Adrian Brasoveanu Draft June 16th 2003

MINIMAL FUSION NORMAL FORM

I. THE PROBLEM AND THE PROPOSED SOLUTION.

THE PROBLEM: For an arbitrary set of vectors S of cardinality n, S={v_1, v_2, …, v_n}, find a set of vectors S' such that:

(1) i.*S' equivalent to S* (where 'equivalent' means 'true in the same models', i.e. characterizing the same set of total constraint rankings)

ii. *all the vectors in S' are independent* (where independent means that, for every vector $v_i \in$ S', there is no non-null subset of S' which does not contain v_i as a member and which entails v_i; 'entailment' is defined as: the set of vectors S entails a vector v (S $\models$v) iff the models in which S is true are a subset of the models in which v is true);

iii. the vectors in S' *'make clear/transparent'* what is the set of models in which they are true, i.e. they *'make clear/transparent'* what are the necessary and sufficient ranking conditions enforced by the set S (and S').

The meaning of 'make clear/transparent' in (1iii) is the same as the one behind the operation of fusion (◦) in Prince 2002, namely obtaining a vector with a maximum number of Ls such that ranking conditions which were previously 'hidden' (i.e. not readable directly from the vector) are now 'transparent', i.e. directly readable from the vector.

The elaboration of the meaning of 'make clear/transparent' anticipates the solution.

THE PROPOSED SOLUTION: obtaining the set S' satisfying the requirements in (1) is achieved through an intermediary step, namely obtaining a set S'' of 'maximally informative' vectors corresponding to the vectors in the initial set S.

(2) A vector v' is maximally informative wrt a vector v and a set S such that v$\in$ S iff:

i. v' $\models$v

ii. S $\models$v'

iii. ~$\exists$v'' (v''$\neq$v' & v'' $\models$v' & S $\models$v'')

For example, if S={WWLW(=a), eeWL (=b)}, the vector which is maximally informative wrt a and S is the vector a'=WWLL=a◦b. The vector a''=eWLL entails a' (by W extension), but is not entailed by S; the same can be said about the vector LLLL, which entails everything but is not entailed by S unless S is inconsistent (throughout this paper I will consider only consistent sets of vectors).

As suggested in the above example, a maximally informative vector is obtained by fusion. However, fusion is not guaranteed to yield a vector entailing any of the vectors

entering the fusion and I will propose a procedure for restricting it depending on the initial set of vectors S.

The set S'' obtained by such a restricted fusion will be called (for lack of a better name), the *Fusion Normal Form* of the initial set S, i.e. FNF(S).

Once FNF(S) is obtained, we know that it is equivalent to the initial set by the definition of 'maximally informative' provided in (2). Moreover, given that we obtained the vectors in S'' by *restricted* fusion (see below), we also know that (1iii) is satisfied. However, there is no guarantee that the set S'' satisfies condition (1ii) above, i.e. that all the vectors in it are independent. Eliminating the entailed vectors in FNF(S) is the last step before obtaining the set S', i.e. the solution to the problem. This last set will be called the *Minimal Fusion Normal Form* of the initial set S, i.e. MFNF(S).

MFNF(S) is a possible answer to the problem in (1).

The rest of the paper will provide the procedure for obtaining FNF(S) and MFNF(S) and relevant examples for each step of the algorithm will be given. The algorithm specification will also answer the two points left obscure in the above solution summary, namely (a) what is *restricted* fusion and (b) obtaining MFNF(S) from FNF(S).

I will not provide the formal proofs that all the steps in the algorithm are necessary and sufficient to yield the claimed solution; also, I will not attempt a comparison with the RCD algorithm (see for example Tesar 1995), e.g. the usage of 'info loss configurations' – see (5) below for the definition - in RCD and in the present algorithm.

The final version of the algorithm is given below for ease of reference:

(17) **The FNF(S) algorithm**:

a. **Info loss configurations**: for a given set S of n vectors that form a matrix with m columns and n rows, identify all the info loss configurations;

b. **Branch construction and elimination**: construct a branch/daughter for each info loss configuration; each daughter contains all the vectors that have an *e* in that particular info loss configuration; for any two daughters that are in a subset-superset relation, eliminate the subset daughter;

c. **Fusion of the mother node**: if the union of the sets of vectors in all the daughters is a proper subset of the set of vectors in the mother, then fuse all the vectors in the mother node;

d. **Recursion**: for each daughter that has a non-singleton set of vectors, reapply the steps (a)-(c);

e. **Pre-FNF(S) formation**: if all the daughters are singleton sets (or empty sets), form the pre-FNF set, i.e. the set containing all the fusions in the tree and all the vectors in singleton daughters;

f. **W extension, path inspection and FNF(S)**: test in a bottom-up manner each path of the tree for W extension entailment, i.e. look only at the nodes that are either singleton leaves or have a fusion and, if the vector of a node A is entailed by the vector of a node B, where A dominates B, remove the vector of the node A from pre-FNF. FNF is the set obtained after all paths in the tree are tested and all the entailed vectors are removed.

(18) **MFNF(S)**: for all nodes that contain a fusion, fuse all the vectors in all its daughters and check if this 'total daughter fusion' is the same as or entails by W

extension the mother fusion. If this is the case, remove the mother fusion from the set FNF(S).

II. THE ALGORITHM FOR OBTAINING FNF(S) AND MFNF(S).

II.1. OBTAINING FNF(S).

FNF(S) was defined as:

(3) for any set of vectors S, FNF(S) [1] is the set S'' such that:
i. $\forall v\ [v \in S \rightarrow \exists v''\ (v'' \in S''\ \&\ v''$ is maximally informative wrt v, S)]
ii. $\forall v''\ [v'' \in S'' \rightarrow \exists v\ (v \in S\ \&\ v''$ is maximally informative wrt v, S)]
i.e. the set S'' contains all and only the vectors that are maximally informative wrt S and the vectors in S.

As mentioned above, I will obtain the maximally informative vectors by fusion, which 'maximizes' the number of Ls in any given vector, i.e. it replaces the maximum number of Ws and e-s with Ls. However, at the same time, fusion will replace e-s with Ws in any coordinate/column that contains no Ls and at least one W. Both fusion and (Kleene) conjunction are defined as the *min* function applied to an input pair of truth values in a trivalent logic:

(4) i. conjunction ($\wedge$): for any x, y in the set {L, e, W}, $x \wedge y = \min(x, y)$ wrt the total order L<e<W [2];
ii. fusion ($\circ$): for any x, y in the set {L, e, W}, $x \circ y = \min(x, y)$ wrt the total order L<W<e [3], [4].

Conjunction is info preserving, i.e. $x \wedge y \models \{x, y\}$, but very often useless, since in many cases the conjunction of a *consistent* vectors is a member of L^+ [5].
Fusion is better equipped to locally maximize information and make 'overt/ transparent' implicit ranking conditions while, at the same time, avoiding to make the result a member of L^+ (as long as the input is a consistent set of vectors). However,

---

[1] It seems like the cardinality of FNF(S) is at most equal to the cardinality of S; however, see (7/7') below for an example in which there can be two maximally informative vectors wrt the same vector and set; it seems more likely that MFNF(S) has a cardinality less than or equal to the cardinality of S.
[2] I use '$\wedge$' as the symbol for conjunction in the object language, reserving '&' for the metalanguage.
[3] the two definitions suggest that fusion can be defined in terms of conjunction and 'restricted' W extension: consider the vectors a=eWLe, b=eeWL; $a \wedge b$=eeLL and $a \circ b$=eWLL; blindly applying W extension to $a \wedge b$ can yield WeLL or WWLL; the 'restriction' on W extension is the following: do W extension only in the coordinate i of the conjunction that (a) contains an e and (b) the column i in the (matrix) input contains only e-s and at least one W; the 'restriction' on w extension is basically the same as the 'info loss configuration' that is defined in the main text and which is central in obtaining the FNF of a set S.
[4] fusion and conjunction can be recursively defined for an arbitrary set of vectors taking the definitions in (4) as the basic case
[5] I will take L^+ to be the set of words over the alphabet {e, L}, i.e. $\{e, L\}^*$; see also Prince 2002.

fusion is not info-preserving, i.e. it is not generally the case that $x \circ y \models \{x, y\}$, although the other direction of the entailment always holds, i.e. $\{x, y\} \models x \circ y$ [6].

It is precisely the input {e, W} that causes fusion to loose information (the same diagnostic is implicit in the notion of W compliance in Prince 2002); thus, whenever we fuse a column/coordinate that contains an e and a W, the row/vector that contains the e will be 'weakened' by the addition of a W, i.e. of one extra disjunction. Generally, I will define 'info loss configuration' as:

(5) a column/coordinate in a set/matrix of vectors that contains no Ls and at least one W is an *info loss configuration*.

Hence, in fusing vectors, one should note all the info loss configurations and 'reassert' the vectors that have an *e* in an info loss configuration, since those vectors contain the information that is lost in the fusion.

An example will make the above observations clearer. Consider the set of vectors given in (6) below:

(6)

	1	2	3	4	5
a	W	L	e	e	W
b	e	W	L	W	L
c	e	e	W	L	W
a∘b∘c	**W**	**L**	**L**	**L**	**L**

Given the set of vectors (the matrix) in (6), we look for the info loss configurations and we identify one in column 1, i.e. in the column of the constrain C_1. Vectors b and c have an *e* in that column so, if we are to fuse all three vectors a∘b∘c, we will loose information in those vectors; hence, they need to be reasserted. However, as far as vector a is concerned, the fusion only adds information. Actually, a∘b∘c is maximally informative wrt vector a and the set {a, b, c}.

Since the set of all reasserted vectors is a subset of the vectors in the mother node, it makes sense to fuse all the vectors in the mother in order to obtain a maximally informative vector; if the set of all reasserted vectors (i.e. the vectors in the daughters) equals the set of vectors in the mother node, fusing the vectors in the mother node would not maximize the information in any vector, hence it would be useless.

The next step is to reassert vectors b and c, as in (6') below.

(6')

	1	2	3	4	5
a	W	L	e	e	W

[6] unlike in the case of conjunction (!) – consider a consistent set of vectors {x, y} (where consistent means that it is satisfiable by at least one model, i.e. there is at least one model/constraint ranking in which is is true) whose conjunction is a member of L^+; a member of L^+ is not satisfiable by any model, i.e. it is false in all models, i.e. it characterizes the empty set of constraint rankings; hence, it is not the case that $\{x, y\} \models x \wedge y$.

b	e	W	L	W	L
c	e	e	W	L	W
a◦b◦c	**W**	**L**	**L**	**L**	**L**

↓

	1	2	3	4	5
b	e	W	L	W	L
c	e	e	W	L	W
b◦c	**e**	**W**	**L**	**L**	**L**

Looking at b and c, we identify an info loss configuration in column 2, i.e. constraint C_2; thus, fusing b and c will maximize the information in vector b - b◦c is maximally informative wrt b and {a, b, c} (not only {b, c}); however, the fusion will loose information in vector c, which needs to be reasserted, as in (6'') below:

(6'')

	1	2	3	4	5
a	W	L	e	e	W
b	e	W	L	W	L
c	e	e	W	L	W
a◦b◦c	**W**	**L**	**L**	**L**	**L**

↓

	1	2	3	4	5
b	e	W	L	W	L
c	e	e	W	L	W
b◦c	**e**	**W**	**L**	**L**	**L**

↓

	1	2	3	4	5
c	**e**	**e**	**W**	**L**	**W**

The FNF is obtained by putting together all the leaves of the tree thus obtained and all the fusions (if any) found at the nonterminal nodes.

For the set of vectors in example (6), FNF({a, b, c}) = {a◦b◦c, b◦c, c}. It so happens that all these vectors are independent, hence FNF({a, b, c}) = MFNF({a, b, c}) = {a◦b◦c, b◦c, c}.

A more complicated example can illustrate additional complications in the algorithm and, at the same time, highlight one of its central benefits. Consider the set of vectors in (7) below:

(7)

	1	2	3	4	5	6	7
a	W	L	W	e	e	e	W
b	W	L	W	L	e	e	e
c	e	W	L	L	L	e	W

d	W	L	L	e	e	L	e

We identify 2 info loss configurations (C_1 and C_7), so we construct 2 branches in the tree, one for each info loss configuration.

However, before doing that, note the logical relations between the vectors in (7): b ⊨a, b◦d ⊨b(⊨a) and b◦d ⊨d (note that b and d are W compliant). The fusion b◦d is capable of sweeping through the tableau/matrix in (7) and eliminate a considerable amount of redundancies – this being one of the great advantages of fusion as characterized in Prince 2002. The general problem is to identify such fusions, given an arbitrary set of vectors – and the algorithm for obtaining FNF seems capable of solving this problem.

After identifying the info loss configurations, the next step in obtaining FNF(S) consists in constructing a branch for each info loss configuration and reassert the vectors that loose information in each such configuration.

(7')

	1	2	3	4	5	6	7
a	W	L	W	e	e	e	W
b	W	L	W	L	e	e	e
c	e	W	L	L	L	e	W
d	W	L	L	e	e	L	e
a◦b◦c◦d	**W**	**L**	**L**	**L**	**L**	**L**	**W**

	1	2	3	4	5	6	7
c	**e**	**W**	**L**	**L**	**L**	**e**	**W**

	1	2	3	4	5	6	7
b	W	L	W	L	e	e	e
d	W	L	L	e	e	L	e
b◦d	**W**	**L**	**L**	**L**	**e**	**L**	**e**

On the second branch of the tree, i.e. on the branch which reasserts b and d, we look again for an info loss configuration (C_1); this info loss configuration does not contain any *e*, so no vectors need to be reasserted; since the set of reasserted vectors (i.e. the empty set) is a subset of the vectors in the mother (i.e. the set {b, d}), we fuse the two vectors and the fusion b◦d is maximally informative with respect to both b and d; this is how the present algorithm captures W compliance.

On the first branch we are left with only one vector (c), hence we stop.

FNF({a, b, c, d}) = {a◦b◦c◦d, c, b◦d}

As anticipated, the FNF algorithm identifies the 'powerful' fusion b◦d. However, we lost part of its 'power', namely the entailment b◦d ⊨a. Thus, the minimal set equivalent to the initial set is actually MFNF(S)={c, b◦d} and FNF(S) contains the redundant fusion a◦b◦c◦d.

Example (7) establishes the point that FNF is not yet a solution to the problem in (1), since the vectors in FNF are not independent. Hence, an independent mechanism is

needed to 'prune' such redundant vectors in FNF; as we will see later on, all such redundancies can be linked more or less directly to entailment by W extension [7].

(The key to eliminate the redundant fusion a∘b∘c∘d will be the requirement that, at each step of constructing daughters/branches for a mother node which contains a fusion (remember that a mother node contains a fusion iff the set of vectors in the daughters is a proper subset of the set of vectors in the mother), one should fuse all the vectors in daughters and check if this 'total daughter fusion' is the same as or entails by W extension the fusion in the mother. In the present example, note that the 'total daughter fusion' c∘b∘d is the same as the mother fusion a∘b∘c∘d.)

Example (7) makes the additional point that, given a vector v and a set S (v∈S), there might be more than one vector which is maximally informative wrt v, S (according to the definition in (2)). Thus, the vector a∘b∘c∘d is maximaly informative wrt a, {a, b, c, d} and the vector b∘d is also maximally informative wrt a, {a, b, c, d}. In particular, note that b∘d does not entail a∘b∘c∘d and a∘b∘c∘d does not entail b∘d.

As far as I know, there is no principled way to identify that in FNF({a, b, c, d}) = {a∘b∘c∘d, c, b∘d} there are two vectors maximally informative wrt the same vector a.

I will examine two more things before giving the final form of the FNF(S) algorithm; first, I will look at certain sets of info loss configurations and improve the 'tree construction' procedure suggested above; second, I will look at how the algorithm fares wrt initial sets of vectors that contain entailments by W extension and L retraction.

Info loss configurations, 'branch elimination' and maximum number of branches/daughters per node.

Consider the vectors in (8) below:

(8)

	1	2	3	4	5
a	W	L	e	W	W
b	e	W	L	e	e
c	e	e	L	W	e

There are 3 info loss configurations in (8), namely C_1, C_4 and C_5. The algorithm as previously stated would have to construct a branch/daughter for all three of them; however, the third daughter corresponding to C_5 would just repeat the first daughter, hence it would be redundant as far as FNF goes. Moreover, the second daughter would contain only {b} which is a subset of the set in the first daughter {b, c} – hence, constructing a branch for C_2 would be again redundant, since that branch will certainly be a sub-branch of the daughter corresponding to C_1.

[7] the algorithm as developed up until now is incapable of 'detecting' entailment by W extension, although it can 'detect' and eliminate all entailments/redundancies based on L retraction – this is due to the nature of fusion and to our definition of info loss configuration; more about this later.

Thus, at any point in the construction of the tree, there should be a 'branch elimination' step, requiring that, for all info loss configurations that would license daughter nodes that contain set of vectors in the subset-superset relation, construct a branch only for the info loss configuration that licenses the superset.

Thus, the next step in the algorithm is to construct only one branch/daughter containing the set {b, c}. Moreover, since the set of vectors in the daughters is a proper subset of the vectors in the mother node, we fuse the vectors in the mother node, as in (8') below:

(8')

	1	2	3	4	5
a	W	L	e	W	W
b	e	W	L	e	e
c	e	e	L	W	e
a∘b∘c	**W**	**L**	**L**	**W**	**W**

↓

	1	2	3	4	5
b	e	W	L	e	e
c	e	e	L	W	e

At this point, we identify two info loss configurations in the daughter, namely C_2 and C_4. For each of them, we construct a branch, each containing a single vector (b and c respectively); since all the daughters in all the branches are singleton sets, the computation stops and FNF({a, b, c}) ={a∘b∘c, b, c}.

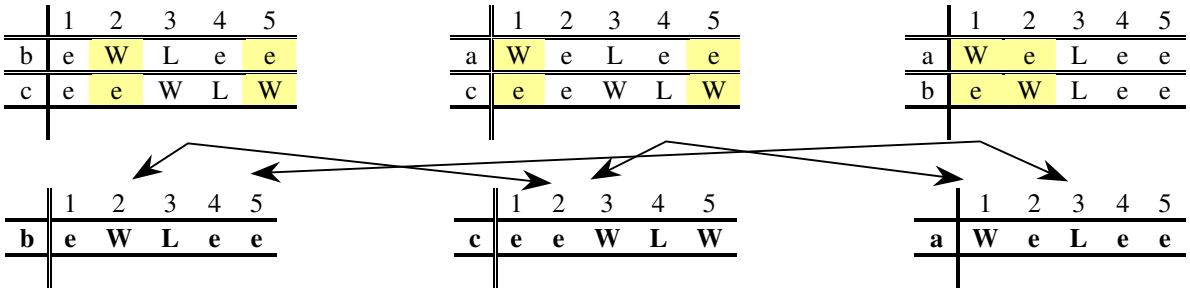
However, there is one thing in (8') suggesting that the computation should stop and that all the vectors in the set {b, c} are in the final FNF: given the info loss configurations in the set {b, c} we should construct a number of branches/daughters equal to the number of vectors in the mother node even after branch elimination.

There seems to be a generalization at work: *if, after the application of 'branch elimination' (see the discussion above), the number of daughters to be constructed for a mother node is equal to or greater than the number of vectors in the mother node, then the mother node is a subset of the final FNF, i.e. all the vectors in the mother node should be added to the FNF.*

The generalization seems confirmed by the sets of vectors in (9) and (10) below.

(9)

	1	2	3	4	5
a	W	e	L	e	e
b	e	W	L	e	e
c	e	e	W	L	W



The FNF of the set in (9) is identical to the initial set of vectors. A similar computation yields a FNF identical to the initial set in (10) below – where the number of branches is greater than the number of vectors of the mother node [8].

(10)

	1	2	3	4	5	6
a	W	W	e	W	e	L
b	W	e	W	e	W	L
c	e	W	W	e	e	L
d	e	e	e	W	W	L

However, the set of vectors in (11) below provides a counterexample to the proposed generalization:

(11)

	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
b	e	e	W	e	e	L	L	W
c	e	W	e	e	W	L	e	W
d	W	e	e	e	W	L	e	e
e	W	W	W	W	e	L	W	L

We have to construct 5 branches.
The 1st branch is C_1:

C_1	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
b	e	e	W	e	e	L	L	W
c	e	W	e	e	W	L	e	W

[8] I am indebted to Alan Prince for pointing out a major error in my previous observations about these matters.

Looking at the info loss configurations in C_2, C_3 and C_4, we see that this case is similar to the one in (9), hence it will 'break up' in the set {a, b, c}.

The 2nd branch is C_2:

C_2	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
b	e	e	W	e	e	L	L	W
d	W	e	e	e	W	L	e	e

Again, the info loss configurations in C_1, C_3 and C_4 reduce this branch to (9), i.e. to the set {a, b, d}.

The 3rd branch is C_3:

C_3	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
c	e	W	e	e	W	L	e	W
d	W	e	e	e	W	L	e	e

The info loss configurations in C_1, C_2 and C_4 reduce this branch to (9), i.e. to the set {a, c, d}.

The 4th branch is C_4:

C_4	1	2	3	4	5	6	7	8
b	e	e	W	e	e	L	L	W
c	e	W	e	e	W	L	e	W
d	W	e	e	e	W	L	e	e

The info loss configurations in C_1, C_2 and C_3 reduce this branch to (9), i.e. to the set {b, c, d}.

The 5th branch is C_5:

C_5	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
b	e	e	W	e	e	L	L	W
e	W	W	W	W	e	L	W	L
a◦b◦e	**W**	**W**	**W**	**W**	**e**	**L**	**L**	**L**

	1	2	3	4	5	6	7	8
a	e	e	e	W	e	L	W	e
b	e	e	W	e	e	L	L	W

	1	2	3	4	5	6	7	8
a	**e**	**e**	**e**	**W**	**e**	**L**	**W**	**e**

	1	2	3	4	5	6	7	8
b	**e**	**e**	**W**	**e**	**e**	**L**	**L**	**W**

Thus, the final FNF({a, b, c, d, e}) = {a, b, c, d, a◦b◦e}, despite constructing 5 branches for the top node of the tree.

I do not know whether the generalization suggested above (linking the number of vectors in a node and the maximum possible number of daughters for that node *after 'branch elimination'*) captures a real regularity; if something like the above generalization is true, the complexity of the FNF algorithm will be more tightly limited by the number of initial vectors. As it stands, the algorithm predicts that the maximum number of branches for a node with n vectors (after 'branch elimination') is: (a) n choose n/2 for n an even natural number and (b) n choose (n-1)/2 (equal to n choose (n+1)/2) for n an odd natural number.

In any case, note that the number of branches for a given node does not depend on the number of constraints, but only on the number of vectors.

I will now turn to the examination of initial sets of vectors that contain entailments by L retraction or/and W extension.

Consider the set of vectors in (12) below, in which a entails b by L retraction:

(12)

	1	2	3
a	W	L	L
b	W	L	e
a◦b	**W**	**L**	**L**

As expected, if there are vectors entailed by L retraction only, they are W compliant and the FNF algorithm successfully identifies and 'reduces' them to the more informative vector.

However, due to the definition and role of fusion and info loss configuration, the FNF algorithm cannot identify and eliminate vectors that are entailed by W extension, as shown by (13) below:

(13)

	1	2	3
a	e	W	L
b	W	W	L
a◦b	**W**	**W**	**L**

↓

	1	2	3
a	**e**	**W**	**L**

By the definition of fusion, the fusion of two vectors a and b such that a ⊨b by L retraction is identical with the more informative vector, i.e. a◦b=a; however, the fusion of two vectors a and b such that a ⊨b by W extension is identical with the less informative vector, i.e. a◦b=b. In this case, the FNF algorithm correctly identifies the info loss

configuration and reasserts the more informative vector; however, it includes in the final FNF a vector which is NOT maximally informative; in example (13), the maximally informative vector wrt b, {a, b} is a and NOT a◦b=b.

Hence, an extra step has to be added in the FNF algorithm:

Once the entire tree is constructed, collect all fusions and leaves in the tree; call this set pre-FNF. Then, test in a bottom-up manner each path of the tree for W extension entailment; that is, look only at nodes that are either leaves or have a fusion and, if the vector of a node A is entailed by the vector of a node B, where A dominates B, remove the vector of the node A from the set pre-FNF. FNF is the set obtained after all paths in the tree are tested in this way and all the entailed vectors are removed from pre-FNF [9]***.***

Note that it makes sense to test a path only if its leaf vector contains at least one *e*.

The same point is established by the following set of vectors:

(14)

	1	2	3	4
a	W	L	W	W
b	W	L	e	e
c	L	e	W	e
a◦b◦c	**L**	**L**	**W**	**W**

↓

	1	2	3	4
b	W	L	e	e
c	L	e	W	e
b◦c	**L**	**L**	**W**	**e**

↓

	1	2	3	4
b	**W**	**L**	**e**	**e**

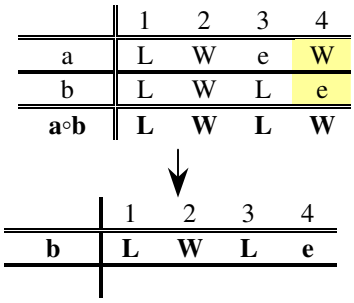
The fusion b◦c entails the fusion a◦b◦c. The FNF is {b, b◦c}. Thus, the algorithm correctly identifies that b ⊨a by W extension and, moreover, that b◦c is maximally informative wrt both c, {a, b, c} and a, {a, b, c}.

I will now look at some examples that involve entailment by both L retraction and W extension and show that the FNF algorithm with the extra step of bottom-up path inspection correctly identifies and eliminates the entailed vectors.

Consider the set of vectors in (15) below, which combine entailment by L retraction and W extension:

(15)

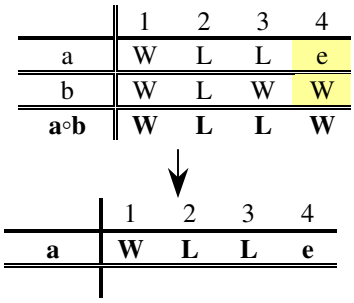
[9] a proof has to be provided that the test for W extension should be limited only to bottom up path inspection, i.e. that it is impossible to have two nodes A and B such that A ⊨B by W extension and A is not dominated by B or identical to some node C that is dominated by B.



As desired, FNF({a, b})={b}

The same welcome result is given for the set of vectors in (16) below, where L retraction and W extension apply in a slightly more complicated fashion:

(16)



Summarizing, the algorithm for obtaining FNF(S) is the following:

(17) **The FNF(S) algorithm**:

a. **Info loss configurations**: for a given set S of n vectors that form a matrix with m columns and n rows, identify all the info loss configurations;

b. **Branch construction and elimination**: construct a branch/daughter for each info loss configuration; each daughter contains all the vectors that have an *e* in that particular info loss configuration; for any two daughters that are in a subset-superset relation, eliminate the subset daughter;

c. **Fusion of the mother node**: if the union of the sets of vectors in all the daughters is a proper subset of the set of vectors in the mother, then fuse all the vectors in the mother node;

d. **Recursion**: for each daughter that has a non-singleton set of vectors, reapply the steps (a)-(c);

e. **Pre-FNF(S) formation**: if all the daughters are singleton sets (or empty sets), form the pre-FNF set, i.e. the set containing all the fusions in the tree and all the vectors in singleton daughters;

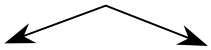
f. **W extension, path inspection and FNF(S)**: test in a bottom-up manner each path of the tree for W extension entailment, i.e. look only at the nodes that are either singleton leaves or have a fusion and, if the vector of a node A is entailed by the vector of a node B, where A dominates B, remove the vector of the node A from pre-FNF. FNF is the set obtained after all paths in the tree are tested and all the entailed vectors are removed.

II.2. OBTAINING MFNF(S).

As noted in the previous section, the FNF(S) is not guaranteed to contain only independent vectors. The set of vectors in (7/7'), repeated below, provides a relevant example:

(7/7')

	1	2	3	4	5	6	7
a	W	L	W	e	e	e	W
b	W	L	W	L	e	e	e
c	e	W	L	L	L	e	W
d	W	L	L	e	e	L	e
a∘b∘c∘d	**W**	**L**	**L**	**L**	**L**	**L**	**W**



	1	2	3	4	5	6	7
c	**e**	**W**	**L**	**L**	**L**	**e**	**W**

	1	2	3	4	5	6	7
b	W	L	W	L	e	e	e
d	W	L	L	e	e	L	e
b∘d	**W**	**L**	**L**	**L**	**e**	**L**	**e**

FNF({a, b, c, d}) = {a∘b∘c∘d, c, b∘d}. However, a∘b∘c∘d=c∘b∘d, but the step in (17f) above (entailment by W extension) – even if not restricted to paths (!) – is not capable of detecting this entailment; we see here an interesting consequence of the FNF algorithm (more exactly of the interaction between branch construction based on info loss configurations and the condition on mother node fusion): if vector c would have had an *e* instead of an L in column C_5, step (17f) (W extension) would have been enough to detect the entailment. Also, it is precisely this configuration that generates two different vectors (i.e. both b∘d and c∘b∘d= a∘b∘c∘d) as maximally informative wrt a, {a, b, c, d}.

This example shows that FNF is not yet a solution to the problem in (1), since the vectors in FNF are not independent.

In order to eliminate the redundant fusion a∘b∘c∘d from the FNF and obtain the MFNF, a final step will be added to the algorithm in (17):

(18) **MFNF(S)**: for all nodes that contain a fusion, fuse all the vectors in all its daughters and check if this 'total daughter fusion' is the same as or entails by W extension the mother fusion. If this is the case, remove the mother fusion from the set FNF(S).

References:

- Prince, Alan 2002. *Entailed Ranking Arguments*, ROA 500.

RCD references:

- Tesar, Bruce 1995. *Computational Optimality Theory*, PhD Dissertation, University of Colorado, ROA-90.
- Tesar, Bruce and Paul Smolensky 1998. Learnability in Optimality Theory, *Linguistic Inquiry* 29.2, 229-268.
- Tesar, Bruce and Paul Smolensky 2000. *Learnability in Optimality Theory*, MIT Press.