Sd

A Square Dance Caller's Helper

William B. Ackerman and Stephen Gildea

This manual documents Sd version 27.

Introduction

Sd is a square dance caller's helper. The program assists in writing material 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 was intended to be used to write challenge-level material where the sequences are often complex and the checker pushing tedious and error-prone. Most challenge callers write out the material they will call before they get to the dance, unlike Mainstream callers, who often invent the material on the fly from the stage.

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

This program is not appropriate for Traditional-style squares. Designing the exquisitely timed and well-phrased sequences that make up Traditional squares may or may not be beyond the capability of computer programs, but it is definitely beyond the capability of this program.

Since the emphasis here is on checker pushing, you will find various mainstream staples missing, such as Circle Left, Grand Square, and Allemande Left. Since these calls are technical zeros, there is no reason to have them in a checker-pushing program. However, if you wish to write material 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.

1 Starting Sd

The program is invoked with a single argument—the level. This is one of `m` (mainstream), `p` (plus), `a1`, `a2`, `c1`, `c2`, `c3a`, `c3`, `c3x`, `c4a`, or `c4`. The default is `m`. This 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, call list customization, and 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. This takes about (200/MIPS) seconds at C4, correspondingly less at lower levels.

1.1 Command Switches

Sd accepts all Xt command-line options. The following are some of the more useful options for use with Sd:

`-rv` reverse video

`-bg` background color

`-fg` foreground color

`-bd` border color

`-font` font used everywhere

`-geometry`
geometry in pixels

`-title` title for window manager and icon use

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

`-xrm` an X resource manager string.

Additional options specific to this program:

`-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`.

There are X resources associated with both switches: `Sd.sequenceFile` is equivalent to `-sequence`, 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.

1.2 Abridgement

In addition to the calling level and any X Window System arguments on the command line, the program recognizes three special flags:

`-write_list` *`filename`*
: write out the call list for the indicated level instead of running the program.

`-write_full_list` *`filename`*
: write out the call list for the indicated level and all lower levels instead of running 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)`' after its level.

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

```
sd c2 -write_list my_group
```

Then delete from the file `my_group` those calls that the group has learned. When writing material, use the command

```
sd c2 -abridge my_group
```

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.

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 is not important, though it is possible that leaving it in the original menu order may someday yield faster program startup.

2 Calling

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

2.1 The Startup Screen

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 obvious program exit button, this permits you to select the starting operation. The selections here, and almost everywhere else in the program, are made by moving the mouse until the cursor is over the chosen item, and pressing Mouse Button 1 (typically the left button).

2.1.1 Heads/Sides Start

If you click on `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 click on `heads start`, and then `star thru`, and then `double pass thru`, and so on. You can also click on `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.

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 click on `centers` before all calls after the initial one. Whether you actually read the card that way while calling is up to you.

2.1.2 1P2P Lines

The `heads 1P2P` and `sides 1P2P` starting actions refer to caller jargon for a common starting maneuver, typified by *so-and-so 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.

2.1.3 Just As They Are

The `just as they are` button starts the sequence on squared set spots, without having the heads or sides begin. This is nearly useless, since the program doesn't know about the *all 4 couples* concept or about thars or alamo rings. This shortcoming may be fixed in the future.

2.2 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.)

2.2.1 Call Menus

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 `<N>`, `<ANYONE>`, or `<ANYTHING>` items. Hence, `<ANYONE> run` will be found under `r`. Digits are listed first, followed by upper case letters, followed by lower case letters.

A number of calls have limitations or quirks that you should be aware of. See Chapter 5 [Call Notes], page 13.

2.2.2 The Scroll Bar

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.

2.2.3 Variations on Calls

Some calls require a numeric designator (e.g., *square thru 3*) or a people designator (e.g., *boys run*) or another call (e.g., *clover and star thru*). Such designators appear in the call menu as `<N>` or `<ANYONE>` or `<ANYTHING>`. When you select one of the first two kinds of calls, a popup will appear containing numbers or people designators. Click on the chosen item. Not all items will be legal in all cases, such as `leads run` from a grand line. If you made a mistake, just move the mouse away from the popup.

When you select a call that requires a subordinate call, the program will prompt you for it.

Some calls or concepts may require more than one numeric argument. If this is the case, numeric popups will appear more than once. Enter the numbers in sequence.

Some calls or concepts require a number that is naturally expressed as a fraction indicating some number of quarters, such as touch 1/4, 1/2, or 3/4. Enter the integer number of quarters. The program will turn the fraction into the appropriate form, e.g., 2/4 becomes 1/2.

2.3 Entering Concepts

A call may be preceded by concepts. These may be nested (*stacked*) to any reasonable depth. 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. The submenus are

`phantom concepts`
: those that involve picking out virtual setups, that may include phantoms, from a real formation that is larger than 8 people. Examples: *split phantom lines*, *triple boxes*, *triple lines working forward*.

`couples/tandem concepts`
: the obvious thing.

`distorted concepts`
: those that involve picking out distorted virtual setups, without phantoms, by identifying real people out of a formation that is larger than 8 people, or the phantom equivalent of same. Examples: *parallelogram*, *stairstep lines*, *bigblock*.

`4-person distorted concepts`
: those that involve picking out 4-person virtual setups, without phantoms, by identifying real people, or the phantom equivalent of same. Examples: *trapezoid*, *once removed*, *jay*.

`miscellaneous concepts`
: all others. There are a lot of them.

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.

Some concepts require two calls (e.g., *checkpoint* and *interlace*). 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.

2.4 Call Modifications

Some calls can be modified in a natural way, such as '`in roll motivate`' 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, click on `allow modifications` before choosing the call. There are two levels of this feature. If you click on `allow modifications` just once, 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` a second time, you allow a potentially large number of modification possibilities, sometimes several in the same call.

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 so 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?` Moving the mouse away constitutes a '`no`' reply. If you click in the active area of the popup, 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 modification. For example, in *motivate*, you can replace the initial *circulate* and can turn the star a different amount. You may select any or all of these modifications.

For both these optional modifications and the mandatory subcalls (e.g., `clover and <anything>`) discussed earlier, 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.

2.5 Tag/Scoot Call Modifications

To use tagging call modifications, like '`tag the star your leader`', enable modifications, click on `tag your leader` (the generic form of all tagging calls is '`tag ...`') followed by the '`1/2`' version of the replacement tagging call, e.g., `1/2 tag the star` or `loop and 1/2 tag`. The program will remove the superfluous '`1/2`'. You may also choose the `revert` or `reflect` calls here.

To get something like '`flip chain thru and scatter reaction`', enable modifications and select `tag chain thru and scatter reaction` or whatever. Then change the `1/2 tag`.

`1/2 invert` is an acceptable tagging replacement.

2.6 The *Single, Cross,* and *Left* Modifiers

There are a number of calls that contain these words in a nonstraightforward way. In these cases, click on the concept before clicking on the call, even though the word order seems wrong. The program will repair the word order.

Examples:

Chase left or *left roll the*
: Click on `LEFT` and then `chase right` or `right roll the`.

Scoot and cross ramble, trans cross reactivate, etc.
: Click on `CROSS` and then the call. The program will move the word '`cross`' into the correct place in the transcript. For example, click on `CROSS` and then `scoot and ramble` to get *scoot and cross ramble*. Click on `CROSS` and then `transactivate` to get *trans cross reactivate*. Click on `MAGIC`, `CROSS` and then `trans nuclear reaction` to get *magic trans cross nuclear reaction*.

Some calls use a '`single file`' modifier. Click on `SINGLE` to get this. The program will change the wording to '`single file`'. Note in particular that the call described by some callers as *on a double track, dixie style* (even when there is only one *track*) is known to the program as *single file dixie style*.

Not all calls that have the word '`cross`' or '`single`' in them are invoked by clicking on the concept. Some, like '`cross your neighbor`' and '`single rotate`' are simply in the

database under their full names. Note in particular that the '`single`' concept for '`rotate`' gets the call '`single file rotate`', which is a 1x4 column call.

2.7 Asymmetric Selectors

The people selector popup contains various asymmetric selectors such as `near line` and `far box`. You can use these selectors with the `so-and-so only` concept to call *near column pass thru*, etc. The resultant setup must be reasonable—no shape-changers on one side of the set. Most calls have not been checked for robustness in the presence of asymmetrical setups. Use this with care! The resolver checks all 8 people, so it should work. If you have not restored symmetry, resolves may be extraordinarily difficult to find.

3 Resolving and Searching

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

3.1 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*, *dixie grand*, or *promenade* from 8-chain, trade by, double pass thru, waves, crossed waves, 1/4 tag, 3/4 tag, and left-hand two-faced lines, as appropriate. If the sequence needs an *extend*, *circulate*, *pass thru*, *trade by*, or *cross by* to get to a resolve, that is close enough.

Whenever the '`resolve`' message appears, you can end the sequence and write it to a file by clicking on `end this sequence`. 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 click on `resolve`, the program goes into a special mode in which it searches for resolves, saves them, and lets you look through them and pick one that you like. Up to 50 resolves may be stored while it is doing this. A resolve 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 resolve. That information tells how many resolves are currently stored and which one is currently shown. When you click on `resolve`, the program finds the first resolve and displays it, showing information '`1 out of 1`', which means that it has one resolve stored, and resolve number one is currently displayed. The transcript area shows the effect of that resolve.

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

Clicking on `accept` will leave resolve mode, causing the current resolve to be added to the sequence exactly as though you had entered those calls manually. You can then `undo` individual calls if you wish.

Clicking on `abort the search` will throw away all of the saved resolves and leave resolve mode, but will not destroy the sequence. The sequence will be left just as it was before `resolve` was selected.

Clicking on anything else, such as `exit`, `end this sequence`, or a concept, is equivalent to `accept` followed by whatever that action is. So, for example, you can click on `end this sequence` as soon as you see a resolve that you like.

If one or more concepts are pending when you select `resolve`, the program will search only for resolves whose first call starts with those concepts. So, for example, clicking on `once removed` and then `resolve` might get you this resolve:

```
ONCE REMOVED reverse the pass
```

```
linear flow
right and left grand  (7/8 promenade)
```

The program searches for resolves 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 (1 call) rather than long ones (3 calls) and in favor of resolves that do not contain *all 8 circulate*. If, after 5000 attempts, no resolve is found, the `find another` operation fails. (This tends to happen if you try to resolve out of an hourglass at mainstream.) You can click on `find another` again to make another attempt if you wish.

3.2 Making Nice Setups

Clicking on `nice setup` invokes an operation very similar to the resolver, except that it searches for sequences of up to three calls that normalize a 4x4 matrix into a 2x4. It uses *split phantom lines* and similar concepts to do its work.

The program does not yet have the ability to perform other normalization tasks, such as using the *triple box* or *triple line* concepts to get out of various 12 matrices.

3.3 The `Do Anything` operation

Clicking on `do anything` invokes an operation very similar to the resolver, except that it searches 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 cute uses of difficult concepts such as *checkpoint*, *trace*, or *on your own*.

3.4 The `Reconcile` operation

Clicking on `reconcile` invokes an operation like resolving, but it puts the generated calls someplace other than at the end. 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 give the program a clue. 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. Click on `raise insertion point` or `lower insertion 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, click on `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 3 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.

4 Miscellaneous Commands

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

4.1 Writing Out the Sequence

At any time when the sequence is resolved (for example, after a successful use of the `resolve` operation), you can click on `end this sequence`. This will append the current sequence to the current output file, and then go back to the startup screen. 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 versions of the program and database.

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

4.2 Changing the Output File

TBA

4.3 `Abort`, `Exit`, and `Undo`

If you click on the `exit` button, the program will exit. If a sequence is in progress, the program will ask for confirmation first. Using your window manager to send a `Delete Window` message to the program is equivalent to clicking on the `exit` button. The `exit` button appears on the startup screen as well as the normal screen.

If you click on the `abort` button, the program will abort the sequence but not exit. It will go back to the startup screen to allow you to start another sequence. It will ask for confirmation first.

If you click on the `undo` button 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.

4.4 Inserting Comments

TBA

4.5 Saving Pictures

The program shows pictures in the transcript for the current position and the position just before the last call. Normally, those pictures will not appear in the sequence written to the file. If you want the current position to have its picture written in the 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") click on `save picture`.

5 Call Notes

5.1 *Box the Gnat*

Box the gnat is equivalent to *touch 1/2*. Common usage of this call leaves hands sort of joined. If you call *box the gnat* followed by *hinge*, there will be trouble.

5.2 Sweeping Direction

Sweep 1/4, *with the flow*, and *by golly* all assume a sweeping direction. The program does not currently remember sweeping direction from the preceding call. Hence, it assumes clockwise. Click on `REVERSE` if you want it the other way.

5.3 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.

5.4 Spread

'`Spread`' is the call you use after *follow your neighbor*. '`And spread`' is the one you use after *wheel and deal*, or after starting a sequence with something like '`heads star thru`'.

5.5 Colliding *Recycle*

"Colliding" *recycle*, *ah so*, and *cycle and wheel* are not implemented. Do not call these if the ends of the line are facing the same way. *Wheel and deal* from a one-faced line is okay.

5.6 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.

5.7 <*anything*> *the Yellow Bricking* <*anything*>

To use the second `<anything>` in <*anything*> *the yellow bricking* <*anything*>, as in *follow the yellow bricking turn and deal*, enable simple modifications and click on `follow the yellow brick road`.

To use the first `<anything>`, as in *revert cross loop and tag the yellow brick road*, click on `tag the yellow brick road`. If simple modifications are enabled, you will be able to turn the *1/2 tag* into whatever tagging variation you want. The philosophy is the same as for things like *tagger's dilemma*.

If you want to modify both, you must enable all modifications (click twice on `allow modifications`), do `tag the yellow brick road`, and use the second modification. It will come out looking ugly. Sorry.

6 Concept Notes

6.1 Gruesome

In this program, *gruesome* simply means *phantom couples or tandem* with the phantoms added to make a 2x8 rather than the 4x4 that would normally be created, and with the added requirement that the pairing be parallel to the long axis. *Gruesome twosome* is just an abbreviation for *gruesome as couples twosome.*

6.2 Divided

The concepts '`divided lines`' and '`divided columns`', done from a 2x8, and '`12 matrix divided lines`' and '`12 matrix divided columns`', done from a 2x6, mean that the setup is to be separated into 2x4's or 2x3's, respectively. The former two are equivalent to *split phantom twin boxes*, except that they demand that people be facing in the indicated way. For all of these concepts, if people are T-boned, you can use `standard` to permit the concept to be used. 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.

6.3 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 "line-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 or whatever, 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 of *their own* 1x4 or diamond, not the long axis of the set. Yes, I realize that this is controversial and may not be in accordance with Callerlab definitions.

If the concept was *concentric*, so these people started on the outside in a 1x4 or diamond, their own long axis is presumably 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, their own long axis might not be the long axis of the set. If we do *cross concentric square thru 2* from a crossedwave, so everyone rears back, the original centers finish perpendicular to their own 1x4, not the original ends' 1x4. They finish in approximately the places where the ends began. If we do it from a 1/4 tag, the centers finish where the ends began.

6.4 Two Calls in Succession

The `do two calls in succession` 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. The script will appear as above. 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. Or you could just read the parentheses.

6.5 Diamond

You should not need to click on `diamond` except when you actually want the diamond concept, as in *diamond single wheel* or *diamond swing thru*. There are cases in which it may seem that the word *diamond* ought to be added after clicking on `interlocked` and/or `magic`. The program will insert the extra word *diamond* for you. (At least that is the intention.) So, for example, clicking on `magic` and then `alter the diamond` will get '`magic diamond, alter the diamond`', and clicking on `magic`, `interlocked`, `diamond`, `as couples`, and `quarter right` will get '`magic interlocked diamond, diamond as couples quarter right`'.

6.6 Implied Piecewise

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. Furthermore, replacements and interruptions are *normal*, that is, they do not carry any concepts that were on the call being replaced but not 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 3rd 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]
```

6.7 So-And-So Only

This concept means that the designees do their part of the call in the complete setup, as if it were '`own the` *`so-and-so`* `for` *`call`* `by nothing`'. It does not do the call in their disconnected setup, even though that is a common interpretation.

7 Known Bugs and Misfeatures

The *funny* concept is not fully implemented. Only *funny circulate* types of calls are supported, as in *funny go first class* or *funny bingo*. Do not attempt a *funny square thru* or *funny slide thru*.

The behavior of the *ends* concept is far too lenient. It will allow the ends of a 2x4 to do any 2x2 call, ignorant of the fact that real people will object to having the centers in the way. For example, from a column or DPT, it will allow the ends to circulate. Don't do this.

This chapter is very incomplete.

8 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.

Warning: The database is not without errors or misunderstandings in nonstandard uses of calls. The program generally tries to be extremely conservative, its author being aware of how annoying it is to have to sight out of what seemed to start out as a beautiful card. 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.

Table of Contents