DMC Call Interface Specification

This is the API specification for the DMC (DELTA Multiple Criteria) interface layer. It is intended to run on several platforms including Unix, PC/Windows, and Macintosh.

The DMC interface is a glue layer to DeltaLib/DADE (DELTA Analytical Decision Engine). DeltaLib is a general-purpose library for the evaluation of decision frames based on the author's Ph.D. thesis *Computational Decision Analysis*.

The main extensions in DMC compared to the thesis are connected to the introduction of weight bases containing criteria weights. The weight bases are similar but not equal to probability bases. In addition, they necessitate the solving of tri-linear systems of inequalities (criteria weights, probabilities and values) whereas the algorithms in the thesis solve bi-linear systems (probabilities and values). See Chapter 6 in the thesis for a description of the bi-linear algorithms used in DMC both for consistency checks and for the calculation of expected values in interval decision analysis.

The DMC commands are divided into seven groups: system, structure, weight, probability, utility, evaluation, and miscellaneous.

Algorithms and implementation by

Dr. Mats Danielson
Dept. of Computer and Systems Sciences
Stockholm University
Forum 100
SE-164 40 Kista
Sweden
Email: mad@dsv.su.se

The DMC and DeltaLib libraries are today (2002-12-01) handed over to the company Doctor Decide AB for further development. This marks the end of the open source access to the libraries.

Contents

SYSTEM COMMANDS

Start DMC layer

Call syntax: DMC_init(MAX_CRIT,MAX_ALTS,MAX_CONS,MAX_COPA,MAX_STMTS)

```
Return information:
OK    -
ERROR – license expired
        wrong release
        config mismatch
```

Call semantics: Perform initialization of Delta resources and start the DMC layer. The call parameters MAX_CRIT, MAX_ALTS, MAX_CONS, MAX_COPA, MAX_STMTS must be exactly as defined in delta_extract.h and dmc.h. This must be the first call to DMC.

Stop DMC layer

Call syntax: DMC_exit()

```
Return information:
OK    - elapsed time in minutes
ERROR – memory leak
```

Call semantics: Release resources in Delta and DMC. Should be the last call to DMC.

STRUCTURE COMMANDS

Create new frame

Call syntax: DMC_new_PS_frame(const char *name, int n_alts, int n_cons[])

```
Return information:
OK    - frame number
ERROR - input error
        frame corrupted
        no core memory
        too many alternatives
        too many consequences
        out of frames
```

Call semantics: Creates a new probabilistic frame called ‘name’ with one criterion and an initial structure as specified in the call. If name is an empty string, the frame will be called NoName but the name string will not be updated. A frame cannot have less than two alternatives at any point. Each alternative must have at least one consequence. The frame is not loaded and can be filled with data prior to loading.

Call syntax: DMC_new_DM_frame(const char *name, int n_crit, int n_alts)

```
Return information:
OK    - frame number
ERROR - input error
        frame corrupted
        no core memory
        too many alternatives
```

too many consequences

Call semantics: Creates a new deterministic frame called 'name' with n_crit criteria and n_alts alternatives as specified in the call. If name is an empty string, the frame will be called NoName but the name string will not be updated. A frame can not have less than two alternatives at any point. The alternatives have exactly one consequence under each criterion. The frame is not loaded and can be filled with data prior to loading.

Dispose frame

Call syntax: DMC_dispose_frame(int ufnr)

Return information:
OK -
ERROR – no such frame
frame not loaded
frame corrupt

Call semantics: Dispose resources belonging to frame 'ufnr' and free the slot for a new frame. Note: frames can only be disposed when no frame is open.

Get frame name

Call syntax: DMC_frame_name(char *fname)

Return information:
OK -
ERROR – frame not loaded

Call semantics: Returns the name of the current frame.

Load frame

Call syntax: DMC_load_frame(int ufnr)

Return information:
OK -
ERROR - no such frame
frame corrupted
no core memory
frame in use
inconsistent

Call semantics: Attempts to attach the frame 'ufnr' to Delta. Both bases are loaded and checked for consistency. If any base is inconsistent, the frame will not be attached.

Activate frame

Call syntax: DMC_activate_frame(int ufnr)

Return information:
OK -
ERROR - no such frame
frame corrupted
no core memory
frame in use

Call semantics: Attempts to activate the frame 'ufnr' without attaching it to Delta. The bases are not checked for consistency. It is possible to add statements to an activated base without consistency checks. This is used for loading already consistent sets of statements.

Close frame

Call syntax: DMC_unload_frame()

Return information:
OK   -
ERROR - frame not loaded

Call semantics: Detach the frame from Delta and free the interface for new frames. Note: in case of internal problems in Delta, the frame might be detached without an explicit call to DMC_unload_frame.

Add an alternative to an existing frame

Call syntax: DMC_add_alternative(int n_cons[])

Return information:
OK   - alt-number
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        too many consequences
        too many alternatives

Call semantics: Append a new alternative with n_cons[i] consequences for the i:th criterion to the frame. The new alternative receives the alt-number n+1, where n is the previous number of alternatives in the frame.

Delete an alternative in a frame

Call syntax: DMC_delete_alternative(int alt)

Return information:
OK   - number of alternatives remaining
ERROR - input error
        frame corrupted
        frame not loaded
        referenced

Call semantics: Delete alternative 'alt' from the frame. All alternatives with higher numbers within the frame are shifted one position down.

Add a criterion to an existing frame

Call syntax: DMC_add_criterion()

Return information:
OK   - alt-number
ERROR - wrong frame type
        no core memory
        frame corrupted

```
        frame not loaded
        too many consequences
```

Call semantics: Append a new criterion to the frame. The new criterion receives the crit-number n+1, where n is the previous number of criteria in the frame.

Delete a criterion in a frame

Call syntax: DMC_delete_criterion(int crit)

```
Return information:
OK    - number of alternatives remaining
ERROR - input error
        frame corrupted
        frame not loaded
        wrong frame type
        referenced
```

Call semantics: Delete criterion 'crit' from the frame. All criteria with higher numbers within the frame are shifted one position down.

Add a consequence

Call syntax: DMC_add_consequence(int crit, int alt)

```
Return information:
OK    - consequence-number within the consequence set
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        too many consequences
```

Call semantics: Append a new consequence to the consequence set. The new consequence receives the number n+1, where n is the previous number of consequences in the set.

Delete a consequence

Call syntax: DMC_delete_consequence(int crit, int alt, int cons)

```
Return information:
OK    - number of consequences remaining in the consequence set
ERROR - input error
        frame corrupted
        frame not loaded
        wrong frame type
        referenced
```

Call semantics: Delete consequence 'cons' from consequence set 'crit alt'. All consequences with higher numbers within the set are shifted one position up.

WEIGHT COMMANDS

Add a weight interval statement

Call syntax: DMC_add_W_statement(struct user_stmt_rec* ustmtp)

```
Return information:
OK    - statement number in the weight base
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        too many statements
        too narrow statement
        inconsistent
```

Call semantics: Add the user weight statement w(crit) = [lobo,upbo] to the weight base within the decision frame. If the base is loaded, it is checked for consistency with respect to the new interval. In case of inconsistency, nothing is added to the base, while for adding to a non-loaded base, the statement is added without consistency checks (from version 1.5). The latter is only for loading preconsistent data.

Add a weight link statement

Call syntax: DMC_add_W_statement(struct user_stmt_rec* ustmtp)

```
Return information:
OK    - statement number in the weight base
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        too many statements
        too narrow statement
        inconsistent
```

Call semantics: Add the user weight link w(crit1) - w(crit2) = [lobo,upbo] to the weight base within the decision frame. The base is checked for consistency with respect to the new link. In case of inconsistency, nothing is added to the base, while for adding to a non-loaded base, the statement is added without consistency checks (from version 2.06). The latter is only for loading preconsistent data.

Change the bounds of a weight statement

Call syntax: DMC_change_W_statement(int stmt_number, real lobo, real upbo)

```
Return information:
OK    -
ERROR - input error
        frame corrupted
        frame not loaded
        wrong frame type
        too narrow statement
        inconsistent
```

Call semantics: Change the existing user weight statement w(crit) = [old_lobo,old_upbo] or w(crit1) - w(crit2) = [old_lobo,old_upbo] to w(crit) = [lobo,upbo] or w(crit1) - w(crit2) = [lobo,upbo] in the weight base. The base is checked for consistency with respect to the change. In case of inconsistency, nothing is changed the base and the call is rolled back.

Replace a weight statement with an interval statement

Call syntax:
DMC_replace_W_statement(int stmt_number, struct user_stmt_rec* ustmtp)

Return information:
OK -
ERROR - input error
no core memory
frame corrupted
frame not loaded
wrong frame type
too narrow statement
inconsistent

Call semantics: Replace the user weight statement (interval or link) with w(crit) = [lobo,upbo] in the weight base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base and the call is rolled back.

Replace a weight statement with a link statement

Call syntax:
DMC_replace_W_statement(int stmt_number, struct user_stmt_rec* ustmtp)

Return information:
OK -
ERROR - input error
no core memory
frame corrupted
frame not loaded
wrong frame type
too narrow statement
inconsistent

Call semantics: Replace the user weight statement (interval or link) with w(crit1) - w(crit2) = [lobo,upbo] in the weight base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base and the call is rolled back.

Delete a weight statement

Call syntax: DMC_delete_W_statement(int stmt_number)

Return information:
OK - number of statements remaining in the weight base
ERROR - input error
frame corrupted
frame not loaded
wrong frame type

Call semantics: The user weight statement (interval or link) with position stmt_number in the weight base is deleted from the base. All statements with higher positions within the base are shifted one position down.

Make a weight estimate

Call syntax: DMC_make_W_estimate(struct user_stmt_rec* ustmtp)

Return information:
OK -
ERROR - input error
no core memory
frame corrupted
frame not loaded
wrong frame type
inconsistent

Call semantics: Add the user weight estimate w(crit) = [lobo,upbo] to the weight base within the decision frame. If the base is loaded, it is checked for consistency with respect to the new interval. In case of inconsistency, nothing is added to the base, while for adding a non-loaded base, the statement is added without consistency checks. The latter is only intended for loading pre-consistent data.

Remove a weight estimate

Call syntax: DMC_remove_W_estimate(struct user_stmt_rec* ustmtp)

Return information:
OK -
ERROR - input error
frame corrupted
frame not loaded
wrong frame type

Call semantics: The user weight estimate w(crit) = [lobo,upbo] in the weight base is deleted from the base. This can be considered unlocking the estimate.

Add weight sensitivity

Call syntax: DMC_add_W_sensitivity(h_matrix lobo, h_matrix upbo)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded
inconsistent

Call semantics: The hull is replaced by a user determined orthogonal hyper-cube in order to facilitate manual sensitivity analyses (as opposed to the automatic contraction procedure).

Remove weight sensitivity

Call syntax: DMC_remove_W_sensitivity()

Return information:
OK -

```
ERROR - wrong frame type
        frame corrupted
        frame not loaded
```

Call semantics: The user determined orthogonal hypercube is replaced by the ordinary hull.

Get the weight hull

Call syntax: DMC_get_W_hull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

```
Return information:
OK    -
ERROR - wrong frame type
        frame corrupted
        frame not loaded
```

Call semantics: The hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons].

Get the symmetric weight hull

Call syntax: DMC_get_W_symhull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

```
Return information:
OK    -
ERROR - wrong frame type
        frame corrupted
        frame not loaded
```

Call semantics: The symmetric hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons].

Get the weight core

Call syntax: DMC_get_W_core(h_matrix loco, h_matrix upco)

```
Return information:
OK    -
ERROR - wrong frame type
        frame corrupted
        frame not loaded
```

Call semantics: The core is returned in two matrices loco and upco indexed as [crit][alt][cons].

Get the weight intersection

Call syntax: DMC_get_W_intersection(h_matrix lobo, h_matrix upbo)

```
Return information:
OK    -
ERROR - frame corrupted
        frame not loaded
        no intersection
        output error
```

Call semantics: The call must be preceded by an evaluation. The weight base contraction at the evaluation graph's intersection with the contraction axis is returned in two matrices lobo and upbo indexed as [crit][alt][cons].

PROBABILITY COMMANDS

Add a probability interval statement

Call syntax: DMC_add_P_statement(struct user_stmt_rec* ustmtp)

Return information:
OK - statement number in the probability base
ERROR - input error
no core memory
frame corrupted
frame not loaded
wrong frame type
too many statements
too narrow statement
inconsistent

Call semantics: Add the user probability statement p(crit:alt:cons) = [lobo,upbo] to the probability base within the decision frame. If the base is loaded, it is checked for consistency with respect to the new interval. In case of inconsistency, nothing is added to the base, while for adding to a non-loaded base, the statement is added without consistency checks (version 2.06). The latter is only for loading preconsistent data. NOTE: lobo and upbo are local probabilities.

Add a probability link statement

Call syntax: DMC_add_P_statement(struct user_stmt_rec* ustmtp)

Return information:
OK - statement number in the probability base
ERROR - input error
no core memory
frame corrupted
frame not loaded
wrong frame type
too many statements
too narrow statement
inconsistent

Call semantics: Add the user probability link p(crit1:alt1:cons1) - p(crit2:alt2:cons2) = [lobo,upbo] to the probability base within the decision frame. The base is checked for consistency with respect to the new link. In case of inconsistency, nothing is added to the base, while for adding to a non-loaded base, the statement is added without consistency checks (version 1.5). The latter is only for loading preconsistent data. NOTE: lobo and upbo are local probabilities.

Change the bounds of a probability statement

Call syntax: DMC_change_P_statement(int stmt_number, real lobo, real upbo)

Return information:
OK -
ERROR - input error

```
        frame corrupted
        frame not loaded
        wrong frame type
        too narrow statement
        inconsistent
```

Call semantics: Change the existing user probability statement p(crit:alt:cons) = [old_lobo,old_upbo] or p(crit1:alt1:cons1) - p(crit2:alt2:cons2) = [old_lobo,old_upbo] to p(crit:alt:cons) = [lobo,upbo] or p(crit1:alt1:cons1) - p(crit2:alt2:cons2) = [lobo,upbo] in the probability base. The base is checked for consistency with respect to the change. In case of inconsistency, nothing is changed the base. NOTE: lobo and upbo are local probabilities.

Replace a probability statement with an interval statement

Call syntax:
```
DMC_replace_P_statement(int stmt_number, struct user_stmt_rec* ustmtp)
```

Return information:
```
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        too narrow statement
        inconsistent
```

Call semantics: Replace the user probability statement (interval or link) with p(crit:alt:cons) = [lobo,upbo] in the probability base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base. NOTE: lobo and upbo are local probabilities.

Replace a probability statement with a link statement

Call syntax:
```
DMC_replace_P_statement(int stmt_number, struct user_stmt_rec* ustmtp)
```

Return information:
```
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        too narrow statement
        inconsistent
```

Call semantics: Replace the user probability statement (interval or link) with p(crit1:alt1:cons1) - p(crit2:alt2:cons2) = [lobo,upbo] in the probability base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base. NOTE: lobo and upbo are local probabilities.

Delete a probability statement

Call syntax: `DMC_delete_P_statement(int stmt_number)`

```
Return information:
OK    - number of statements remaining in the probability base
ERROR - input error
        frame corrupted
        frame not loaded
        wrong frame type
```

Call semantics: The user probability statement (interval or link) with position stmt_nbr in the probability base is deleted from the base. All statements with higher positions within the base are shifted one position down.

Make a probability estimate

Call syntax: DMC_make_P_estimate(struct user_stmt_rec* ustmtp)

```
Return information:
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        wrong frame type
        inconsistent
```

Call semantics: Add the user probability estimate p(crit:alt:cons) = [lobo,upbo] to the probability base within the decision frame. If the base is loaded, it is checked for consistency with respect to the new interval. In case of inconsistency, nothing is added to the base, while for adding a non-loaded base, the statement is added without consistency checks. The latter is only for loading pre-consistent data. NOTE: lobo and upbo are local probabilities.

Remove a probability estimate

Call syntax: DMC_remove_P_estimate(struct user_stmt_rec* ustmtp)

```
Return information:
OK    -
ERROR - input error
        frame corrupted
        wrong frame type
        frame not loaded
```

Call semantics: The user probability estimate for the consequence 'crit alt cons' in the probability base is deleted from the base. This can be considered unlocking the estimate.

Add probability sensitivity

Call syntax: DMC_add_P_sensitivity(h_matrix lobo, h_matrix upbo)

```
Return information:
OK    -
ERROR - wrong frame type
        frame corrupted
        frame not loaded
        inconsistent
```

Call semantics: The hull is replaced by a user determined orthogonal hyper-cube in order to facilitate manual sensitivity analyses (as opposed to the automatic contraction procedure). NOTE: lobo and upbo are local probabilities.

Remove probability sensitivity

Call syntax: DMC_remove_P_sensitivity()

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded

Call semantics: The user determined orthogonal hypercube is replaced by the ordinary hull.

Get the global probability hull

Call syntax: DMC_get_P_hull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded
too many consequences

Call semantics: The hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons]. NOTE: lobo and upbo are global probabilities.

Get the local probability hull

Call syntax: DMC_get_LP_hull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded
too many consequences

Call semantics: The hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons]. NOTE: lobo and upbo are local probabilities.

Get the global symmetric probability hull

Call syntax: DMC_get_P_symhull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded

too many consequences

Call semantics: The hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons]. NOTE: lobo and upbo are global probabilities.

Get the global probability core

Call syntax: DMC_get_P_core(h_matrix loco, h_matrix upco)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded

Call semantics: The core is returned in two matrices loco and upco indexed as [crit][alt][cons]. NOTE: lobo and upbo are global probabilities.

Get the local probability core

Call syntax: DMC_get_LP_core(h_matrix loco, h_matrix upco)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded

Call semantics: The core is returned in two matrices loco and upco indexed as [crit][alt][cons]. NOTE: lobo and upbo are local probabilities.

Get the probability intersection

Call syntax: DMC_get_P_intersection(h_matrix lobo, h_matrix upbo)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
no intersection
output error

Call semantics: The call must be preceded by an evaluation. The probability base contraction at the evaluation graph's intersection with the contraction axis is returned in two matrices lobo and upbo indexed as [crit][alt][cons]. NOTE: lobo and upbo are global probabilities.

UTILITY COMMANDS

Add a utility interval statement

Call syntax: DMC_add_V_statement(struct user_stmt_rec* ustmtp)

Return information:
OK - statement number in the utility base
ERROR - input error

```
        no core memory
        frame corrupted
        frame not loaded
        too many statements
        too narrow statement
        inconsistent
```

Call semantics: Add the user utility statement u(crit:alt:cons) = [lobo,upbo] to the utility base within the decision frame. The base is checked for consistency with respect to the new interval. In case of inconsistency, for a loaded frame, nothing is added to the base, while for a non-loaded the statement is added without consistency checks. Non-loaded addition is only intended for loading pre-consistent data.

Add a utility link statement

Call syntax: DMC_add_V_statement(struct user_stmt_rec* ustmtp)

```
Return information:
OK    - statement number in the utility base
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        too many statements
        too narrow statement
        inconsistent
```

Call semantics: Add the user utility link u(crit1:alt1:cons1) - u(crit2:alt2:cons2) = [lobo,upbo] to the utility base within the decision frame. The base is checked for consistency with respect to the new link. In case of inconsistency, nothing is added to the base. Note: crit1 = crit2, two criteria arguments required only to preserve symmetry.

Change the bounds of a utility statement

Call syntax: DMC_change_V_statement(int stmt_number, real lobo, real upbo)

```
Return information:
OK    -
ERROR - input error
        frame corrupted
        frame not loaded
        too narrow statement
        inconsistent
```

Call semantics: Change the existing user utility statement u(crit:alt:cons) = [old_lobo,old_upbo] or u(crit1:alt1:cons1) - u(crit2:alt2:cons2) = [old_lobo,old_upbo] to u(crit:alt:cons) = [lobo,upbo] or u(crit1:alt1:cons1) - u(crit2:alt2:cons2) = [lobo,upbo] in the utility base. The base is checked for consistency with respect to the change. In case of inconsistency, nothing is changed the base.

Replace a utility statement with an interval statement

Call syntax:
DMC_replace_V_statement(int stmt_number, struct user_stmt_rec* ustmtp)

Return information:

```
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        too narrow statement
        inconsistent
```

Call semantics: Replace the user utility statement (interval or link) with u(crit:alt:cons) = [lobo,upbo] in the utility base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base.

Replace a utility statement with a link statement

Call syntax:
DMC_replace_V_statement(int stmt_number, struct user_stmt_rec* ustmtp)

```
Return information:
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        too narrow statement
        inconsistent
```

Call semantics: Replace the user utility statement (interval or link) with u(crit1:alt1:cons1) - u(crit2:alt2:cons2) = [lobo,upbo] in the utility base. The base is checked for consistency with respect to the new interval. In case of inconsistency, nothing is replaced in the base. Note: crit1 = crit2, there are two arguments only to preserve call symmetry.

Delete a utility statement

Call syntax: DMC_delete_V_statement(int stmt_number)

```
Return information:
OK    - number of statements remaining in the utility base
ERROR - input error
        frame corrupted
        frame not loaded
```

Call semantics: The user utility statement (interval or link) with position stmt_nbr in the utility base is deleted from the base. All statements with higher positions within the base are shifted one position down.

Make a utility estimate

Call syntax: DMC_make_V_estimate(struct user_stmt_rec* ustmtp)

```
Return information:
OK    -
ERROR - input error
        no core memory
        frame corrupted
        frame not loaded
        inconsistent
```

Call semantics: Add the user utility estimate u(crit:alt:cons) = [lobo,upbo] to the utility base within the decision frame. If the base is loaded, it is checked for consistency with respect to the new interval. In case of inconsistency, nothing is added to the base, while for adding a non-loaded base, the statement is added without consistency checks. The latter is only intended for loading pre-consistent data.

Remove a utility estimate

Call syntax: DMC_remove_V_estimate(struct user_stmt_rec* ustmtp)

Return information:
OK -
ERROR - input error
frame corrupted
frame not loaded

Call semantics: The user utility estimate for the consequence 'crit alt cons' in the utility base is deleted from the base. This can be considered unlocking the estimate.

Add utility sensitivity

Call syntax: DMC_add_V_sensitivity(h_matrix lobo, h_matrix upbo)

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded
inconsistent

Call semantics: The hull is replaced by a user determined orthogonal hypercube in order to facilitate manual sensitivity analyses (as opposed to the automatic contraction procedure).

Remove utility sensitivity

Call syntax: DMC_remove_V_sensitivity()

Return information:
OK -
ERROR - wrong frame type
frame corrupted
frame not loaded

Call semantics: The user determined orthogonal hypercube is replaced by the ordinary hull.

Get the utility hull

Call syntax: DMC_get_V_hull(h_matrix lobo, h_matrix ccp, h_matrix upbo)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
too many consequences

Call semantics: The symmetric hull and the contraction point are returned in three matrices lobo, ccp, and upbo indexed as [crit][alt][cons].

Get the utility core

Call syntax: DMC_get_V_core(h_matrix loco, h_matrix upco)

Return information:
OK -
ERROR - frame corrupted
frame not loaded

Call semantics: The core is returned in two matrices loco and upco indexed as [crit][alt][cons].

Get the utility intersection

Call syntax: DMC_get_V_intersection(h_matrix lobo, h_matrix upbo)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
no intersection
output error

Call semantics: The call must be preceded by an evaluation. The utility base contraction at the evaluation graph's intersection with the contraction axis is returned in two matrices lobo and upbo indexed as [crit][alt][cons].

Get the utility order

Call syntax: DMC_get_V_order(int crit, cs_matrix order)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
too many consequences

Call semantics: The utility order of all consequences in criterion crit is returned in a matrix order indexed as [cons1][cons2]. cons1 and cons2 are the consequences' total index numbers over all alternatives. Each entry pair (cons1,cons2) is either 0 (FALSE) or 1 (TRUE). TRUE indicates that there is information available that guarantees u(cons1) > u(cons2) in the entire consistent space. FALSE indicates that there exists at least one point where u(cons1) <= u(cons2).

Get the utility alt-order

Call syntax: DMC_get_V_altorder(o_matrix max_alt_order, o_matrix max_dom_class, o_matrix min_alt_order, o_matrix min_dom_class)

Return information:
OK -
ERROR - frame corrupted
frame not loaded

too many consequences

Call semantics: The utility order within each alternative is returned in a set of matrices indexed as [crit][alt][cons]. For each criterion and alternative, the utility ordering within the alternative is represented as dominance classes. Each class contains all consequences that possibly dominate each other in the consistent space. The start of each class is indicated by a number > 0 in max_dom_class (and min_dom_class), where the number is the size of the class. A zero represents a continuing class. For each class, the consequence numbers are given in max_alt_order (and min_alt_order). There are two orders, for maximizing and minimizing operations. They need not coincide. Consequently, for an ordered alternative, the size of all classes is 1. The alt_order elements [crit][0][0] contain the TRUE/FALSE flag for a total ordering within that criterion.

Get the utility order graph

Call syntax: DMC_get_V_order_graph(int crit, cs_matrix graph)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
too many consequences

Call semantics: The utility order graph of all consequences in criterion crit is returned in a graph matrix indexed as [cons1][cons2]. cons1 and cons2 are the consequences' total index numbers over all alternatives. Each entry pair (cons1,cons2) is either 0 (FALSE) or 1 (TRUE). TRUE indicates that u(cons1) > u(cons2) and that there exists no cons3 such that u(cons1) > u(cons3) and u(cons3) > u(cons2). Then there is an arrow from cons1 to cons2 in the corresponding graph.

Get the utility alt-order graph

Call syntax: DMC_get_V_altorder_graph(int crit, cs_matrix graph)

Return information:
OK -
ERROR - frame corrupted
frame not loaded
too many consequences

Call semantics: The utility alt-order graph of all consequences in criterion crit is returned in a graph matrix indexed as [cons1][cons2]. cons1 and cons2 are the consequences' total index numbers over all alternatives. Each entry pair (cons1,cons2) is either 0 (FALSE) or 1 (TRUE). If cons1 and cons2 belong to different alternatives, alt-order[cons1][cons2] is FALSE. TRUE indicates that u(cons1) > u(cons2) and that there exists no cons3 within the alternative such that u(cons1) > u(cons3) and u(cons3) > u(cons2). Then there is an arrow from cons1 to cons2 in the corresponding graph.

EVALUATION COMMANDS

Evaluate frame

Call syntax:
DMC_evaluate_frame(int method, int Ai, int Aj, int ncs, e_matrix e_result)

DMC_evaluate_opt(int method, int Ai, int Aj, int opt, int ncs, e_matrix e_result)

The 'opt' version of the call requires the caller to supply the number of the solver desired (M1-M4), while in the standard call, the solver is selected automatically. This can be used for evaluating frames approximately that otherwise could not be evaluated because they are QP-hard.

The number of contraction steps is given in 'ncs'. The valid range is from 6 (20% steps) to 21 (5% steps).

```
Method subfields:
Graph:   0 Relative pair
         1 Relative to set
         2 Absolute
Eval:    0 DELTA
         4 GAMMA
         8 PSI
        12 OMEGA (extended)
Hull:    0 Asymmetric
        16 Symmetric
W_con:  32 No contraction
         0 Contraction
        64 Point contraction
P_con: 128 No contraction
         0 Contraction
       256 Point contraction
V_con: 512 No contraction
         0 Contraction
      1024 Point contraction

Return information:
OK    -
ERROR - input error
        frame corrupted
        frame not loaded
        output error
        solver aborted
        not implemented
        license expired
```

Call semantics: All alternatives are evaluated using the Delta, Gamma, Psi, or Omega rules. For the requested alternative(s) Ai (and Aj), the result is stored in e_result. Each result has the form of a matrix {min,mid,max} x {contraction}, with values from increasing contraction. Currently there are 6-21 values corresponding to contractions of 0-100% in 5-20% steps. Delta and Gamma cannot be evaluated in absolute graph mode since it is impossible to generate an absolute scale from relative ones. For extended Omega, the stepwise contracted core is returned. Aj is relevant only for relative pair graph evaluations.

Get evaluation graph area

Call syntax: DMC_get_eval_area(real* area)

```
Return information:
OK    -
ERROR - output error
```

Call semantics: Obtains the fraction [0,1] of the area residing above the x-axis for a standard (cone) plot of the evaluation result. The call must be preceded by an evaluation.

Security levels

Call syntax: DMC_sec_level(int method, real v_min, int ncs, s_matrix s_result)

The number of contraction steps is given in 'ncs'. The valid range is from 6 (20% steps) to 21 (5% steps).

```
Method subfields:
Hull:    0 Asymmetric
        16 Symmetric
P_con: 128 No contraction
         0 Contraction
       256 Point contraction
V_con: 512 No contraction
         0 Contraction
      1024 Point contraction

Return information:
OK    -
ERROR - input error
        frame corrupted
        frame not loaded
        license expired
```

Call semantics: For a PS-frame, the security level 'v_min' specified in the call is evaluated. The result has the form of a matrix containing a number of result values for each alternative from increasing contraction. Currently there are 6-21 values corresponding to contractions of 0-100% in 5-20% steps and three such sets: min, mid, and max. They are stored in the matrix s_result[alt][set][con] where 'alt' is the sequence number of the alternative, 'set' is min, mid or max, and 'con' is the contraction step. For a DM-frame, the call is not allowed as the function is undefined.

MISCELLANEOUS COMMANDS

Get release version

Call syntax: DMC_get_release(char* relstrg)

```
Return information:
OK    -
```

Call semantics: Obtains the release of DMC and DeltaLib. The format is "M1.F1 (M2.F2.T2)", where M1=main, F1=functional for DMC and M2=main, F2=functional, T2=technical for DELTALIB. User interface code must assure that M2 matches expectations. If F2 does not match, let user proceed at own risk. T2 is not for user actions.

Get solver capacity

Call syntax: DMC_get_capacity(char* capstrg)

```
Return information:
OK    -
```

Call semantics: Obtains the capacities of the DADE solver. Returns the string "maxcrit maxalt maxcons maxcopa (maxWstmt maxPstmt maxVstmt)" with the

maximum number of criteria, alternatives, consequences, consequences per alternative, W_statements, P_statements, and V_statements respectively.

Get solver matrix size

Call syntax: DMC_get_solver_size(char* sizestrg)

Return information:
OK -

Call semantics: Obtains the matrix size of the DELTALIB solver. Returns the string "maxrows x maxcols" with the maximum number of rows and columns.

Get platform identifier

Call syntax: DMC_get_platform(char* pfstrg)

Return information:
OK -

Call semantics: Obtains the name of the platform that DeltaLib has been compiled for.

Get solver precision

Call syntax: DMC_get_precision()

Return information:
OK - precision number

Call semantics: Obtains the floating point precision that DeltaLib has been compiled for. Possible precision numbers are 1=single precision or 2=double precision.

Get frame information

Call syntax: DMC_get_frame_info(char* infostrg)

Return information:
OK -
ERROR - frame corrupted
frame not loaded

Call semantics: Returns basic information about the current frame as a string "Wn Pn Vn Mn". Wn is the W_base type (W0-W4). Pn is the P_base type (P0-P4). Vn is the V_base type (V0-V3). Mn is the comparison algorithm required by the frame (M1-M5) where M1=N-Opt, M2=V-Opt, M3=P-Opt, M4=VP-Opt, M5=QP-Opt.

Get number of weight statements

Call syntax: DMC_nbr_of_W_stmts()

Return information:
OK - number of weight statements in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of weight statements in the currently loaded frame.

Get number of probability statements

Call syntax: DMC_nbr_of_P_stmts()

Return information:
OK - number of probability statements in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of probability statements in the currently loaded frame.

Get number of utility statements

Call syntax: DMC_nbr_of_V_stmts()

Return information:
OK - number of utility statements in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of utility statements in the currently loaded frame.

Get number of criteria

Call syntax: DMC_nbr_of_crit()

Return information:
OK - number of criteria in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of criteria in the currently loaded frame.

Get number of alternatives

Call syntax: DMC_nbr_of_alts()

Return information:
OK - number of alternatives in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of alternatives in the currently loaded frame.

Get total number of consequences

Call syntax: DMC_total_cons()

Return information:
OK - number of consequences in the specified alternative
ERROR - frame not loaded

Call semantics: Returns the total number of consequences in all alternatives in the currently loaded frame.

Get number of consequences

Call syntax: DMC_nbr_of_cons(int crit, int alt)

Return information:
OK - number of consequences in the specified alternative
ERROR - frame not loaded

Call semantics: Returns the number of consequences in the specified alternative in the currently loaded frame.

ERROR HANDLING

All DMC calls (except DMC_get_errtxt() below) return an integer which serves as an error code and information carrier at the same time. In the event of an error, a negative number is returned. The caller should interpret the error code and take action accordingly.

DMC error codes

DMC_DELTA_ERROR

The error occurred in the Delta solver. This value is not returned alone, but instead added to DMC_delta_errcode().

DMC_INPUT_ERROR

One of the input parameters contained invalid information.

DMC_OUTPUT_ERROR

The requested output from the DMC function could not be generated. This usually refers to a request for impossible evaluation data.

DMC_OUT_OF_FRAMES

The number of frames in the DMC layer has been exceeded.

DMC_NO_SUCH_FRAME

The requested frame number does not exist. Either it is not created, or the number is out of range.

DMC_FRAME_IN_USE

An attempt to delete or in another way eliminate a frame that is currently attached (loaded).

DMC_FRAME_NOT_LOADED

An attempt to use frame modification commands while no frame is loaded.

DMC_FRAME_CORRUPT

Internal error. The frame has been rendered corrupt, either by modifications outside of Delta or because of an internal error in Delta.

DMC_WRONG_FRAME_TYPE

An attempt to issue a PS-only command to a DM frame or vice versa.

DMC_WRONG_STATEMENT_TYPE

The user statement passed in the call is inappropriate for the type of frame currently loaded.

DMC_CONS_OVERFLOW

Too many consequences in the problem for DMC to handle. This should be prohibited in the user interface at an earlier point.

DMC_LICENSE_EXPIRED

The license for the DMC package is time limited. The end date has passed. The user must renew his license agreement.

DMC_NO_INTERSECTION

The current evaluation graph does not intersect with the contraction axis. No contracted base can be returned.

DMC_CONFIG_MISMATCH

At least one of the five parameters MAX_CRIT, MAX_ALTS, MAX_CONS, MAX_COPA, or MAX_STMTS does not have the same value in the caller's name space as in DeltaLib's name space.

DMC_MEMORY_LEAK

At reconciliation time, allocated memory still remains in use even though it should all be freed. Internal error in DMC.

Get Delta error code

Call syntax: DMC_delta_errcode()

Return information:
OK - Delta error code
ERROR - input error
frame corrupted
frame not loaded

Call semantics: Returns the error code from the latest DeltaLib call.

Delta error codes

In the event of a DMC_DELTA_ERROR, a problem with the request has been detected in the Delta kernel. DMC records the error and it is passed on to the DMC caller by DMC_delta_errcode(). The possible codes are:

INCONSISTENT

The call results in a previously consistent frame becoming inconsistent. The call has been rolled back, and the frame is in the same state as it was before the call.

INPUT_ERROR

An input parameter contains illegal data, for example an index out of range or values not within given intervals.

NO_MEMORY

The kernel has run out of memory. This is the result of allocating too little virtual memory to the application in which the Delta solver is hosted.

TOO_MANY_CONS

The problem contains too many consequences. This should be prohibited in the user interface at an earlier point.

REFERENCED

An attempt to remove a consequence or an alternative that still contains references from statements. This should be prohibited in the user interface.

NOT_IMPLEMENTED

The function requested or the solver necessary for the problem at hand has not yet been implemented. This should be prohibited in the user interface at an earlier point.

WRONG_RELEASE

The wrong release of Delta is used. Must be upgraded to a more recent.

TOO_MANY_STMTS

The problem contains too many statements. This should be prohibited in the user interface at an earlier point.

ATTACHED

An attempt to delete or in another way eliminate a frame that is currently attached (loaded).

TOO_NARROW_STMT

The solver could operate in a mode where, for reasons of speed and stability, intervals of very small size are not allowed. This excludes the use of pointwise statements.

SOLVER_ABORTED

Internal error in Delta or an attempt to solve a problem whose size is on the border of the solver's capability.

Get error text

Call syntax: char *DMC_get_errtxt(int jrc)

Return information:
OK - pointer to error text
ERROR – pointer to text "- no text -"

Call semantics: Returns the text string that corresponds to the supplied DMC error number.