Assume Waves], page 76 and Section 12.9 [Assuming a Quarter-Tag], page 78.

`invert` This inverts the centers' and ends' parts of the call.

`fan`

`yoyo`

`central`

`invert central`

`snag`

`invert snag`

`mystic`

`invert mystic`
These are straightforward.

`<ANYONE> are standard in`
This is used in front of concepts like '`split phantom lines`'.

`stable`

`<ANYONE> are stable`

`<N/4> stable`

`<ANYONE> are <N/4> stable`

`trace` `[must be entered by itself]`

`ferris`

`release`

`stretch`

`stretched setup`

`stretched box`

`stretched line`
These are straightforward.

`centers`

`ends`

`centers and ends` `[must be entered by itself]`

`center 6`

`outer 6`

`center 2`

`outer 2`

`center 6/outer 2` `[must be entered by itself]` `center 2/outer 6` [must be entered by itself]

`<ANYONE>`

`<ANYONE> (while the others)` [must be entered by itself]

`<ANYONE> disconnected`

`<ANYONE> disconnected (while the others)` [must be entered by itself]

`<ANYONE> do your part`

`<ANYONE> do your part (while the others)` [must be entered by itself]

`on your own` [must be entered by itself]

`own the <ANYONE>` [must be entered by itself]

`ignore the <ANYONE>`

These are straightforward. See Section 12.10 [Designating Certain People], page 79 for a detailed discussion of these concepts.

`<ANYONE> lead for a`

This is used from a "promenade, do not stop ..." situation. The call will be done as if in 2-faced lines, with the designated people in the lead. The final promenade distance will be computed as though you had allowed the dancers to promenade about halfway around the set before they did the call.

`ends concentric`

`centers and ends concentric` [must be entered by itself]

These are like '`ends`' and '`centers and ends`', except that the concentric rules (e.g., lines-to-lines and columns-to-columns) are used to dictate where the ends should finish.

`checkpoint` [must be entered by itself]

`reverse checkpoint` [must be entered by itself]

`checkerboard`

`checkerbox`

`checkerdiamond`

`<ANYONE> preferred for trade, checkerboard`

`<ANYONE> preferred for trade, checkerbox`

`<ANYONE> preferred for trade, checkerdiamond`

`shadow box`

These are straightforward.

`<ANYONE> work <concept>`

This concept is rather difficult to do in the most general case. The program executes it effectively as an '`own the <ANYONE> for a <concept> <call> by <call>`'. That is, `ends work tandem, swing thru` is performed by having the ends do their part of a '`tandem swing thru`' while the centers do a '`swing thru`'. This concept is entered by typing the designated people instead of the '`<anyone>`', and then "work `<concept>`". You literally type the "`<concept>`", including the angle brackets. (Of course, type-in completion makes it unnecessary to type the whole thing.) After you press ⟨ENTER⟩, type the concept, press ⟨ENTER⟩, and type the call.

`two calls in succession` `[must be entered by itself]`

This concept allows a pair of calls to be executed atomically under a concept. This makes it possible to do a '`concentric (recycle ; roll)`' and have the lines-to-lines rule embrace the roll. It also makes it possible to place two calls on the same line that need to be spoken together for clarity, such as '`recycle and roll`'. The finished transcript will show the two calls within parentheses, and separated by a semicolon. The proper way to say the concept when calling is problematical. Something like "consider the following two calls to be one unit, and do a stable swing thru and turn thru" might be appropriate.

`precede it by` `[must be entered by itself]`

`follow it by` `[must be entered by itself]`

The concepts `precede it by` and `follow it by` take two subject calls. They perform the first call (first in textual order, that is) *before* or *after*, respectively, the second. For example, `follow it by roll, swing thru` does a swing thru and roll. There is only one context in which these concepts are sensible—they can be the subject of the `piecewise`, `random`, or `reverse random` meta-concepts. For example, `piecewise follow it by criss cross the deucey, turn the key` will do a '`criss cross the deucey`' after each of the three parts of '`turn the key`'.

These concepts are among the situations in which the program does not attempt to print out the exact words that are appropriate to use when calling, but instead prints out an unambiguous description of what is supposed to happen. The exact words you should use depend on the context and on your judgement. You might say `piecewise add a criss cross the deucey, and turn the key`, or `do a criss cross the deucey after each part, and turn the key`, or something similar.

`crazy`

`reverse crazy`

`<N/4> crazy`

`<N/4> reverse crazy`

These are straightforward.

`<N>/<N>`

`1-<N>/<N>`

`twice`

`<N> times`

`do the last <N>/<N>`

The `M/N` concept, where `M` and `N` are numbers, causes that fraction of the subject call to be executed, as in '`3/5 swing the fractions`'. In `Sdtty`, just type in the numbers directly, as in `3/5`. In `Sd`, the concept is listed as `<M/N>`. A popup will appear asking for the numbers. Enter two of them.

The `1-M/N` concept causes the subject call to be executed once and the indicated fraction of a second time, as in '`1-1/2 split circulate`'.

Not all calls are fractionalizable in this way. Also, the exact way you should say the fraction might vary according to the call or your individual taste. It will always appear before the call in the transcript, but you might want to say the fraction after the call, as in '`swing the fractions four fifths`' or '`split circulate once and a half`'.

In extremely tricky cases, such as '`1/2, mix`', '`3/4, mix`', '`3/4 mix`', and '`2/3, 3/4 mix`'. these concepts can lead to ambiguity. It may be necessary to type the concept by itself, that is, to press ⟨ENTER⟩ at the appropriate moment, to get what you want. If you really want 3/4 of the call '`mix`', type

```
3/4 ⟨ENTER⟩
mix ⟨ENTER⟩
```

In the finished transcript, a comma will appear after fractionalizing concepts, but not in the call `3/4 mix`. You of course never type any commas.

The `twice` concept simply causes the subject call to be executed twice. This is a notion familiar at all levels from Mainstream ('`spin chain thru, the girls double circulate`') to C4 ('`piecewise twice, recoil`').

The `<N> times` concept causes the subject call to be executed that number of times. In `Sdtty`, just type in the number directly, as in `3 times`. In `Sd`, the concept is listed as `<N> times`. A popup will appear asking for the number.

No universally recognized words for these exist, so the program uses the generic phrases noted above. When calling, use whatever words you think are appropriate, such as '`twice`', '`thrice`', '`circulate two positions`', or whatever.

`piecewise`

`initially`

`finally`

`random`

`reverse random`

These are followed by the concept to which they apply.

`shifty`

`shift <N>`

`interlace` [must be entered by itself]

`reverse order`

These are straightforward.

`finish`

`like a`

To Sd, `finish` means "skip the first part," and is only legal for calls for which the first part is recognizable. Sd also recognizes the concept `like a`, meaning "do the last part." You need to be aware, however, that there are calls for which, while it is technically legal to say such things, they are not accepted as common usage. For example, `finish shazam` as a way of saying *U-turn back*, or `finish mix` or `like a mix` as ways of saying *centers trade*, would have to be considered rather peculiar. You must exercise your judgement when using these concepts.

We believe that this is consistent with common usage when that common usage is correctly applied. For example,

```
cast off 3/4
LIKE A shazam
```

is a commonly accepted and completely correct usage.

One occasionally hears things like *cross chain reaction, centers finish like a wheel the ocean*, in which the final cast off 3/4 of the chain reaction was "pushy." While the plain English language meaning of that is clear, the way you must enter that to Sd is

```
CROSS chain reaction
CENTERS LIKE A wheel the ocean
```

You must, of course, verify that the casting direction is correct for smooth dancing of this figure.

`<ANYONE> start`

`skip the <Nth> part`

`do the <Nth> part <concept>`

`replace the <Nth> part` [must be entered by itself]

`interrupt after the <Nth> part` [must be entered by itself]

These concepts cause the call to be interrupted or replaced as indicated.

The `<ANYONE> start` concept has the indicated people do their own part of the first part of the call, after which the others join them. You should type this in as, for example

```
boys start quarter thru ⟨ENTER⟩
```

The `do the <Nth> part <concept>` concept applies the given concept to the indicated part of the call, but not to the rest. You literally type the "<concept>", including the angle brackets. (Of course, type-in completion makes it unnecessary to type the whole thing.) For example, you might type

```
do the 2nd part <concept> ⟨ENTER⟩
tandem remake ⟨ENTER⟩
```

or

```
do the 2nd part <concept> tandem remake ⟨ENTER⟩
```

The `replace` and `interrupt` concepts require two calls. After typing the concept, type the call that is to have a part interrupted or replaced. The program will then ask you for the call that comprises the interruption or replacement.

Some of the replacement/interruption meta-concepts push the semantics of the language to the limit. Whenever a call undergoes an interruption or enters or leaves a part with an additional concept on it, there is an implicit *piecewise* at that instant. That is, concepts and setups are re-evaluated. This behavior is not always obvious to the dancers. Furthermore, in some cases it may be perceived as wrong. For example, an implied *piecewise* on a concept such as *cross concentric* or *single cross concentric* is very likely to be considered wrong, because the centers and ends will switch with each other multiple times. If you have any doubt about whether some instance of concept stacking is correct and will be understood by the dancers, do not use it.

Furthermore, replacements and interruptions are *normal*, that is, they do not carry any concepts that were on the call being replaced—only those on the entire operation.

Example, from a starting DPT setup:

```
DELAY: TANDEM TWOSOME clean sweep 1/4 BUT REPLACE THE 3rd
PART WITH A [CHECKPOINT crossfire BY crossfire]
```

Normally, the tandem twosome behavior is not re-evaluated after each part of the clean sweep. But, since the third part was replaced with something else to which that concept did not apply, the setup is re-evaluated before doing the final part. Note in particular that the replacement was *normal*—the call having a part replaced was a *tandem twosome clean sweep 1/4*. If we wanted the entire operation, including the replacement, to be tandem twosome (with no re-evaluation, of course) we might call:

```
TANDEM TWOSOME DELAY: clean sweep 1/4 BUT REPLACE THE 3rd
PART WITH A [SINGLE CROSS CONCENTRIC turn thru]
```

If you think that some instance of a replacement or interruption will cause confusion on the dance floor because of this sort of issue, you should not use it.

`inside triangles`

`outside triangles`

`in point triangles`

`out point triangles`

`tall 6`

`short 6`

`wave-based triangles`

`tandem-based triangles`

`<ANYONE>-based triangles`

These are straightforward.

`concentric`

`cross concentric`

`single concentric`

`single cross concentric`

`concentric diamonds`

`cross concentric diamonds`

See Section 12.7 [Concentric], page 75.

`grand working forward/backward`

`grand working right/left`

`grand working as centers`

`grand working as ends`

`<ANYONE> are centers of a double-offset 1/4 tag`

`<ANYONE> are centers of a double-offset 3/4 tag`

`<ANYONE> are centers of a double-offset thing`

`<ANYONE> are centers of double-offset diamond spots`

`inrigger`

`outrigger`

`leftrigger`

`rightrigger`

`backrigger`

`frontrigger`

These are straightforward.

`common point galaxy`

`common spot columns`

`common point diamonds`

`common end lines`

`common end waves`

The program assumes that the “common spot” concepts treat people as though they had collided and taken *right* hands. Left-hand miniwaves are not accepted. If you want to use these concepts with people in left-hand miniwaves, you must apply the '`mirror`' concept to the entire operation.

The `common point galaxy` concept is used from a "rigger" setup in which the "wings" are collided points. The `common spot columns` concept is used from offset right-hand columns. The `common point diamonds` concept is used from the kind of setup one obtains after a '`6x2 acey deucey`' from facing diamonds. The `common end lines/waves` concept is used from a parallelogram in which the "wings" are collided ends of the lines.

`1x2/2x1/2x2/1x3/3x1/3x34x4`
: These are straightforward.

`all 4 couples`

`all 8`

`all 8 (diamonds)`
: The `all 4 couples` and `all 8` concepts appear to be undergoing some re-examination. The program's behavior on these might not be satisfactory at the present time. The `all 8 (diamonds)` concept is used from a thar, in which the call is intended to be performed in each of two interleaved diamonds. In real life, one can generally call this as just `all 8`, and the dancers will know what is required. The program is not that smart, and requires the "(diamonds)" hint.

12.6 Miscellaneous Concepts Appearing Only in the Main Menu

These appear only in the concept menu that is always visible on the left side of the screen.

`each 1x4`

`each line`

`each wave`

`each diamond`
: These can be used in troublesome situations to force a setup to be split in the indicated way. For example, from a tidal wave, the `each wave` concept can be used to cause a '`counter rotate`' to be performed as a '`lockit`'. The `each 1x4` concept splits the setup into 1x4 setups without regard for individual dancers' facing direction. From columns, `each 1x4 sidetrack` can cause the '`sidetrack`' to be done in individual columns (though `single sidetrack` will do the same thing).

`1x12 matrix`

`1x16 matrix`

`2x6 matrix`

`2x8 matrix`

`3x4 matrix`

`4x4 matrix`

`3x8 matrix`

`4x6 matrix`

`16 matrix of parallel diamonds`

These force the setup to be expanded as shown, to cause various other concepts to be interpreted in the desired way. For example, from parallel waves, the concept `phantom tandem` would expand the setup to a 4x4 matrix, making each person in tandem with a phantom, and creating a virtual 2x4. `2x8 matrix phantom tandem` would leave live people tandem with live people, create phantoms in tandem with other phantoms, and create a virtual tidal line. The `16 matrix of parallel diamonds` concept expands to quadruple diamond spots.

12.7 Concentric

The '`concentric`' and '`cross concentric`' concepts are formulated as follows: The rule tells how the people who finish on the outside should elongate their 2x2 if they finish in a 2x2. This is the only case that needs to be addressed.

1. If the people who finished on the outside in a 2x2 started in a 2x2, they use the "lines-to-lines/columns-to-columns" rule, relative to what setup *they* thought they were in.

   If the concept was *concentric*, so these people started on the outside in an elongated 2x2, they can make a decision about whether they had line or column elongation. They do not have to be in a 2x4 for this. For example, from a dogbone created by Heads touch, the sides think they are in columns. But from ordinary diamonds or an hourglass, they think they are in lines.

   If the concept was *cross concentric*, so these people started in the center in a 2x2, they must have the assistance of the original ends. The original setup must have been a 2x4, except for weird pinwheel-like setups noted below. Hence it is *illegal* to call *cross concentric square thru 2* from a rigger in which the center box is facing couples and the wings have right hands and are expected to rear back. The centers do not know whether they were in lines or columns because the ends are not in an acceptable setup. The centers do *not* use the "checkpoint elongate perpendicular to the 1x4 rule"—that rule does not apply here. If the outsides are in a pinwheel, that is, occupying a 4x4 matrix in an irregular way, each center makes the line vs. columns decision based on the end person in their own quadrant. That is, the center person's own facing direction and that end person's *location,* but not that end person's facing direction.

2. If the people who finished on the outside in a 2x2 started in a 1x4 or diamond, they use the *checkpoint rule*—they elongate their resulting 2x2 perpendicular to "the long axis". In order to handle all cases correctly, it is important to be very careful in choosing what that long axis is. As a good approximation, it is the long axis of "the whole set".

   If the concept was *concentric*, so these people started on the outside in a 1x4 or diamond, they use their own long axis, which is the same as the long axis of the whole set.

   If the concept was *cross concentric*, so these people started in the center in a 1x4 or diamond, what they do depends on what setup the original ends began in. If the original ends began in a 1x4 or diamond, the axis of that 1x4 or diamond is the same as the axis of the whole set. The original centers elongate their 2x2 perpendicular to that. This rule applies, for example, when '`centers lockit; cross concentric recycle`' is called from a tidal wave.

   If the concept was *cross concentric* and the original ends began in a 2x2, the original centers *ignore* the elongation of that 2x2. Instead, they elongate their resulting 2x2 perpendicular to *their own axis*, that is, the axis of the 1x4 or diamond in which they began. This rule applies, for example, when '`cross concentric recycle`' or '`center 2 shazam; cross concentric diamond recycle`' is called from a quarter-tag.

12.8 Assume Waves

There are several concepts that you can use to tell the dancers how phantoms are facing. They are

```
assume waves
assume miniwaves
assume couples
assume two-faced lines
assume one-faced lines
assume inverted lines
assume inverted boxes
assume normal columns
assume magic columns
assume eight chain
assume trade by
assume dpt
assume cdpt
assume facing lines
assume back-to-back lines
assume general diamonds
assume general 1/4 tags
assume right 1/4 tags
assume left 1/4 tags
```

```
assume right 3/4 tags
assume left 3/4 tags
assume right 1/4 lines
assume left 1/4 lines
assume right 3/4 lines
assume left 3/4 lines
assume normal diamonds
assume facing diamonds
assume normal interlocked diamonds
assume facing interlocked diamonds
```

Additionally, there are two related concepts, `assume normal casts` and `with active phantoms`, which will be discussed below.

When these concepts are used by themselves, the information that they convey tells the dancers how to begin certain calls such as '`hinge`', '`vertical tag`', and '`in roll circulate`'. The information is thrown away after the call begins. That is, the program does not hold the dancers responsible for tracking phantoms through a call. It only tells them how to begin.

Use these concepts with care. Many dancers don't like to be held responsible for the facing direction of phantoms.

These concepts will be implicitly applied whenever something like `split phantom waves` or `triple diamonds` is used.

It is of course illegal to use one of these concepts, either explicitly with `assume` or implicitly with something like `split phantom waves`, if the live dancers are not facing in a way that is consistent with the assumption.

12.8.1 Assume Normal Casts

The concept `assume normal casts` may be used in cases in which you do not require the dancers to track phantoms through the call, but you want to resolve any ambiguity about the '`cast off 3/4`' operations that happen at the end of such calls as '`chain reaction`' and '`motivate`'. You might read such an operation as '`12 matrix motivate, assume the final cast off is normal`'.

12.8.2 With Active Phantoms

The concept `with active phantoms`, used after one of the `assume` concepts (either implicit or explicit), effectively places invisible but otherwise intelligent dancers on the phantom spots, with facing directions deduced from the assumption. Those phantoms proceed through the entire call, interacting with each other and with the live dancers as necessary. If the call is impossible for the phantoms to do, or if illegal collisions occur between live dancers and phantoms, an error message is printed. At the end of the call, any live dancers or phantoms that come to the same spot may take right hands with each other, if that is appropriate for the call.

Most of the types of `assume` concepts listed above are sufficiently explicit that, if even a single live dancer is present in the setup, everyone's facing direction may be inferred. This is necessary for the `with active phantoms` concept, of course. For example, a single live dancer can determine the handedness of all the phantoms when `assume two-faced lines` is given. Two of the concepts are not sufficiently explicit for this, and may not be used with `with active phantoms`. They are

```
assume general diamonds
assume general 1/4 tags
```

If you are going to use `with active phantoms`, you must give more detailed assumptions than these two, such as `assume left 1/4 lines` or `assume facing diamonds`.

Use the `with active phantoms` concept with extreme care, if at all. It is extremely dangerous in terms of what it may require the dancers to do. Many dancers find it difficult enough to do the live people's parts of calls without having to do the phantoms' parts also.

You can put `Sd` into a mode in which there is an implicit `with active phantoms` whenever any assumption is made. The `toggle active phantoms` command toggles (turns on or off) this mode. See Section 7.7 [Changing Modes], page 43. When in this mode, `Sd` will not actually insert the text `with active phantoms` after each `assume` concept.

12.9 Assuming a Quarter-Tag

Callers sometimes wish to perform a generalized quarter-tag call (e.g., '`scoot and little`') from a setup in which only the center wave contains real people, and the outsides need to be assumed. For example, from an as-couples wave, one could wish to specify that the live dancers are the as-couples center wave of a quarter-tag. The way to specify this when calling is typically something like '`as couples in a quarter-tag, scoot and little`'.

The proper way to specify this in `Sd` or `Sdtty` is with the `as couples` (or `tandem` or `couples twosome` or whatever) concept, followed by a suitable `assume` concept, such as `assume general 1/4 tags` or `assume right 1/4 tags`. Which of these assumptions you choose doesn't matter as long as it is consistent with the setup you are creating, since you probably won't read the concept verbatim anyway.

For example, from an as-couples wave, you could select:

```
as couples
assume right 1/4 tags
scoot and little
```

Or, from parallel waves, you could select:

```
tandem
assume general 1/4 tags
ping-pong circulate
```

12.10 Designating Certain People

There are six related concepts here. You must have a clear understanding of how the program treats them in order to use them effectively. Compared to the natural and fluent way most dancers and callers use these ideas, the program may seem idiosyncratic and ignorant in its handling of them. See Chapter 9 [Linguistic Idiosyncrasies], page 47.

The six concepts are

```
<ANYONE>
<ANYONE> disconnected
<ANYONE> in your distorted setup
<ANYONE> do your part
ignore the <ANYONE>
own the <ANYONE>
```

In general, be aware that the choice of words and punctuation that the program uses for these concepts is determined by the need to avoid ambiguity, and may not be the words that you should use when calling.

Also, the program is very fussy about using the designators correctly. A large number of designators are provided to help you with this. In addition to the usual `heads`, `sides`, `boys`, `girls`, `head corners`, `centers`, `ends`, `center 2`, `center 6`, `outer 2`, and `outer 6`, there are some that are designed for difficult situations.

`center 4` This is intended for use in a parallelogram or offset lines or columns. It designates the center triple box, or center triple line/column, respectively. Just saying `centers` will not be effective here.

`outer pairs`
: This designates the people that are not in the `center 4`. In a parallelogram, it designates the pairs of people in the outer triple boxes (the "wings"). In offset lines or columns, it designates the the pairs of people in the outer triple lines or columns.

`center diamond`
: This is intended for use in a setup with a wave between, and perpendicular to, two miniwaves. (This is the setup that results from a `1/2 circulate` from waves.) In this setup, `centers` means the center *wave*. If you want the center diamond instead, you must say so.

`center 1x4`
: This is intended for use in things like 3x1 diamonds. (This is the setup that results from a `1/2 acey deucey` from waves.) In this setup, `centers` means the center *diamond*. If you want the center wave instead, you must say `center 1x4`. The program uses the designator `center 1x4` as a catch-all for the center line or the center column. When reading the card, you probably ought to be more specific.

`center 1x6`
: This is similar, but refers to a line or column of 6. It is intended for use in things like 3x1 diamonds, when you want the center wave of 6 to do, for example, a `grand swing thru`. The program will not accept `centers` or `center 6` for this, since those terms are not well-defined in this situation.

12.10.1 <ANYONE>

The concepts are:

```
<ANYONE> <call>
      or
<ANYONE> <call> while the others <other call>
```

If the designated people are centers or ends, the program will do the call(s) according to its best judgement of what the calls mean. Otherwise, it will find the maximal connected undistorted subsets, and do the call in those subsets. For example, from boy-boy-girl-girl waves, the boys are in two miniwaves that have nothing to do with each other. The boys can do 2-person calls, such as trade or hinge, in those setups. If the others are told to do a second call, they do it in their connected undistorted subsets. For example, we could say `boys, hinge while the others shazam`. The results will be reassembled as a 2x4, a C1-phantom setup, stars, or something similar.

From boy-girl-boy-girl waves, the boys are in 4 1-person connected undistorted setups. Using this concept, the boys can quarter right, but they can't trade or hinge.

Now it is commonly accepted practice to say '`boys trade`' when the boys are looking out, and have them effectively trade the wave. This is because the call '`trade`' is a special case, not because calls in general can be done from that setup. To see this, consider the call `roll away`. It, and two-person calls in general, are *not* legal unless the people are adjacent. The special property of the call `trade`, that it can be done by designated people who are not adjacent, is simply an idiosyncrasy that everyone knows. The Sd program recognizes this by having a call in its database `<anyone> trade` as well as the call `trade`. Use that call, not the `<anyone>` concept, to get the appropriate people to trade down the line when in waves.

If you really want non-adjacent people to `roll away`, use the `disconnected` concept.

Incidentally, this is why concepts are capitalized. Otherwise, the line `boys trade` appearing on a printed sequence would be ambiguous. The Sd program is designed to make it possible to determine unambiguously, by looking at the printout, how an action arose.

12.10.2 <ANYONE> Disconnected

The concepts are:

```
<ANYONE> disconnected <call>
      or
<ANYONE> disconnected <call> while the others <other call>
```

This is like `<ANYONE>`, but finds the maximal undistorted subsets, whether they are connected or not. This is a recognized C2 concept. It is generally intended to be used from grand (1x8) setups, in which the designated people leave the room and compress themselves into a 1x4 setup. After doing the call they come back and place themselves in the same four spaces that they had vacated.

Shape-changing calls are permitted. In this case, whichever set of people occupied the centermost spots prior to the call will occupy the center of the result, in accordance with accepted usage for this concept.

12.10.3 <ANYONE> in your Distorted Setup

The concepts are:

```
<ANYONE> in your distorted line <call>
      or
<ANYONE> in your distorted wave <call>
      or
<ANYONE> in your distorted column <call>
      or
<ANYONE> in your distorted diamond <call>
      or
<ANYONE> in your distorted box <call>
```

The designated people are identified in a geometrically distorted setup. If the result of the call has a different shape and orientation from the beginning setup, those dancers may nevertheless be able to go back to the same collective spots, or may be able to maintain the same general location in the total formation.

Be aware that the program requires you to use the most specific concept in each case. The `distorted` concept requires that the setup be geometrically distorted in shape, not just disconnected. Do not say `distorted` when `disconnected` will do, and do not say `disconnected` when just telling the people to do the call will do.

This concept has been set at C2. You can use it at lower levels if you issue the `toggle concept levels` command.

12.10.4 <ANYONE> Do Your Part

The concepts are:

```
<ANYONE> do your part <call>
      or
<ANYONE> do your part <call> while the others <other call>
```

This means that the non-designated dancers leave the room and re-form their own setup in another room. The setups are not shrink-wrapped—they stay the same size in each room, with phantoms where the other people are. The appropriate call is done in each room. The non-designees come back, and the two setups are merged. This merging operation is similar to that used for `<ANYONE>`. It may result in things like C1-phantom setups. From boy-boy-girl-girl waves, `boys do your part, hinge` will leave C1 phantom setups, just as `boys, hinge` will. In fact, the difference between these concepts is rather subtle in many cases. The important point is that, in each room, the dancers work in the entire setup. For example, from facing lines, if the centers do their part of `right and left thru`, they work on their own side, with the nonexistent ends, rather than working in the center.

From boy-girl-boy-girl waves, `boys do your part, hinge` creates a mess. No sensible dancer would consider it acceptable. Do not use such things.

The second form of this concept will be printed in the final transcript as `do your part, <ANYONE> <call> while the others <other call>`. You must still enter it as shown above, with the designator first and the phrase '`do your part`' second.

12.10.5 Ignore the <ANYONE>

The concept is:

```
ignore the <ANYONE>, <call>
```

This means that the designated people do nothing, while the others do the call in the disconnected setup around them. From the standpoint of the non-designated people, the spots occupied by the designated people do not exist.

This concept has been set at C1. You can use it at lower levels if you issue the `toggle concept levels` command.

12.10.6 Own the <ANYONE>

The concept is:

```
own the <ANYONE>, <call> by <other call>
```

This is similar to the `<ANYONE> do your part` concept, except that the result setups are reassembled according to strict matrix positions. That is, instead of a C1 phantom setup, a 4x4 matrix may result. This is a recognized C3A concept.

12.11 12 Matrix and 16 Matrix

When the setup in which a call is to be performed is larger than 8 people due to the presence of phantoms, some modifier is usually required in order to acknowledge the existence of the phantoms' spots. The rules governing the use of such modifiers are not very precisely defined, but Sd attempts to follow accepted usage.

These modifiers are *not* required if some concept has been applied that takes a setup larger than 8 people and forms virtual setups of 8 or fewer people. Examples of such concepts are '`triple boxes`', '`parallelogram`', and '`split phantom waves`'. They are also not required for certain calls that can be done in a setup of any size, such as '`press`', '`truck`', and '`counter rotate`'.

They *are* required whenever a call that is not one of the special ones needs to be extended to more than 8 people, or a setup of more than 8 people needs to be split in order to execute a call in smaller setups.

As an example of the former case, consider offset waves, that is, a 3x4 matrix with the outer waves not fully occupied. If we use a concept like '`offset waves`', '`triple waves`', or '`triple waves working forward`', we can use a call directly. However, if we want to call a 12-person call like '`circulate`' or '`motivate`', we need to legitimize the phantom spots. By saying `12 MATRIX circulate` or `3x4 MATRIX circulate`, we indicate that the phantom spots actually count, and that we want a 12-person version of '`circulate`'. If we are in parallelogram waves, that is, a 2x6 matrix, we can also say `12 MATRIX circulate` or `2x6 MATRIX circulate`. (In this case there are 3 concentric circulate paths of 4 people each.) One could also say `12 MATRIX in roll circulate` from some 2x6 matrices. Exactly what large setups a call can be done from depends on the call.

Also, some calls, like '`circulate`', can be used with either '`12 MATRIX`' or '`3x4 MATRIX`', while others require '`12 MATRIX`'. For example, `12 MATRIX motivate` is legal, but `3x4 MATRIX motivate` is not. The rationale is that '`12 MATRIX`' and '`16 MATRIX`' are call modifiers—`12 MATRIX circulate` and `12 MATRIX motivate` are calls that have definitions in this setup. Concepts like '`3x4 MATRIX`', '`4x4 MATRIX`', '`2x6 MATRIX`', and '`2x8 MATRIX`' simply state how big the setup is. The call '`circulate`' is defined in such a way that it can be executed in a wide variety of sizes and shapes, simply by knowing the size and shape. '`Motivate`' is more restricted. There are calls `12 MATRIX motivate` and `16 MATRIX motivate`, but there is no general rule for all matrices.

The other case in which a matrix size modifier must be given arises when the setup must be split. In setups of 8 or fewer people, it is well accepted that splitting occurs automatically when required. For example, facing lines are split into facing couples when 'flutter wheel' is called. This phenomenon does not automatically happen when the setup is larger than 8 spots.

For example, from parallelogram columns, you can't just say 2x1 transfer the column. You must legitimize the phantom spots by giving a concept such as '12 matrix' or '2x6 matrix'. The setup will then be split into two end-to-end 2x3's, in each of which the call will be performed.

These concepts all have the additional property that they can expand a smaller setup as required, so, for example, '12 MATRIX circulate' is legal in normal (2x4) columns. It expands the setup to a 2x6 be placing pairs of phantoms on the ends. (Special case: in a 2x4 setup, the '16 MATRIX' call modifier doesn't know whether to expand to a 4x4 or a 2x8, so you may need to use '4X4 MATRIX' or '2X8 MATRIX' explicitly. In other cases, the expansion is clear.)

Explicit matrix concepts can also be used to dictate the precise formation with phantoms when using tandem or as-couples concepts. For example, from two-faced lines, '2X8 MATRIX AS COUPLES' leads to virtual 2x4 lines (in which the ends are all phantoms), '4X4 MATRIX AS COUPLES' leads to virtual 2x4 columns (this happens to be the same as 'PHANTOM AS COUPLES' in this setup), and '4X4 MATRIX BOXES WORK SOLID' leads to a virtual 2x2. '2X8 MATRIX BOXES WORK SOLID' leads to a virtual 1x4. From twin diamonds in which each diamond has all 4 people facing the same way, '16 MATRIX OF PARALLEL DIAMONDS DIAMONDSOME' leads to a virtual 1x4 in which both ends are phantoms. 'DIAMONDSOME' leads to a virtual 1x2.

13 Miscellaneous Advice and Warnings

If a call that you believe is legal does not appear on the menu, click on `allow modifications`. This will bring up the universal call menu. You can then thrash out which of you or the computer is correct.

The program sometimes prints non-fatal *warnings* about calls. These are intended principally as advice to you, the user of the program, that the call you have just entered may not be a reasonable thing to call. It is not necessarily the case that reading the warning aloud when calling the sequence is the correct thing to do, though it might be. The call might simply be inappropriate, or it might be wise to say something along the lines of "be careful; everyone do your part." The significance of a warning may also depend on the level. Some things that are extremely peculiar but barely legal at high challenge levels are simply not done at lower levels. You should use your judgement.

Warning: The database is not without errors or misunderstandings in nonstandard uses of calls. The program generally tries to be extremely conservative. However, you should not blindly accept what the program does, particularly if a call was used in an unusual way. The management will not be responsible for any sequences left unattended.

Warning: Some combinations of things that seem obvious to the program might not be agreed to by the dancers. Particularly at high challenge levels, some things are controversial. Proceed with caution. Unless you agree wholeheartedly with what the program did, and believe that the dancers will also agree, it may be best not to do it. Don't stack outrageous interrupts and replacements, with concepts going every which way, unless you are prepared to explain yourself at the end of the tip.

Warning: The program's notion of levels is only keyed to concepts and calls, not to concept/call combinations, and not to calls in the context of certain setups. For example, since *split* and *square thru* are both Mainstream, it thinks that *split square thru* is Mainstream, even though it is A1. Also, *chain reaction* should only be used from the simple '`heads pass the ocean`' setup at A1, but the program doesn't know this.

For debugging, there is a hidden level called `all`, which is above C4. At this level, even the invisible calls that are used in the definitions of other calls are legal. They typically begin with an underscore. Also, when the `all` level is selected, all sequentially-defined calls are fractionalizable, even if they aren't in practice.

The font for the text output window must be fixed-width so the formation pictures are legible. If you do not get this by default, set this X resource:

```
Sd*text*font: -misc-fixed-medium-r-normal--13-*-*-*-c-70-iso8859-1
```

14 Customization with X Resources

The file '`Sd.res`' is a sample X user resource file for the program. It contains values for a number of parameters, including most of the text on the screen which is not fetched from the calls database. Install '`Sd.res`' as '`$HOME/Sd`' to use these resources.

People who have many user resource files for different applications may wish to store them in a directory other than their home directory. To do so, set the environment variable `XAPPLRESDIR` to the directory where your resource files are. For example, one of the authors uses the subdirectory '`app-resources`', and his .cshrc contains this line:

```
setenv XAPPLRESDIR ~/app-resources
```

To use a different value for any of these parameters, either edit the resource file or load the changed value into your X server's resource database. To modify the server resource database, put a modified version of the appropriate line in the '`.Xresources`' file in your home directory (assuming you run `xrdb .Xresources` from '`.xsession`' or whatever your X startup file is called). You will have to re-run `xrdb` after editing your '`.Xresources`' file. You should add '`Sd`' to the front of any lines you move to your '`.Xresources`' file.

All values in the distributed '`Sd.res`' file except the colors are the defaults and are in the file for illustration only. You can remove them if you wish your installed user resource file to contain only the resources you change.

To have mouse button 3 invoke the Undo operation, as one of the authors does, set these resources:

```
Sd*frame*Label*Translations: #override <Btn3Down>: undo()
Sd*frame*List*Translations: #override <Btn3Down>: undo()
```

15 Versions

The version is a string consisting of parts separated by colons.

If you customize any part of Sd for your own use, or fix bugs, you should assign the result a new version by appending to the current version. This will help in keeping track of bug reports. The version string is defined near the top of the file ‘`sdmain.c`’. For example, you might set it to `25.6.joe1` for your private version.

The calls database also has a version. If you modify the database, you should change the version string at the top of the file.