# Ranking and Necessity

## Part I: The Fusional Reduction Algorithm

● 12-16-2005 ●

Adrian Brasoveanu and Alan Prince
Department of Linguistics
Rutgers Center for Cognitive Science
Rutgers University, New Brunswick

## ABSTRACT

Understanding a linguistic theory within OT requires an exact characterization of the ranking conditions necessitated by data. We describe (Part I) and justify (Part II) an algorithm which calculates the necessary and sufficient ranking conditions inherent in any collection of candidates and presents them in a maximally concise and informative way. The algorithm, stemming from the original proposal of Brasoveanu 2003, develops in the setting of the fusional ERC theory of Prince 2002.

## ACKNOWLEDGMENTS

The authors, whose names are arranged alphabetically, would like to thank Jane Grimshaw, Naz Merchant, Paul Smolensky, and Bruce Tesar for useful discussion; and, for helpful comments, audiences at the Rutgers Optimality Research Group (April 2004, Nov. 2005), HUMDRUM (May 2004), and the LSA 2005 Summer Institute. For support during aspects of the investigation, we thank the John Simon Guggenheim Memorial Foundation, the National Science Foundation (Grant No. BCS-0083101), and the National Institutes of Health (NRSA Training Grant No. 1-T32-MH-19975-05). *The opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of the National Institutes of Health.* The authors also claim responsibility for any errors, omissions, misprisions, or infelicities that may have crept into the text.

# Ranking and Necessity
Part I

# Ranking and Necessity
Part I: The Fusional Reduction Algorithm

Adrian Brasoveanu and Alan Prince
Rutgers University

## 1. Beyond the Sufficient

With a constraint set in hand, the first order of analytical business in OT is to match candidates with violation profiles. One of two problems then arises, depending on what is known (or assumed known) about candidate status and the constraint hierarchy.

**The Selection Problem**. If a ranking has been imposed, the problem is to determine which candidates come out as best, under that ranking.

**The Ranking Problem**. If the candidates that are desired optimal have been identified, the problem is to determine which rankings, if any, will produce the desired optima as optimal.

The Selection Problem arises when prior analysis or prior assumptions have fixed a ranking for the constraint set. Selection of optima can be obtained by a simple repeated process of filtration, conducted under the rubric "take the best, ignore the rest" (as Gigerenzer & Goldstein 1996 sharply phrase it). The best overall must lie among the best on the top-ranked constraint; and must lie among the best of *those* as evaluated by the next-highest-ranked constraint, and so on down the hierarchy, selecting the best of the best, until the optimal violation profiles emerge.

The Ranking Problem arises when the desired optima are known, as from empirical observation and analysis, but the rankings they require are yet to be discovered. The task is different in character from selection and rather more complex. It is based on comparisons of relative goodness between grammatical and ungrammatical forms — between desired optima and their suboptimal competitors. Any such comparison yields a pattern of successes, failures, and moot contests on individual constraints. Typically, only certain rankings will be consistent with the choice of desired optima. The partial ranking information gleaned from each of many instances of comparing optimum *vs.* competitor must be integrated to produce the definitive set of ranking conditions imposed by the data. Taken together, these conditions delimit the set of total orders that yield the observed grammatical forms and predict the shape of the language beyond the data considered.

In brief, selection goes from ranking to optima, and presents little difficulty. The ranking problem runs the other way, from (desired) optima to ranking, and has a logic of its own.

To illustrate these problems, and their distinctness, let's look at an example based on the Lardil analysis worked out in Prince & Smolensky 2004:140. We have re-named the constraints to accord with more recent usage, and we have omitted those constraints that award all candidates the same number of violations. The special nonphonetic notations in the candidate transcription are '*A*' for 'inserted [a]' and '■' for 'deleted segment'. They are intended only to make the key input-output disparities obvious to the eye. Syllables are marked off by periods. The numerals indicate the number of violations incurred by the candidate on the constraint named at the head of the column.

(1) **The Selection Problem: ranking given.** Lardil /kentapal/ → [????] 'dugong'

| /kentapal/ → | DEPV | ANCHOR-R | MAX | NOCODA |
|---|---|---|---|---|
| .ken.ta.pal. | 0 | 0 | 0 | 2 |
| .ke.n*A*.ta.pa.■ | 1 **!** | 1 | 1 | 0 |

The left-to-right order of the constraints mirrors their actual ranking (as shown in Prince & Smolensky 2004:132-138). The desired optimum beats its competitor definitively on DEPV, a victory marked by the appearance of the exclamation point.

To see a less trivial filtration, we need to have the optimum face off against more than one competitor. Here's an example, expanded from (1).

(2) **More elaborate selection**. Lardil /kentapal/ → ??? 'dugong'

| /kentapal/ → | DEPV | ANCHOR-R | MAX | NOCODA |
|---|---|---|---|---|
| .ken.ta.pal. | 0 | 0 | 0 | 2 |
| .ken.tap.al. | 0 | 0 | 0 | 3 **!** |
| .ken.ta.pa. ■ | 0 | 1 **!** | 0 | 1 |
| .ke.nA.ta.pa.■ | 1 **!** | 1 | 1 | 0 |

The exclamation point marks where the losing candidate is ejected from the set of 'best' candidates. Shading emphasizes that the candidate is no longer in the filtered set.

Matters look quite different when we start with knowledge that [ken.ta.pal] *is* the output from /kentapal/ and assume nothing about the constraint ranking. Let's take the same two violation profiles as in (1) and annotate the tableau according to the relative performance of the competitors on each constraint. The desired optimum is listed first and candidates desired to be suboptimal are indented below it. The mark 'W' generalizes '!" and indicates that the desired optimum is locally better on the one constraint by virtue of having fewer violations that its suboptimal competitor. The mark 'L' indicates that the *sub*optimum does better on the constraint than the desired optimum. Only the suboptimum row is annotated, for the violation profile of the desired optimum functions as a kind of yardstick against which everything else is measured.

(3) **The Ranking Problem: optimum given**. Lardil /kentapal/ → .ken.ta.pal. 'dugong'

| /kentapal/ → | DEPV | ANCHOR-R | MAX | NOCODA |
|---|---|---|---|---|
| ken.ta.pal | 0 | 0 | 0 | 2 |
| ke.nA.ta.pa■ | 1 **W** | 1 **W** | 1 **W** | 0 **L** |

This comparison shows that NOCODA must be subordinated in the ranking, because it favors the suboptimum (L), which had better do worse than the optimum over the hierarchy. It follows that NOCODA must be dominated by some constraint of opposite polarity that favors the desired optimum — and we have three different choices (marked W). Comparing outputs *ken.ta.pal* with **ke.na.ta.pa* gives us no grounds for choosing among the three potentially dominating constraints, and yields, therefore, quite weak information about what the hierarchy must be. Other comparisons will impose further, tighter constraints on the viable rankings.

With a large collection of comparisons in hand, we may ask for a characterization of the set of rankings that give the data in its entirety. The answer we will be satisfied with varies with our goals. It is commonly the case that a number of different rankings will suffice — after all, many constraints simply do not conflict with each other, and ranking serves to resolve conflicts. (Example: those constraints that are fully satisified in every optimal form can be ranked in any order whatsoever at the top of the hierarchy.) The *learner*, for whom the grammar is a black box, is looking for a ranking sufficient to produce correct output and will be happy with any such. The data modeler might be similarly satisfied, but a more ambitious analyst will want to know the details of the constraint relations and their factual basis. Such information determines the explanatory structure of a theory, and it is crucial to the dynamics of investigation and theory construction. Modifying constraint definitions, adding new constraints, amalgamating or subdividing constraints — the success of any of these theory-developing maneuvers depends on the details of the interactions. The analyst, then, must be interested in both the necessary and the sufficient conditions imposed by the data.

As Tesar found, we can easily derive a sufficient ranking from a collection of comparative data if we ask not what constraints *must* be ranked at the top but what *can* be ranked there. We gather all those rankable constraints into a 'stratum', dismiss the data they solve, and proceed to re-ask the question of the remaining, smaller set of data, obtaining a new, lower stratum. We continue in this fashion until all the data is accounted for, yielding an ordered collection of strata. The stratification procedure is known as 'Recursive Constraint Demotion' (RCD) and a grammar derived from it will work, if there exists any grammar that works. (Basic references include Tesar & Smolensky 1993, 2000; Tesar 1995.) Since the constraints in a stratum do not conflict, we can construct a workable grammar (total ordering of the constraint set) by picking any ranking order for the constraints within each stratum and patching these segments together so as to respect the stratal order. The trade-off is that while we have sufficiency, we have abandoned necessity.

A simple formal example brings out the heart of the matter. Imagine that the entire constraint set has only three members, and suppose that applying RCD to the data of a particular language delivers up the following two-stratum hierarchy:

(4) $\{C_1, C_2\} \gg \{C_3\}$

No fewer than four distinct ranking restrictions could lie behind this one stratified hierarchy.

(a) Both $C_1$ and $C_2$ must dominate $C_3$.
(b) Either $C_1$ or $C_2$ must dominate $C_3$.
(c) $C_1$ must dominate $C_3$, but $C_2$ can go anywhere.
(d) $C_2$ must dominate $C_3$, but $C_1$ can go anywhere

Of these, the latter three shows the effects of the greediness of RCD: some constraints are placed in the top stratum because they *can* be there, not because they must. The four situations differ in their explanatory structure and in the mutability of the constraints involved. In case (c), for example, the violability of $C_3$ is due entirely to the possibility of 'better satisfying' $C_1$, and it is correct to say that $C_1$ 'forces' the violation of $C_3$; but this is not true of any other pattern. Similarly, in cases (b) and (c), the constraint $C_2$ is wide open to reformulation, but not in the other systems.

In this paper, we present and justify an algorithm — 'Fusional Reduction', FRed — that pulls out the necessary and sufficient ranking conditions inherent in a set of data and puts them in a canonical form that is maximally concise and informative. (The algorithm originates in Brasoveanu 2003; here it is somewhat revised and analyzed in the setting of the theory of Prince 2002.) 'Maximally concise' means that there is no other canonical presentation that uses fewer elements. 'Maximally informative' means that each element in the description places the strongest possible conditions consistent with the data. Through FRed, we gain a complete algorithmic solution to the Ranking Problem, understood in its most demanding form.

Such matters may be addressed at various levels of technicality. We will aim to steer clear of the fine grain until the whole argument has been laid out and discussed. Our strategy will be to divide the exposition between two papers, of which the reader has now embarked upon the first. We develop the requisite notional underpinnings in the next section (§2), and then move to an exposition of the FRed algorithm and its output which is accurate but does not delve deeply into the details of justification (§3). We conclude Part I by placing FRed with respect to RCD and showing how other useful representations may be derived (§4), ending with notes on computational complexity and relationship to other work in the area. The more thorough formal analysis that establishes all claims made in this paper appears as Part II.

# 2. The Consequences of Comparison

In this section we review the logic of pairwise candidate comparison that underlies the entire calculus of ranking argumentation in OT.

## 2.1 The Elementary Ranking Condition

Extracting ranking information from even a single pairwise comparison need not be trivial. Each individual constraint awards one of three evaluations to a competing pair:

| | |
|---|---|
| W | the desired optimum is better, with fewer violations than its competitor |
| L | the suboptimum is is better, with fewer violations; the desired optimum loses to it |
| e | neither is better, because the two are equal in violations. |

We will say that a constraint 'distinguishes' two candidates if it awards one of the 'polar' values W or L, and not the 'neutral value' *e*, to their comparison. A typical result, seen in another Lardil example (P&S:140), looks like this:

(5) **Violation Tableau with Comparative Annotations:** /ŋaluk/→ ŋa.lu

| /ŋaluk/ → | CODACOND | | DEPV | | ANCHOR-R | | DEPC | | MAX | | NOCODA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| .ŋa.lu.■ | 0 | | 0 | | 1 | | 0 | | 1 | | 0 | |
| .ŋa.luk. | 1 | **W** | 0 | **e** | 0 | **L** | 0 | **e** | 0 | **L** | 1 | **W** |

The constraint CODACOND, unmentioned in (3) because neutral and therefore irrelevant, is polar here; we have also brought in a couple of neutral constraints for illustrative purposes. Since we are dealing with the Ranking Problem in this paper, we will *never* presuppose, henceforth, that the order of cited constraints reflects a ranking order, nor will we distinguish in the vertical line style of a tableau between ranked and unranked constraints.

A tableau like (5) can be reduced to its core by suppression of the violation profiles, which have done their work when the comparative values are calculated from them. The result is a pure 'comparative tableau'.[1]

(6) Comparative Tableau (CT)

| /ŋaluk/ → | CODACOND | DEPV | ANCHOR-R | DEPC | MAX | NOCODA |
|---|---|---|---|---|---|---|
| .ŋa.lu.■ ~ ŋa.luk. | **W** | **e** | **L** | **e** | **L** | **W** |

Notation: we write '*q*~z' to denote the comparison of a desired optimum *q* with its competitor z. Here we compress out the repeated input term: we are actually comparing *q*: ⟨ŋaluk→.ŋa.lu.■⟩ with z: ⟨ŋaluk → ŋa.luk.⟩, two competing input-output relations based on the same input.

Each row detailing the evaluations of some *q*~*z* comparison over the entire constraint set is associated with a ranking restriction, an 'Elementary Ranking Condition' or ERC, that always has the same logical form. In CT (6), for example, the requirement is this:
*either* CODACOND *or* NOCODA dominates *both* ANCHOR-R *and* MAX.
This condition is satisfied by every ranking in which the desired optimum ⟨ŋaluk→ ŋa.lu⟩ is judged better than the competitor ⟨ŋaluk→ ŋa.luk.⟩. More generally, an ERC holds that that *some* constraint assessing W must dominate *all* constraints assessing L.

To see why this is so, it is useful to recall how optimality is defined in terms of the two 'better than' relations. A form is optimal when it is *better than* all of its violation-distinct competitors over the whole hierarchy. This global 'better than' is in turn derived from the local, violation-sensitive judgments of the individual constraints, as represented in the W-L-e values. One candidate is globally better than another over a whole hierarchy when it is locally better on the highest-ranked constraint that distinguishes them (to use the concise formulation of Grimshaw 1997). This gives us the ERC: from the comparative profile, we can compute which constraints may serve in the decisive role as the highest-ranked distinguishing constraint (any of those assessing W), and which constraints must be absolutely banned from that position (all of those assessing L). The comparative profile also identifies those constraints that are completely irrelevant to the status of the competitors, those neutrals assessing *e*.

The essence of the comparative tableau lies in the sequences of values it assigns to comparisons. Let us recognize the relevant part of the tableau row as a distinct entity, a 'comparative vector'. Such a vector is a list of W-L-e values, with position on the list corresponding to an enumeration of the constraint set. (The enumeration is arbitrary and serves only to keep track of the constraints, in much the same sense as the sequence of entries in a vector of cartesian coordinates keeps track of the spatial dimensions, which have no intrinsic ordering.) The comparative tableau (6) displays a single vector, which we can write in the following compressed form:

(W, e, L, e, L, W)

The entries in a vector will be called its 'coordinates'.

[1] For the CT per se, see Prince 1998, 2000, 2002b. The relevant logic, presented here, is developed in Prince 2002a.

Each such vector is associated with an ERC. In constructing a representation of the ranking relations imposed by a set of data, we will take the ERC as the basic unit of description. We will see how certain operations on the coordinates of comparative vectors compute the logical properties of ERCs and ERC sets. These will enable us to define the algorithm which takes a collection of vectors and renders it into its most compact, informative equivalent.

In the interests of explicitness, we define the ERC formally here:

(7) **Elementary Ranking Condition** ($\forall\exists$ form). Let **a**=$[x{\sim}y] \in \{W,L,e\}^n$ be a comparative vector associated with $\Sigma_n$, an enumeration of the constraint set $\Sigma$. Writing

$$\mathsf{W}(\mathbf{a}) = \{C_k \in \Sigma \mid \mathbf{a}[k] = W\} \quad \text{‘those constraints assessing W of } \mathbf{a}\text{’}$$
$$\mathsf{L}(\mathbf{a}) = \{C_k \in \Sigma \mid \mathbf{a}[k] = L\} \quad \text{‘those constraints assessing L of } \mathbf{a}\text{’}$$

the ERC associated with **a** is this:

$$\forall C \in \Sigma\ \exists D \in \Sigma\ (\ C \in \mathsf{L}(\mathbf{a}) \supset (D \in \mathsf{W}(\mathbf{a})\ \&\ D \gg C)\ )$$

‘Every constraint assessing L is dominated by some constraint assessing W.’

(8) **Elementary Ranking Condition** ($\exists\forall$ form). Using the same notation,

$$\exists D \in \Sigma\ \ \forall C \in \Sigma\ \ (\ C \in \mathsf{L}(\mathbf{a}) \supset (D \in \mathsf{W}(\mathbf{a})\ \&\ D \gg C)\ )$$

‘Some constraint assessing W dominates all constraints assessing L.’

The two forms of the ERC are equivalent because the linear ordering of the constraint hierarchy means that one of W-assessors must be dominate all the others. We will use the some-all form in (8) as our standard.

The truth or falsity of an ERC is determined with respect to a ranking on the constraint set. A constraint hierarchy is then a *model* in which an ERC is judged true or false. When an ERC is true of some hierarchy H we will say, following the usual usage, that the hierarchy H *satisfies* the ERC. If all ERCs in a collection are true of a hierarchy, we will say that the hierarchy satisfies the entire ERC set.

The force of the ERC associated with the comparison *x~y*, then, is that *x* is *better than y* on H if and only if H satisfies the associated ERC. (From the selectional point of view, a hierarchy that satisfies an ERC is one that successfully dismisses the competitor desired suboptimal.) Similarly, if every member of a set of ERCs is simultaneously satisfied by a hierarchy, then that hierarchy guarantees that every comparison behind the ERC set comes out in favor of the desired winner..

Since ERCs and ERC vectors stand in one-to-one relation, let us simplify the discussion by extending the usage of ‘satisfies’ to encompass both entities. We will speak of a hierarchy *satisfying* an ERC *vector*, where this means that the associated ERC is satisfied. It is possible to make this usage central, rather than extended, through a direct model-theoretic account that gives the ERC vector the status of an expression in a formal language, directly interpreted. Since it is useful to have the ERC available as an expression in logical syntax, we will defer this option.

For purposes of reference, we conclude with a concise record of the notions we have referred to in deriving the ERC.

(9) **Constraint.** A constraint assigns violations to candidates; thus, a constraint is a function from the universal set of candidates U to the nonnegative integers. $C{:}U \rightarrow N$.

(10) **Constraint hierarchy.** Given a set $\Sigma = \{C_1,\ldots C_n\}$ of constraints, a grammar or 'constraint hierarchy' is a total order, written $\gg$, on the constraint set.

(11) **'Better than' on a constraint**. For $x,y \in U$, and for a constraint $C\in\Sigma$, we say '*x* is better than *y* on C', writing $x \succ_C y$, iff $C(x)<C(y)$.

(12) **'Better than' on a constraint hierarchy.** For H, a total ordering of $\Sigma$, and $x,y\in U$, we say '*x* is better than *y* on H', writing $x \succ_H y$ , iff (a) $\exists C\in\Sigma$ such that $x \succ_C y$; and (b) $\forall D\in\Sigma$ if $D\gg C$, then $D(x)=D(y)$, or equivalently, (b′) $\forall D\in\Sigma$ if $y \succ_D x$, then $C\gg D$. That is: there is a constraint distinguishing *x* and *y*, and the highest-ranking such constraint favors *x*.

(13) **Optimal**. For $x\in K$ and a constraint hierarchy H, a candidate set, x is 'optimal' on H iff $\neg\exists y\in K$ such that $y \succ_H x$. 'No candidate is better'.

(14) **Comparative Values**. From the violation vectors associated with candidates x,y, whose k$^{th}$ coordinates are given by $C_k(x)$ and $C_k(y)$, we construct the comparative vector [x~y], by defining for each k, $C_k'$ [x~y] = cmp($C_k(y)–C_k(x)$), where cmp(n) = W if n>0; *e* if n=0, L if n<0. This transforms $C_k:U\rightarrow N$ into $C_k':N\times N\rightarrow\{W,L,e\}$.[2]

(15) **Satisfying a Vector**. Given a hierarchy H and an ERC vector **a**, we will say the H satisfies **a**, writing H⊨**a**, iff H satisfies the ERC associated with **a**; that is, iff H is such that some constraint in W(**a**) dominates every constraint in L(**a**).

## 2.2 Informativeness

The same rankings may be demanded by different ERC sets. The simplest nontrivial case involves only three constraints, linearly ordered, and shows quite clearly how this can happen. We use bolded numbers to index constraints, and bolded alphabetic characters to name ERC vectors. The same character, unbolded and bracketed, names the associated ERC.

(16) **Equivalent ERC Sets: 1≫2≫3**

| $CT_1$ | **1** | **2** | **3** |
|---|---|---|---|
| **a** | **W** | **L** | **L** |
| **b** | e | **W** | **L** |

| $CT_2$ | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | **W** | **L** | e |
| **b** | e | **W** | **L** |

| $CT_3$ | **1** | **2** | **3** |
|---|---|---|---|
| **a″** | **W** | **L** | **W** |
| **b** | e | **W** | **L** |

Taken individually, the first-row vectors **a**, **a′**, and **a″** are not equivalent, as is evident from their associated ERCs.

[a] : **1≫2** & **1≫3**

[2] It is natural to identify 'cmp' with 'sgn', the function that returns +1 for positive arguments, 0 for 0, and −1 for negative arguments (Prince 2002:55ff., 100ff.) This portrays a comparative vector as a vector over GF3, where in fact the relations and operations defined in the next section can be stated algebraically. See fn 6. In the interests of retaining a palpable connection with the winner-loser structure of OT, we will use the standard symbols W,L,e.

[a′]: **$1 \gg 2$**

[a″]: **$1 \gg 2 \vee 3 \gg 2$**

Nonetheless, in the context of vector **b**, which requires **$2 \gg 3$**, these three all yield exactly the same conclusion: sets CT$_1$, CT$_2$, and CT$_3$ are satisfied by one and the same linear order on the constraint set, and by no other.

Further collections of ERCs can yield the same result: for example, the union of all three cited sets; the union of any two of them; numerous other admixtures and augmentations. We may freely add in vectors which consist solely of W's and *e*'s: in such cases, no constraints disfavor the desired winners, and no ranking is required to make them win; such vectors place no restrictions and may be added or taken away from any ERC set without changing its content.

All of these variants are commonly encountered. Here's a 4-constraint case from Lombardi's invaluable voicing typology paper (Lombardi 1999:286). For clarity, we suppress *e* in favor of blankness.

(17) **Lombardi's Analysis of Swedish**

| | AGREE | IDLAR | *LAR | IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | **W** | **L** |
| **b**. skuːg → skuːg ~ skuːk | | **W** | **L** | |
| **c**. vigsəl → viksəl ~ vigsəl | **W** | **L** | **W** | |
| **d**. vigsəl → viksəl ~ vigzəl | | | W | W |
| **f**. ægde → ægde ~ ække | | W | L | W |
| **g**. stekde → stekte ~ stekde | W | L | W | L |
| **h**. ægde → ække ~ ække | W | | L | |

Lombardi is considering the voicing patterns in obstruent clusters and aims to show that her constraint system provides a grammar for the kind of pattern seen in Swedish. If we are looking for the ranking, only a proper subset of the cases need be considered. (If, like Lombardi, we are arguing that the desired optima are selected as optimal, we must deal with entire candidate sets.)

The required ranking, which is total and follows the listed order of constraints in the tableau, is determined by any of these ERC sets (as well as many supersets of them):

**a**, **b**, **c**
**a**, **b**, **g**
**a**, **f**, **c**
**a**, **f**, **g**

For illustrative purposes, tableau (17) distinguishes the first of these.

A couple of other remarks: vector **d** is completely uninformative, since the desired optimum harmonically bounds its competitor and will win on any ranking. (Harmonic bounding of this sort is an important fact about the way the constraint set operates to exclude certain items universally, but it does not help with the Ranking Problem.) And even though it does not involve harmonic bounding, row **h** is also uninformative in context, since its content follows from transitivity, via the collective force of rows **b** and **c**.

Since our target is the most concise and informative representation of the ranking conditions, we will be interested only in those ERC vector sets of smallest size. Tableau (17) has 7 vectors; all are distinct, but only 3 are needed to encapsulate the tableau's ranking restrictions. Let us use the term 'basis' to refer to any minimal-cardinality ERC vector set that fully determines the rankings required by some designated set, which we will call the 'reference set'. There are quite a few three-member bases for Lombardi's Swedish hierarchy (27 in total; tableau (17) contains 4 of them), and among these yet further distinctions may be made. Since these distinctions are fully exemplified in simpler form in the abstract case we began with, which has 3 rather than 4 constraints in linear order, let us focus on that.

Two basic observations about the 3 constraint situation (16):

- The ranking relation **2**$\gg$ **3** can come from just one vector, **b**: (e, W, L).
- The ranking relation **1**$\gg$**2**, in the context of **b**, has 3 distinct possible sources:

| | | |
|---|---|---|
| **a** : | (W,L,L) | **1**$\gg$**2** & **1**$\gg$**3** |
| **a′**: | (W,L,e) | **1**$\gg$**2** |
| **a″**: | (W,L,W) | **1**$\gg$**2** $\vee$ **3**$\gg$**2** |

Vector **a** is the most informative in the sense that it spells out every condition on the ranking of constraint **1**, while the others either omit a requirement (**a′**) or hide the truth among a disjunction that turns out to be misleadingly weak (**a″** ).

Logically speaking, the informativeness relation among these ERCs is that of *entailment*. ERC [a] entails ERC [a′] in exactly the same sense that 'p&q' entails 'p' — every condition under which the first of these is true is also a condition under which the second is true. In model talk: every model that satisfies the first also satisfies the second. Similarly, ERC [a′] entails ERC [a″] in the same way that 'p' entails 'p$\vee$q'. Informativeness, then, *is* entailment: a stronger, asymmetrically entailing ERC is more informative than its entailee. Since entailment is based on satisfaction, it extends just as directly to ERC vectors. We will take entailment, thus extended, to be a relation among vectors as well, exactly tracking the relation among their associated ERCs.

The informativeness relation holds meaningfully among vectors arising from a given reference set, which is what they are being informative about. If you know 'p&q' you know more than if you know 'p' alone — so long as 'p' and 'q' are both worth knowing in the first place.[3] In particular, we are only interested in vectors that are entailed by the reference set, because we seek a characterization of its logical consequences.

There is then a hierarchy of informativeness among **a**, **a′**, and **a″**, running from more to less, based on asymmetric entailment. This local hierarchy propagates to the bases containing them. The three different bases for the **1**$\gg$**2**$\gg$**3** case can be rated according to their relative informativeness. All must contain exactly two elements and one of those elements must be (e,W,L), but the second element must be chosen from among

**a**: (W, L, L) **a′** :(W, L, e) **a″**: (W,L,W)

Clearly, we choose **a**, since it entails the others, but not vice versa.

The minimal cardinality of the *basis* also has an effect on entailment relations within it: no such relations can exist. If one basis element were entailed by others, then it could be removed without loss of information, which means that the original putative 'basis' didn't have minimal cardinality and couldn't have been a basis at all.

Our overall goal can now be made quite concrete: given a set of ERC vectors, we wish to obtain that vector set giving back the entirety of required rankings which is most concise (smallest) and most informative (containing the strongest elements, entailmentwise). This we call the 'most informative basis' (MIB) for the original set. Its uniqueness is proved in Part II.

[3] For example, '2+2=5' entails everything, but 'p & (2+2=5)' is not notably more informative than 'p' by itself.

For Lombardi's Swedish as evidenced in (17), the MIB will contain exactly these 3 elements:

| | |
|---|---|
| $\mathbf{b_1}$: (e,   e, W, L) | $\mathbf{3 \gg 4}$ |
| $\mathbf{b_2}$: (e,  W, L, L) | $\mathbf{2 \gg 3}$ & $\mathbf{2 \gg 4}$ |
| $\mathbf{b_3}$: (W, L, L,  L) | $\mathbf{1 \gg 2}$ & $\mathbf{1 \gg 3}$ & $\mathbf{1 \gg 4}$ |

Other bases exist, but they all contain weaker elements. For example, $\mathbf{b_2}$ can be replaced by (e, W, L, e) or (e, W, L, W), but their associated ERCs are asymmetrically entailed by that of $\mathbf{b_1}$, in exactly the same way as discussed above: the first omits a requirement present in $\mathbf{b_2}$ and the second involves a disjunction that is cancelled out in the broader scheme of things.

Notice that there is no restriction that the MIB, or any other basis, be a subset of the original set. We have left behind the notion of the ERC vector as the creature of data observation and advanced to regarding it as formal expression, with a certain interpretation, in which guise it serves as the essential element for representing ranking requirements.

For purposes of explicitness, we record the notions involved in the discussion.

(18) **Entailment among vectors.** Let **A** be a set of ERC vectors, and **b** be an ERC vector, ERCs. Then $\mathbf{A} \vDash \mathbf{b}$ iff every ranking that satisfies **A** also satisfies **b**.

(19) **Informative**. Let **A** be an ERC vector set, and **b**,**c** be ERC vectors.. Suppose also that $\mathbf{A} \vDash \mathbf{b},\mathbf{c}$. Then '**b** is more informative than **c** about **A**' iff $\mathbf{b} \vDash \mathbf{c}$ but not vice versa.

(20) **Basis**. Suppose **B** is such that $\forall x \in \mathbf{A}$, $\mathbf{B} \vDash \mathbf{x}$. Further suppose there is no set **C** with the same property, such that $|\mathbf{C}|<|\mathbf{B}|$. Then we say that **B** is a *basis* for **A**.[4]

(21) **Most Informative Basis (MIB)**. Let **B** be a basis for **A**. **B** is the MIB for **A** iff for every $\mathbf{b} \in \mathbf{B}$ whenever $\exists x$ s.t. $\mathbf{A} \vDash \mathbf{x}$ and $\mathbf{x} \vDash \mathbf{b}$, then $\mathbf{x} = \mathbf{b}$.

## 2.3 ERCs and Consequences

Defining and finding the Most Informative Basis turns on the notion of entailment between ERCs. Given an ERC set **A**, the elements of a basis are drawn not just from **A**, but from the entire set of ERCs entailed by **A**. The relation of informativeness between ERCs is based on entailment. The elements of a basis are logically independent, in the sense that no one of them is entailed by any collection of others.

The relation of entailment between ERCs has been studied in Prince 2002 (*Entailed Ranking Arguments*; henceforth ERA). Here we present the basic findings that underlie the present investigation and move forward to develop the key ideas that lead to the fusional reduction algorithm.

Entailment relations fall naturally into two types. The simplest involves entailments following from a single ERC. More complex is the relation between a set of ERCs and its consequences. We will see that the second reduces to the first, when an appropriate method of combining ERCs, 'fusion', is introduced.

[4] It is also possible to define *basis* in terms of logical properties and derive its minimal cardinality. For purposes of the present paper, we settle for the direct approach.

Let us first examine single ERC entailment. Given an ERC vector that contains at least one L coordinate, we can derive an entailment from it by removing an L and replacing it with an *e*. We write '→' between vectors connected by this operation:

(22) **L-retraction**

$$\begin{array}{lcl} (W, L, L) & \rightarrow & (W, L, e) \\ \mathbf{1 \gg 2\ \&\ 1 \gg 3} & \vDash & \mathbf{1 \gg 2} \end{array}$$

Similarly, if we replace an *e* with a W, we also obtain a legitimate entailment. Again, we signify the relation between vectors with '→'.

(23) **W-extension**

$$\begin{array}{lcl} (W, L, e) & \rightarrow & (W, L, W) \\ \mathbf{1 \gg 2} & \vDash & \mathbf{1 \gg 2} \vee \mathbf{3 \gg 2} \end{array}$$

Putting aside those 'trivial' vectors that are satisfied in every model or in none, it turns out that *every* entailment from a single ERC arises from a sequence of L-retractions and W-extensions (ERA:6, Prop 1.1).[5]

The trivial vectors deserve a moment's notice. These lack the familiar configuration in which both W's and L's are present, and they therefore do not express a contentful ranking relation. They come in two sorts, valid and invalid. (1) Valid. Those lacking L impose no ranking requirements, because there is no constraint that must be subordinated; they are therefore satisfied by any hierarchy and universally valid. (2) Invalid. Those lacking W but containing L, which no ranking can satisfy, because they contain no legitimate dominator to satisfy the requirement that L be subordinated.

To provide for convenient reference to these entities, let us designate the set of all valid vectors of any length by the name W*, mnemonically marking their composition from W's and *e*'s. The set of invalid vectors we will designate $L^+$, indicating that its members contain at least one L (but no W's).

Trivial vectors, because of their indiscriminate logic, misbehave with respect to L-retraction and W-extension. Invalid statements entail anything, and anything entails a valid statement. But L-retraction and W-extension are more particular. For example,

(24) $(e, L) \nrightarrow (L, W)$ *but* $(e, L) \vDash (L, W)$ : *ex falso quodlibet.*

The relationship $e \rightarrow L$ in the first coordinate is not sanctioned by L-retraction. Similarly,

(25) $(W,L) \nrightarrow (e, W)$ *but* $(W,L) \vDash (e, W)$. : *verum ex quodlibet.*

Here the first-coordinate relationship $W \rightarrow e$ runs in the wrong direction, but has no effect on entailment.

L-retraction and W-extension can be brought under one roof if we see the entailment relation as being based on an order. Assume the scale: $L<e<W$.[6] Then we can say that **a**

[5] Recall that a model is a grammar or total ranking of the constraints (p. 8), yielding the semantics of satisfaction (see ex. (15), p. 9). We are thus taking a model-theoretic perspective on the logic of natural language grammar. The centrality of entailment in OT metatheory attests to the utility of the approach, here and in linguistics generally.

[6] Tthe arithmetization of fn. 2 (with $L=-1$, $e=0$, $W=1$) would serve handily. See Meyer 1975:400 for the original introduction of the scalar idea for interpreting arrow and related connectives; and ERA:55-57 for discussion.

'arrows' **b** by definition iff every coordinate of **a** is less than or equal to the corresponding coordinate of **b**. Our fundamental result identifies the 'arrowing' relation as the perfect mirror of semantic entailment, as long as trivial ERCs are not involved.

(26) **Individual Nontrivial Entailment ⇔ Arrow**. For **a**,**b** nontrivial, **a**→**b** iff **a**⊨**b**.
Proof. ERA:6, Prop 1.1.

Observe that moving from 'arrow' to entailment is always legitimate, regardless of triviality. The arrow relation *refines* entailment, adding further structure but never contradicting it.

Entailments may follow from a set of ERCs that do not follow from any individual member of the set. For example, consider this pair:

(27) **1≫2 and 1≫3**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | L | e | **1≫2** |
| **b** | W | e | L | **1≫3** |

Neither **a** nor **b** independently yields the conclusion that **1** must dominate both **2** and **3**. In this kind of case, there is a vector that exactly encapsulates the set, namely (W,L,L). But when transitivity of ranking order is involved, such a handy reduction will not be available. Consider the following, the simplest case where transitivity asserts itself.

(28) **1≫2≫3**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | L | e | **1≫2** |
| **b** | e | W | L | **2≫3** |

From these two vectors, it follows that **1≫3**. But the vector expressing this requirement is not individually arrowed (or entailed) by either **a** or **b**:

(29) **Failures of simple arrowing**.

| | Non-arrowed | Non-entailed | Cause |
|---|---|---|---|
| **a**: | (W, L, e) ↛ (W, e, L) | **1≫2** ⊭ **1≫3** | e↛L in **3** |
| **b**: | (e, W, L) ↛ (W, e, L). | **2≫3** ⊭ **1≫3** | W↛e in **2** |

Furthermore, there is no way to combine **a** and **b** into a third vector that expresses their full content. The reason is straightforward: the totality of information in **a** and **b** is represented in the logical conjuction of their associated ERCs, and this expression is simply not of the ERC form.

(30) **Conjunction**. [a]&[b] = **1≫2** & **2≫3**

In this conjunction, constraint **2** is identified both as a dominator (W) and a dominee (L). But any ERC must treat the W and L classes as disjoint. An ERC relates two disjoint classes of constraints; transitivity mentions three constraints falling into two overlapping classes.

What's needed is a different kind of logical combination, one which may sacrifice some information, but preserves enough to support the calculation of entailments. The appropriate method is 'fusion', which combines values according to the following scheme:

(31) **Fusion of Values**.

| | | |
|---|---|---|
| $L\circ X = X\circ L = L$ | | 'L is dominant' |
| $e\circ X = X\circ e = X$ | | 'e is identity' |
| $W\circ W = W$ | | 'fusion is idempotent: more generally, $X\circ X = X$' |

We write X∘Y for 'X fused with Y'. To get the fusion of multi-coordinate vectors, we apply the operation to corresponding coordinates, paralleling the way vectors are added. As is customary in such situations, we use the same symbol for the derived operation as for the basic one.

(32) **Fusion of Vectors**. Fuse coordinatewise. Writing **v**[k] for the kth coordinate of **v**, we define the coordinates of the vector **a**∘**b** in terms of the fusion of its coordinates:

$$\mathbf{a}\circ\mathbf{b}[k] =_{def} \mathbf{a}[k]\circ\mathbf{b}[k]$$

Like more familiar operations such as *and*, *or*, *plus*, and *times*, the order of fusion of multiple entities is immaterial; we may therefore easily extending it to sets.[7] We will write $f$**A** for the fusion of a nonempty set of ERC vectors **A**, with the convenient proviso that $f$\{**v**\}= **v** for singleton sets.

Returning to case (28) with fusion in hand, we can now derive the desired collective consequence. First, we fuse the two vectors:

(33) **Fusion and Transitivity**

| | **1** | **2** | **3** |
|---|---|---|---|
| **a** | W | L | e |
| **b** | e | W | L |
| **a∘b** | *W* | *L* | *L* |

The vector requiring **1**≫**3** is arrowed, and therefore entailed, by the fusion **a**∘**b**:

(W, L, L ) → (W, e, L) *nb*: L<e in **2**.

Fusion has delivered a vector that allows the collective consequence to be derived from the rules for individual entailments. This pattern turns out to be entirely general.

The single fusion-derived vector is telling us something about ranking requirements that the entire set also tells us. Fusion respects the content of the set of vectors that are fused. More important: applied to subsets of a given set, it suffices, provably, to recover any ranking requirement imposed collectively by members of that set.

To put the content-respecting property more exactly: the fusion of any set of vectors is entailed by that set, so that fusion never leads us outside the realm of consequences of the original set. (This is a 'closure' property, since the set of entailed vectors is closed under fusion.)

(34) **Closure**: **Fusion respects entailment**. **A** ⊨ $f$**A**.

Proof. ERA:10, Prop. 2.1.

[7] Fusion is commutative by definition and associativity follows from a straightforward calculation.

The closure property is true of a variety of possible combining operations, some uselessly weak — for example, if we replaced X-combine-L with *e*, yielding a valid ERC, the entailment would hold vacuously (*verum ex quodlibet*). But fusion has great strengths: it reduces the collective entailment problem to single ERC entailment. For any ERC at all entailed by a set **A**, we are guaranteed that there is a subset of **A** that fuses to entail it.

(35) **Entailment by Fusion**. Let **v** be an ERC vector and **A** a set of ERC vectors.

If $\mathbf{A}\vDash\mathbf{v}$, then for some $\mathbf{X}\subseteq\mathbf{A}$, $f\mathbf{X}\vDash\mathbf{v}$.

Proof. ERA:14, Prop. 2.5.

Even better, for all nontrivial entailments, we have a perfect match between entailment and W-extension/L-retraction — i.e., the 'arrow' ordering relation on vectors.

(36) **General Nontrivial Entailment ⇔ Arrow**. Let **v** be a nontrivial ERC vector and **A** a satisfiable set of ERC vectors.

$\mathbf{A}\vDash\mathbf{v}$ iff for some $\mathbf{X}\subseteq\mathbf{A}$, $f\mathbf{X}\rightarrow\mathbf{v}$.

Proof. RL from (26) and (34). LR from (35) and (26). See ERA:6, 10, 14.

Only the left-to-right direction — entailment to arrow — actually needs the caveats about nontriviality and satisfiability. This reflects the mismatch, noted above, between the hyperparticularity of arrow and the nonparticularity of entailment with respect to relations involving valid and invalid vectors. Compare the observations in (24) and (25), which identify particular instances of divergence.

The arrow relation, however, is a sturdily reliable guide to entailment, without qualification:

(37) **Arrow ⇒ Entailment** (all vectors)

If for some $\mathbf{X}\subseteq\mathbf{A}$ we have $f\mathbf{X}\rightarrow\mathbf{v}$, then $\mathbf{A}\vDash\mathbf{v}$ .

Proof. Follows directly from ERA:10, Prop. 2.1.

Fusion, then, is pretty much as strong as possible. With these results, we have completely transported the entailment problem from the semantics of ranking arguments into the domain of vectorial operations and relations.

To get a glimpse of the utility of fusion, let's return to the Lombardi analysis, gathering some highly disjunctive ERCs that nonetheless jointly and completely determine the ranking.

(38) **A basis for Lombardi's Swedish**

| | **1**:AGREE | **2**: IDLAR | **3**: *LAR | **4**: IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | W | L |
| **f**. ægde → ægde ~ ækte | | W | L | ***W*** |
| **g**. stekde → stekte ~ stekde | W | L | ***W*** | L |

Each ERC-internal disjunction (marked ***W***) suggests an illusory explanation for the optimality of a correct form. For example, comparison **e** taken by itself allows for **4**:IDONSLAR to be regarded as explaining the choice of [ægde]. But comparison **a** shows that this cannot be right; it is only **2**:IDLAR that can be doing the job. Fusion allows us to suppress the nonviable disjunctions (bolded and italicized) that come with this selection of data.

(39) **Relevant Fusions**

| | **1:**AGREE | **2**: IDLAR | **3**: *LAR | **4**: IDONSLAR |
|---|---|---|---|---|
| **a**. | | | W | L |
| **a∘f** | | W | L | ***L*** |
| **a∘f∘g** | W | L | ***L*** | L |

We know from (34) that any fusion expresses a ranking restriction inherent in the original data. Those displayed here are particularly useful, since the last two rows are individually stronger than the corresponding rows in the original basis (38), and resolve the ranking problem with maximal local information.

| (40) | Fusion | Associated ERC |
|---|---|---|
| | **a∘f** | **2**:IDLAR ≫ {**3**:*LAR , **4**:IDONSLAR} |
| | **a∘f∘g** | **1**:AGREE ≫ {**2**:IDLAR, **3**:*LAR , **4**:IDONSLAR} |

By selective fusion, we have created a new basis — the Most Informative Basis — which in this case has no disjunctions at all. It is immediate from inspection of the tableau that a total order on the constraint set is mandated.

The flipside of entailment is inconsistency/unsatisfiability. If p entails q, then p and ¬q are inconsistent — and vice versa. (Example: being square entails having corners, so you can't have a round square.) An ERC set may be internally inconsistent, with no hierarchy satisfying it. This is no exotic mischance: in the course of research, assumptions about the constraint set or linguistic structure frequently reveal themselves as nonviable because they give rise to unsatisfiable ERC sets. Similarly, much of Tesar's learnability work depends on inconsistency detection to reject mistaken hypotheses (Tesar 1995 *et seq.*).

Fusion detects inconsistency unerringly: a set of ERC vectors is unsatisfiable if and only if it contains a subset that fuses to some member of $\mathsf{L}^+$, a vector with no W's.

(41) **Failure** / **Fusion**. No hierarchy satisfies **A** iff for some $\mathbf{X}\subseteq\mathbf{A}$ and some $\lambda\in \mathsf{L}^+$, $f\mathbf{X}\rightarrow\lambda$.
Proof. ERA Prop. 2.4, p. 11.

To see how this works, consider the following example:

(42) **A** = {**a**,**b**}

| | **1** | **2** | **3** | **4** | ERC |
|---|---|---|---|---|---|
| **a** | W | L | W | L | (**1**≫**2** & **1**≫**4**) ∨ (**3**≫**2** & **3**≫**4**) |
| **b** | L | W | L | W | (**2**≫**1** & **2**≫**3**) ∨ (**4**≫**1** & **4**≫**3**) |
| **a∘b** | *L* | *L* | *L* | *L* | ∃**x**∈Ø s.t. **x** ≫{**1**,**2**,**3**,**4**} |

ERC-wise, it will take some tangled work with the distributive law to crank out the conclusion that no ranking can satisfy both vectors. But fusion delivers the fact directly. The ERC associated with the fusion is invalid, unsatisfiable by any hierarchy,

Why does fusion work? Formally, the response might be — because it does, and provably. Qualitatively, though, we can see that it has the virtue of preserving *necessary* subordination (through the dominance of L) and *possible* domination (through the dominance of W when L is not around). A major effect of this strategy of preservation is to capture the consequences of transitivity. In the simplest case, this is straightforward: if **a** says **1**$\gg$**2** and **b** says **2**$\gg$**3**, then both **2** and **3** are necessarily dominated; fusion records this fact and thereby obtains the transitivity of the domination relation (**1**$\gg$**3**).

From this point of view, it is natural to expect that an unsatisfiable set should contain a subset that fuses to $L^+$. It is transitivity that produces contradiction between inconsisent ranking requirements, yielding a cycle of required constraint dominations. In such a cycle, every constraint is necessarily dominated, earning L somewhere, eliminating W and *e* from its fusion.

Fusion is not, however, a panacea that automatically and indiscriminately resolves all problems. In order to retrieve the consequences of transitivity, the value *e* must serve as an identity, to preserve possible domination in W∘e. This leads to cases where fusional combination weakens informativeness. Consider the following, in which fusion produces an ERC that is entailed by both fusands, and entails neither.

(43) **When fusion weakens**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | e | L | **1**$\gg$**3** |
| **b** | e | W | L | **2**$\gg$**3** |
| **a∘b** | *W* | *W* | *L* | **1**$\gg$**3** ∨ **2**$\gg$**3** |

Two complementary questions then arise: when does fusion preserve information exactly? when does fusion lose information? These are equivalent to asking when fusion is identical to conjunction. Conjunction is entirely information-preserving, since a conjunction entails both conjuncts (and the conjuncts jointly entail the conjunction.) To answer the question, then, we need only ascertain the circumstances in which the fusion is guaranteed to entail both fusands (or, complementarily, when it fails to entail one of them). This is the crucial pair of relations

$$\mathbf{p} \circ \mathbf{q} \rightarrow \mathbf{p}$$
$$\mathbf{p} \circ \mathbf{q} \rightarrow \mathbf{q}$$

A quick review of the rules for fusion reveals these considerations:

● in any coordinate where at least one of **p** or **q** has L, both entailments are secure:

$\mathrm{L}\circ\mathrm{X} \rightarrow \mathrm{Y}$ *because* 'L arrows anything'

● in any coordinate where both **p** and **q** have W, the two entailments are secure:

$\mathrm{W}\circ\mathrm{W} \rightarrow \mathrm{W}$ *because* 'W arrows W'

● similarly where both are *e*:

$e\circ e \rightarrow e$ *because* '*e* arrows *e*'

This leaves the situation where one vector has e and the other has W. Here arrow fails from the fusion to the *e*-bearing vector:

$e\circ\mathrm{W}=\mathrm{W}$ *so* $\boldsymbol{e}\circ\mathrm{W} \nrightarrow \boldsymbol{e}$ $(\mathbf{p}\circ\mathbf{q} \nrightarrow \mathbf{p})$

Fusion becomes equivalent to conjunction, and fully-information preserving, when a W in the fusion comes only from the fusion of W's in the fusands.[8] This is true not just of pairs, but also of arbitrary sets. When every W in the fusion of the whole comes only from W's, the fusion of the whole — a single vector — encapsulates everything that the entire set has to say. In ERA, this situation is called 'W-compliance'. To see the power of fusion in a W-compliant environment, consider the following:

(44) **W-compliance**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | W | L | **1**$\gg$**3** $\vee$ **2**$\gg$**3** |
| **b** | W | L | W | **1**$\gg$**2**$\vee$ **3**$\gg$**2** |
| **a**∘**b** | *W* | *L* | *L* | **1**$\gg$ {**2**,**3**} |

Here again we face a struggle with the distributive law if we wish to assault the consequences of {**a**, **b**} within propositional logic. But the only W in the fusion derives entirely from fusand W's, signaling W-compliance, so that fusion delivers not just a consequence, but a complete equivalent. We may discard both fusands in favor of their fusion and lose nothing.

When W-compliance fails in at least one coordinate, due to the collocation of W's and *e*'s there, the fusion is doomed to lose information. A dead loss, as it were, occurs when the fusion entails none of its component fusands. A simple example illustrates the typical outcome:

(45) **Failure of W-compliance I: increase in disjunctivity**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | e | L | **1**$\gg$**3** |
| **b** | e | W | L | **2**$\gg$**3** |
| **a**∘**b** | *W* | *W* | *L* | **1**$\gg$**3** $\vee$ **2**$\gg$**3** |

To recognize the crucial points of departure from W-compliance, let's use the term 'info loss configuration' to describe any constraint column that contains both W and *e* but no L's — equivalently, any coordinate in the vector set that fails to be W-compliant under fusion. In example (45), every W-coordinate in **a**∘**b** belongs to an info loss configuration. In such circumstances it is pointless to form the fusion.

In other circumstances, as we've seen, the loss of information can be compensated for by a local increase in informativeness. Consider the following:

(46) **Failure of W-compliance II: compensatory increase in conjunctivity**

| | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | L | e | **1**$\gg$**2** |
| **b** | e | W | L | **2**$\gg$**3** |
| **a**∘**b** | *W* | *L* | *L* | **1**$\gg$**3** $\vee$ **1**$\gg$**3** |

[8] The converse is challenged only when both of the fusands are valid. Consider the simplest possible case, with just one coordinate: e∘W $\not\twoheadrightarrow$e, yet e$\vDash$W.

Although the fusion loses all information about constraint **2** from **b**, it provides a more complete account of the ranking restrictions on contraint **1** than anything in the original set. The fusion asymmetrically entails one of its fusands, a desirable increase in informativeness. This means that we may replace the entailed fusand **a** with the more informative fusion of the pair, but we must still keep the unentailed **b** around because fusion obliterates its content.

Let us designate as 'fusionally reducible' any subset of vectors that can be replaced in this way: any subset whose members are each entailed the fusion of the whole. It's clear that W-compliant subsets are fusionally reducible. But the phenomenon is more general. We are interested not in isolated sets, but in subsets of some fixed reference set. Fusional reducibility is a property of subsets in the context of the whole set under consideration. Strikingly, a subset that is not internally W-compliant can still be reducible in that broader context, if every info loss configuration within the subset fuses to L in the fusion of the whole. Trivial vectors aside, a subset of vectors is fusionally reducible in the context of a superset if and only if every member of the subset is *pairwise* W-compliant with the fusion of the superset.

Here's an example: the focus is on the behavior of {**a**,**b**} within {**a**,**b**,**c**}. Note particularly coordinate **4** and vector **b**.

(47) **Fusionally reducible in context, but not internally W-compliant**

| **A** | **1** | **2** | **3** | **4** | ERC |
|---|---|---|---|---|---|
| **a** | W | L | e | **W** | **1**≫**2** ∨ **4**≫**2** |
| **b** | W | e | L | ***e*** | **1**≫**3** |
| **c** | e | W | e | L | **2**≫**4** |
| **a**∘**b**∘**c** | **W** | **L** | **L** | **L** | **1**≫{**2**,**3**,**4**} |

The fusion of the whole set can replace the subset {**a**,**b**}without loss of entailments. Observe that the fusion is, as expected in such cases, pairwise W-compliant with each of its components. But the internal fusion **a**∘**b** of the subset cannot legitimately stand for its fusands, due to the info loss configuration in constraint **4**:

(48) **Subset fusion, with info loss configuration**

| | **1** | **2** | **3** | **4** | ERC |
|---|---|---|---|---|---|
| **a** | W | L | e | **W** | **1**≫**2** ∨ **4**≫**2** |
| **b** | W | e | L | **e** | **1**≫**3** |
| **a**∘**b** | *W* | *L* | *L* | ***W*** | **1**≫{**2**,**3**} ∨ **4**≫{**2**,**3**} |

The fusion can replace **a** but not **b**. The key successes and failures are tabulated here:

(49)

| Arrow | Notable Coordinate | Remark |
|---|---|---|
| **a**∘**b** → **a** | **4**:W→W | **a** entailed |
| **a**∘**b** ↛ **b** | **4**: W↛e. | **b** *unentailed* |

In the context of the entire set, though, all weaknesses disappear:

(50)

| Arrow | Notable Coordinate | Remark |
|---|---|---|
| **a**∘**b**∘**c** → **a** | **4**:L→W | **a** entailed |
| **a**∘**b**∘**c** → **b** | **4**:L→e | **b** entailed |

But the fusion of the whole is not *equivalent* to the subset {**a**,**b**}. It is stronger, since it imposes a relation between **1** and **4** of which the subset is ignorant. The additional requirement **1**≫**4** is collectively mandated via transitivity involving constraint **2** as expressed in vector **c**. It is only vector **c** that provides the rationale for dismissing the illusory hope, asserted disjunctively in **a** and not contradicted in **b**, that constraint **4** is fit to serve as a possible dominator.

We may therefore replace {**a**,**b**} with the fusion of the whole. The resulting vector set is equivalent to the original set, with an increase in both conciseness and informativeness.

(51) **Fusionally reduced set**

| **A′** | **1** | **2** | **3** | **4** | ERC |
|---|---|---|---|---|---|
| **a**∘**b**∘**c** | W | L | L | L | **1**≫{**2**,**3**,**4**} |
| **c** | e | W | e | L | **2**≫**4** |

The set of ERC vectors presented in (51) is the smallest that can fully recover the content of the original set, and its component vectors are individually as informative as possible. We have therefore calculated the Most Informative Basis (MIB). Let us now turn to the full fusional reduction algorithm, which takes the techniques used selectively here and forms them into a procedure that is guaranteed to the produce the MIB of any satisfiable set.

In this section, we have developed the techniques that yield a syntactic or algebraic counterpart of the essentially semantic (model-theoretic) notion of entailment. Fusion is the key operation on sets of vectors. Its result contains the maximum possible number of L's, picking up every L in the set, identifying every necessarily subordinated constraint, and thereby making the fusion potentially more informative than any single vector in the set. Among the remaining coordinates, the fusion also collects every W, identifying all potential dominator constraints among those that needn't be subordinated. Fusion respects entailment and allows us to reduce entailment from any *set* of premise vectors to the arrowing relation between single vectors. The logic of fusion identifies among the constraints those configurations where fusion degrades the information content of the set that is fused: 'info loss configurations', constraints fusing to W with *e* in the fusion. Complementarily, we identified among the vectors 'fusionally reducible sets', those whose informational content is preserved, or even (in context) enhanced, by fusion.

For the explicit record, we conclude with a concise listing of the key notions and results.

(52) Def. **Trivial Vector**. A vector **v** is 'trivial' over a constraint set Σ, iff either of two conditions holds:

a. For every hierarchy H, a linear ordering of Σ, we have H satisfies **v**,

b. For every hierarchy H, a linear ordering of Σ, H does not satisfy **v**.

In the first case, we say that **v** is *valid*; the second case, *invalid*. In the first case, **v**∈W*, the set of all vectors lacking L; in the second, **v**∈ $L^+$, the set of vectors with at least one L but no W.

(53) Def. **Arrow**. Let **a**,**b** be vectors over a set of n constraints. Vector **a** 'arrows' a vector **b**, written $\mathbf{a}\rightarrow\mathbf{b}$, iff for every coordinate *k*, $1\leq k\leq n$, we have $\mathbf{a}[k]\leq\mathbf{b}[k]$, where the order relation is determined by the scale $L<e<W$, of which '$\leq$' is the reflexive closure.

(54) Thm. **Individual Nontrivial Entailment $\Leftrightarrow$ Arrow**. For **a**,**b** nontrivial, $\mathbf{a}\rightarrow\mathbf{b}$ iff $\mathbf{a}\vDash\mathbf{b}$.
See (26).

(55) Def. **Fusion.** Let **a**,**b** be vectors over a set of *n* constraints. The 'fusion' of **a** and **b**, written $\mathbf{a}\circ\mathbf{b}$ or $f\{\mathbf{a},\mathbf{b}\}$, is the vector **c** whose coordinates **c**[*k*] are given by the following:

| | |
|---|---|
| $\mathbf{c}[k] = \mathbf{a}\circ\mathbf{b}[k] = \mathbf{b}\circ\mathbf{a}[k]$ | 'fusion is commutative' |
| $\mathbf{c}[k]$= X if $\mathbf{a}[k]$=X and $\mathbf{b}[k]$=X | 'fusion is idempotent' |
| $\mathbf{c}[k]$ = L if $\mathbf{a}[k]$=L | 'L is dominant' |
| $\mathbf{c}[k] = \mathbf{b}[k]$ if $\mathbf{a}[k]=e$ | '*e* is identity' |

By convention, $f\{\mathbf{v}\} = \mathbf{v}$.

In the following, **A** is a set of ERC vectors, **v** an ERC vector, Σ the constraint set.

(56) Thm. **Closure**: **Fusion respects entailment**. $\mathbf{A} \vDash f\mathbf{A}$. (See (34).)

(57) Thm. **Entailment by Fusion**. If $\mathbf{A}\vDash\mathbf{v}$, then for some $\mathbf{X}\subseteq\mathbf{A}$, $f\mathbf{X}\vDash\mathbf{v}$. (See (35).)

(58) Thm. **Arrow $\Rightarrow$ Entailment** If for some $\mathbf{X}\subseteq\mathbf{A}$ we have $f\mathbf{X}\rightarrow\mathbf{v}$, then $\mathbf{A}\vDash\mathbf{v}$ . (See (37).)

(59) Thm. **General Nontrivial Entailment $\Leftrightarrow$ Arrow**. For **v** nontrivial, and **A** satisfiable,
$\mathbf{A}\vDash\mathbf{v}$ iff for some $\mathbf{X}\subseteq\mathbf{A}$, $f\mathbf{X}\rightarrow\mathbf{v}$. (See (36).)

(60) Def.**W-compliant**. **A** is 'W-compliant' iff $f\mathbf{A}[k]$= W $\Rightarrow$ C[*k*]=W for all $\in\Sigma$.

(61) Thm. **Fusion / Conjunction**. For $\mathbf{a},\mathbf{b}\notin W^*$, $\mathbf{a}\circ\mathbf{b} \vDash \mathbf{a},\mathbf{b}$ iff **a** and **b** are W-compliant. More generally, if **A** contains no valid vectors, then $f\mathbf{A}\vDash$ **v** for every $\mathbf{v}\in\mathbf{A}$ iff **A** is W-compliant.
Proof. ERA:16, Props. 3.1, 3.2.

(62) Def. **Info Loss Configuration**. Let $C_k\in\Sigma$ provide the $k^{th}$ coordinate for the vectors in **A**. We say $C_k$ is an 'Info Loss Configuration' over **A** iff $f\mathbf{A}[k]$=W and there is a $\mathbf{v}\in\mathbf{A}$ such that $\mathbf{v}[k]$=e.
'A constraint column containing both W and *e* but no L. Equivalently, any coordinate in the vector set that fails to be W-compliant under fusion.'

(63) Def. **Fusionally Reducible**. $\mathbf{X}\subseteq\mathbf{A}$ is 'fusionally reducible in **A**' iff for every $\mathbf{v}\in\mathbf{X}$, $f\mathbf{A}\vDash\mathbf{v}$.
'A set of vectors whose content is encodable via fusion of a containing superset.'

## 2.4 Remark on the Logical Background

Fusion began its life in linguistics under the name 'summation' (Prince 2000). With the introduction of the 'negative', which tranforms [x~y] to [y~x] (ERA:12), a full-fledged logic emerged, identifiable as the implication-negation fragment of the relevance logic RM3 (Anderson and Belnap 1975; Meyer 1975; Parks 1972), which first appears, on its own, in Sobociński 1952. RM3 is a three-valued propositional logic, in which our W corresponds to T(rue), L to F(alse), and *e* marks the third value (which can be interpreted as 'both T and F'). In this logic, fusion is a kind of weakened analog of conjunction, arrow of material implication, and negative of standard negation. Aspects of the relation between OT and RM3 (as well as its big brother RM), are explored in ERA:47-80.

# 3. The Fusional Reduction Algorithm (FRed)

Our goal is to calculate the maximally concise, maximally informative representation of the content of any set of ERCs, its Most Informative Basis (MIB). The demand for conciseness has two distinct consequences. First, the MIB must be logically independent, in the sense that none of its vectors can be entailed by the rest. Any entailed vector is superfluous, adding nothing to the content, and can be removed — producing a smaller, equivalent set. Second, the MIB must contain no fusionally-reducible subsets of cardinality greater than one (a singleton set is uninterestingly reducible). If a multi-element subset of the MIB were fusionally reducible, we could replace it with a fusion, a single vector, thereby reducing the cardinality. These two properties cannot be unified: a fusionally reducible set can perfectly well be logically independent:

(64) **Fusionally reducible logically independent**

|  | **1** | **2** | **3** | ERC |
|---|---|---|---|---|
| **a** | W | W | L | $\mathbf{1 \gg 3 \vee 2 \gg 3}$ |
| **b** | W | L | W | $\mathbf{1 \gg 2 \vee 3 \gg 2}$ |
| **a∘b** | *W* | *L* | *L* | $\mathbf{1 \gg \{2,3\}}$ |

This is the simplest case of fusional reducibility, a W-compliant set. Here, neither element entails the other, yet their collective content is entirely present in their fusion.

From such broad considerations, we can assemble a reasonably concrete picture of what MIB elements will look like: each vector in the MIB describes the complete ranking requirements of a unique W-set.

This characterization rests on the definition of basis, of maximal informativeness, and the properties of ERC entailment. The W-set of any basis vector must be different from all the other W-sets of its fellow basis members. Two vectors with the same W-set will be W-compliant, and therefore reducible to one, contradicting the minimal cardinality requirement. Furthemore, the L-set attached to any given W-set in the MIB must be maximal, in the sense that, for any vector **v** in the MIB, there can be no other vector **x**, among all of the MIB's entailments, with the same W-set and with an L-set that properly includes that of **v**. In this case, **v** would be asymmetrically entailed by **x**, via L-retraction (22) and thus not maximally informative. The MIB, then, consists of a set of vectors with distinct W-sets, each one with the maximal L-set allowed for its proprietary W-set.

The requirement that the MIB be maximally informative plays out crucially in cases like the following, where a linear order on a three-constraint set is represented in three different ways (repeated from section 2.1 above). Each vector set imposes the same ranking requirements, and each qualifies as a *basis*, because they are each logically independent and of the same minimum cardinality.

(65) **Bases for 1≫2≫3**

| **B**.1 | **1** | **2** | **3** |
|---|---|---|---|
| **a** | W | L | L |
| **b** | e | W | L |

| **B**.2 | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | W | L | e |
| **b** | e | W | L |

| **B**.3 | **1** | **2** | **3** |
|---|---|---|---|
| **a″** | W | L | W |
| **b** | e | W | L |

The vector **b** is a fixture of all bases. There is a scale of informativeness — entailment — that linearly orders the other vectors:

(66)

| Arrow/Entailment | Crucial Coordinate |
|---|---|
| **a** → **a′** | **3**: L→e |
| **a′** → **a″** | **3**: e→ W |

The vector **a** entails both of the others, but is entailed by neither. Basis **B**.1 is therefore the MIB. But only the criterion of maximal informativeness distinguishes among the three, since all are equally concise.

The tools we have in hand to calculate the MIB are fusion and the single-vector entailment scale L→e→W. They will suffice. The MIB-producing Fusional Reduction Algorithm (FRed) works, in outline, like this:

1. Fuse everything
2. Use the fusion to identify info loss configurations, and the information lost.
3. Check for entailment of the fusion; retain if untentailed

● Recurse on all vector subsets whose information is lost in the fusion.

The first step hunts for fusionally reducible sets and less-than-maximally informative vectors. The second step finds the info loss configurations, and the vectors that participate in them. The third step determines whether the fusion (and the vectors whose content it encodes) is entailed by the other vectors; if unentailed, it is kept as part of the output of FRed. The recursion step repeats the same process on all subsets of vectors that create info loss configurations. In this way, FRed traverses the entire set, digging out and condensing all of its ranking requirements. The collection of retained fusions turns out to be exactly the MIB.

In the general case, the process applies repeatedly, fusing smaller and smaller subsets until none are left. Nowhere in the algorithm are the defining features of the MIB explicitly checked; and even the property of logical independence — a global property of bases — is examined only locally (at step 3). Consequently, we must show that the object computed by FRed, which we can call the 'Fusional Normal Form' (FNF) of the original set, is actually its MIB. This is demonstrated in Part II, along with a number of other properties of the MIB, including existence and uniqueness. Here we will focus on presenting the algorithm itself.

To see how maximal informativeness emerges, it is instructive to examine the simple **1≫2≫3** case. Suppose we start out with a basis, but not the MIB, which we will rename '**A**'.

(67) **Non-MIB for 1≫2≫3**

| **A** | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | W | L | W |
| **b** | e | W | L |

## Step 1: Fuse.

FRed's first step is to form the fusion of the whole thing. This step has the dual role of condensing fusionally reducible sets and obtaining maximal informativeness.

(68) Fusion of **A**

| **A** | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | W | L | W |
| **b** | e | W | L |
| **a′∘b** | ***W*** | ***L*** | ***L*** |

## Step 2: Identify Lost Information.

Constraint **1** is an info loss configuration, as indicated in (69) below. This configuration implicates **b** as the info loser — all information provided by **b** is indeed lost in the fusion. Let use the term 'info loss residue' for the set of vectors that contribute *e* to a given info loss configuration. Each such residue is associated with a given constraint, an info loss configuration. We can notate it as '$X_n$', where n is the index of the constraint where info is lost and 'X' is the name of the whole vector set. In the case at hand, there is only one such residure, $\mathbf{A}_1$.

(69) Info Loss Configuration in **A**, $\mathbf{A}_1 = \{\mathbf{b}\}$.

| **A** | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | **W** | L | W |
| **b** | ***e*** | W | L |
| **a′∘b** | ***W*** | *L* | *L* |

## Step 3. Check for entailment.

Is the fusion entailed by the collectivity of lost information, here just $\mathbf{A}_1$, namely {**b**}, whose fusion is identical to its sole member?

(70) Info Loss Residue $\mathbf{A}_1$

| $\mathbf{A}_1$ | **1** | **2** | **3** |
|---|---|---|---|
| **b** | e | W | L |

Entailment fails

: (e, W, L) ↛ (W,L,L) nb: **2**: W↛L

The unentailed fusion *f***A** is therefore retained as an element of the FNF of our original set A.

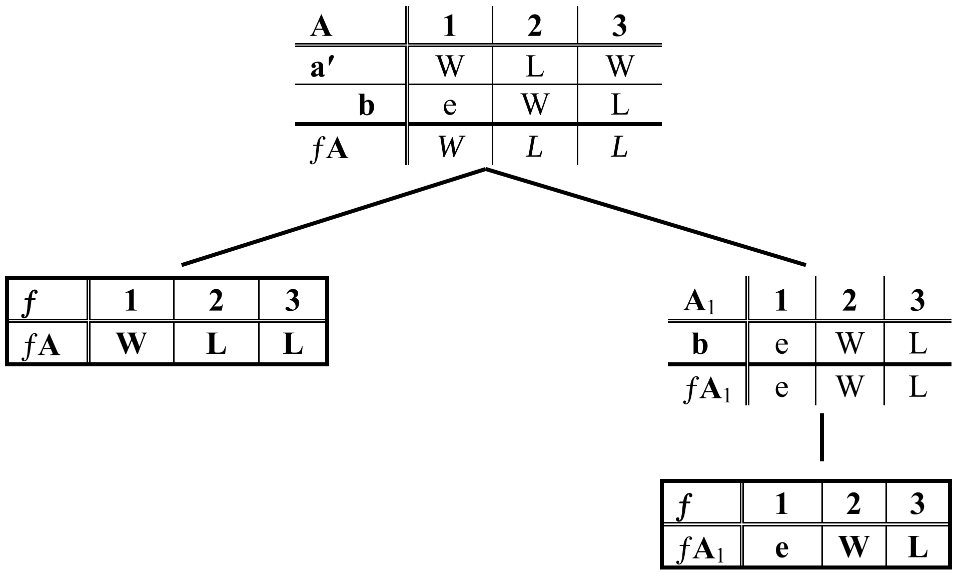
**Recursion step**. We now continue recursively with the info loss residue $\mathbf{A}_1$. In this case, the process is trivial, since $\mathbf{A}_1$ contains a single vector. The fusion of $\mathbf{A}_1$ is safely computable by inspection. There is no info loss residue, and no entailment to check. We add this fusion to the FNF, and we are done, since there is nothing left for FRed to operate on. The FNF of **A**, and therefore its MIB, which we have just calculated, is as follows:

(71) **MIB** of **A** = {**a′**, **b**}

| **MIB(A)** | **1** | **2** | **3** |
|---|---|---|---|
| **a′∘b** | W | L | L |
| **b** | e | W | L |

Application of FRed can be represented perspicuously as the construction of a tree which tracks the stages by which the algorithm generates the FNF. We begin with the original set as the root. We place the fusion, when retained, as the leftmost branch, and any info loss residues as sister nodes to it. These residue nodes may then be the site of further recursive expansion of the tree. The processing of **A** by FRed would look like this:

(72) **FRed(A) as Tree**



| **A** | **1** | **2** | **3** |
|---|---|---|---|
| **a′** | W | L | W |
| **b** | e | W | L |
| *f***A** | *W* | *L* | *L* |

| *f* | **1** | **2** | **3** |
|---|---|---|---|
| *f***A** | **W** | **L** | **L** |

| $\mathbf{A}_1$ | **1** | **2** | **3** |
|---|---|---|---|
| **b** | e | W | L |
| *f*$\mathbf{A}_1$ | e | W | L |

| *f* | **1** | **2** | **3** |
|---|---|---|---|
| *f*$\mathbf{A}_1$ | **e** | **W** | **L** |

The terminal nodes of the tree (boxed) give the MIB. For ease of checking the calculation, we also show the fusion of the whole in the node it arises from. In the interests of painstaking exactitude, we have drawn out the final step (right branch). We will supress it in future., being content to box any node that contain a single vector.

To see how the entailment check step can be crucial to the success of the algorithm, let us consider a typical case in which fusion of the whole proves uninformative:

(73) **Crucial entailment** in the course of FRed.

| **D** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **a** | W | e | e | L |
| **b** | e | W | L | e |
| **a∘b** | *W* | *W* | *L* | *L* |

| *f* | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| *f***D** | **W** | **W** | **L** | **L** |

$f(\mathbf{D}_1 \cup \mathbf{D}_2) \rightarrow f\mathbf{D}$

| **D**$_1$ | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **b** | e | W | L | e |

| **D**$_2$ | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **a** | W | e | e | L |

The info loss residues collectively entail the fusion of the whole, and it is ejected from the FNF.

How is the entailment relation to be determined? The fusion of the whole divides the original set into two disjoint parts: the fusionally-reducible subset — possibly null — that it entails and replaces, and the collection of info loss residues (one for each constraint that is an info loss configuration), which we can call 'the total residue' (TR). What matters for the MIB is that a retained fusion not be entailed by the total residue. If it were so entailed, it would be redundant with respect other fusions yet to be calculated. Retaining an entailed fusion would destroy the logical independence of the output of FRed.

The total residue can be a set of any size, given the right starting point. From the fundamental entailment/arrow relation (36), we know that entailment from the total residue amounts to arrowing from the fusion of *some subset* of the residue (under appropriate nontriviality conditions.) The appearance of the term 'some subset' might suggest the risk of a lengthy calculation. But in this situation, it turns out that we can simply fuse the *entire* residue and use that fusion to check for entailment via arrow. Valid and invalid vectors aside, the core of entailment checking runs like this, writing 'TR(**A**)' for the total residue of **A**, the union of all its info loss residues.

(74) **Entailment check** (nontrivials). If $f$**TR(A)** $\rightarrow f$**A**, omit $f$**A** from the FNF. Else, keep it.

Not included in calculation (74) are trivial vectors, valid and invalid, which arise frequently enough in the course of empirical investigation, but which do not belong in bases. FRed detects them naturally when they threaten to enter the FNF, and the algorithm can easily be outfitted to handle them with an extra clause or two. A set consisting entirely of valid vectors may show up as an info loss residue (or even, in principle, as the starting point of the process, though only the most insouciant investigator could fail to notice the uniform W* glare of harmonic bounding that renders every ranking argument vacuous.) When a *valid* set in encountered, its fusion should merely be excluded: it is entailed by anything, including nothing.

An unsatisfiable set is a more alarming discovery, since it indicates failure of the constraint set to analyze the data. When FRed encounters an unsatisfiable set, it should issue an announcement to that effect. An unsatisfiable set always contains a subset fusing to $\mathsf{L}^+$ , as noted in (41). Any such subset will be isolated and fused in the course of FRed, revealing its identity.

We these considerations in mind, the entailment check step can be given as follows:

(75) **Entailment Check** on a set **A**.

a. Trivial-True. If $f$**A**∈ W*, then omit $f$**A**.

b. Trivial-False. If $f$**A**∈ $L^+$, EXIT and announce: '**A** is unsatisfiable.'

c. Else: If $f$**TR** → $f$**A**, then omit $f$**A** from the FNF. If not, retain it.

A complete statement of FRed runs as follows:

(76) **FRed**

```
0. Base Step..                                    # Terminate when A is empty.
     If A=Ø then FNF(A):=Ø
1. Fuse All.                                      # Collect the fusion of the whole.
     HoldFus:= ƒA                      .
2. Identify Lost Information.                     #  Collect the  Info Loss Residues
     ILC(A):= {C_i | C_i∈ Σ & ƒA[i] = W & ∃v∈A such that v[i]=e}
     Res(A,i):= {v| v∈A  & ∃C_i∈ILC(A) such that v[i]=e}
     TR(A):= ∪i Res(A,i)
3. Check Entailment.                              # Omit trivial and entailed fusions
     If ƒA∈W* then HoldFus:=Ø
     If ƒA∈L^+, EXIT and announce: 'A is unsatisfiable.'
     If ƒTR(A) → ƒA then HoldFus:=Ø
4. Recurse.                                       # Do FRed on each info loss residue
     FNF(A):= HoldFus ∪  ∪_k FNF(Res(A,k))
```

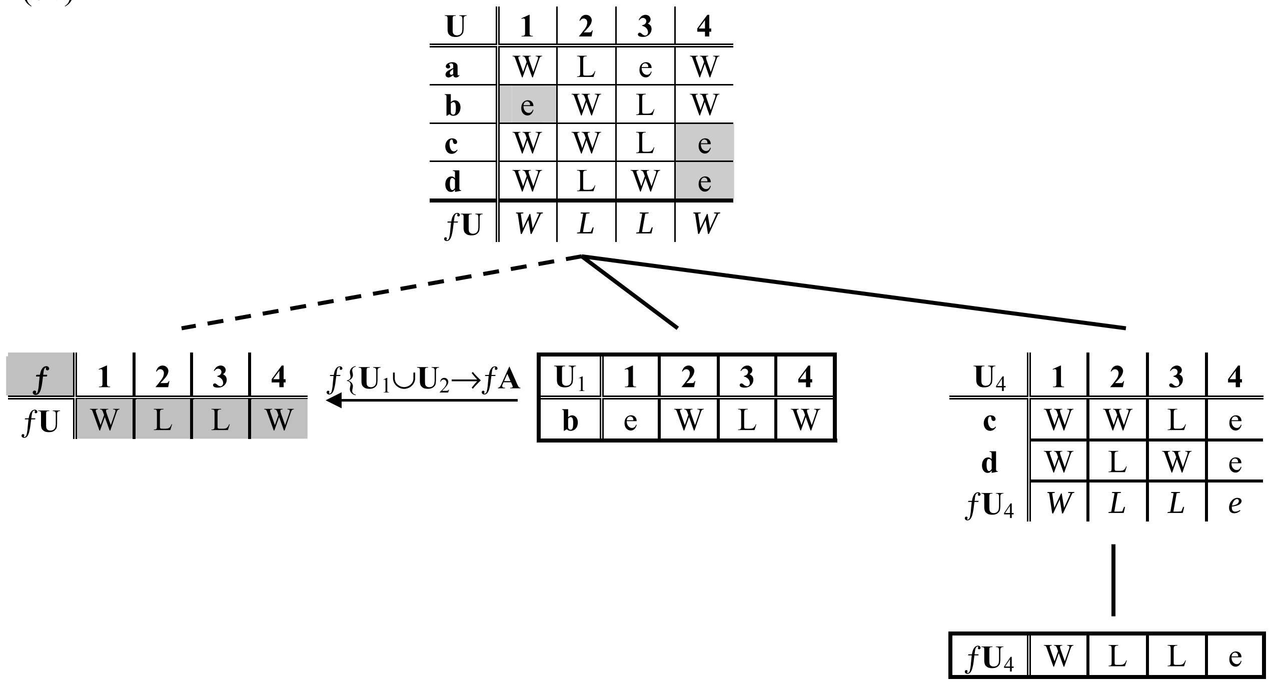
Let's conclude the illustration of FRed with a couple of examples where recursion does more than report the obvious. Consider the following set of ERC vectors:

(77) **A set U of vectors**

| **U** | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **a** | W | L | e | W |
| **b** | e | W | L | W |
| **c** | W | W | L | e |
| **d** | W | L | W | e |

FRed's run is pictured in the following tree:

(78)



| U | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **a** | W | L | e | W |
| **b** | e | W | L | W |
| **c** | W | W | L | e |
| **d** | W | L | W | e |
| *f***U** | *W* | *L* | *L* | *W* |

| *f* | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| *f***U** | W | L | L | W |

*f*{$\mathbf{U}_1 \cup \mathbf{U}_2 \rightarrow f\mathbf{A}$}

| $\mathbf{U}_1$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **b** | e | W | L | W |

| $\mathbf{U}_4$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **c** | W | W | L | e |
| **d** | W | L | W | e |
| *f*$\mathbf{U}_4$ | *W* | *L* | *L* | *e* |

| | | | | |
|---|---|---|---|---|
| *f*$\mathbf{U}_4$ | W | L | L | e |

Of note in the first round of daughterly processing is the discovery that the whole fusion is redundant. This encapsulates the fact that the fusionally reducible subset {**a**} is entailed by its complement, the total info loss residue. Although no single vector in the residue is an entailer, FRed finds the collective entailment without difficulty. In the second, recursive round of processing, FRed ascertains that the W-compliant subset **U**$_4$ can be completely reduced to its fusion. There are no info loss residues, and the procedure terminates, having discovered that the original set of four vectors describes a ranking system that requires only two ERCs.

As a final example, let's look closely at a case where FRed has to work recursively and nontrivially on residues — Lombardi's Swedish analysis.

(79) **FRed(S), Step 1**: **Fuse All.**

| S | 1:AGREE | 2:IDLAR | 3:*LAR | 4:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | ***e*** | | W | L |
| **b**. skuːg → skuːg ~ skuːk | ***e*** | W | L | |
| **c**. vigsəl → viksəl ~ vigsəl | W | L | W | |
| **d**. vigsəl → viksəl ~ vigzəl | ***e*** | | W | W |
| **f**. ægde → ægde ~ ækte | ***e*** | W | L | W |
| **g**. stekde → stekte ~ stekde | W | L | W | L |
| **h**. ægde → ækte ~ ækte | W | | L | |
| *f***S** | ***W*** | *L* | *L* | *L* |

For emphasis, we have omitted most of the e's in the tableau. The crucial cells inducing info loss are highlighted by shading. Only constraint **1** has an info loss residue, which we collect:

(80) **FRed(S), step 2. Identify the Info Loss Residue of S: $\mathbf{S}_1$ = {a,b,d,f}**

| **$\mathbf{S_1}$** | **1**:AGREE | **2**:IDLAR | **3**:*LAR | **4**:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | W | L |
| **b**. skuːg → skuːg ~ skuːk | | W | L | |
| **d**. vigsəl → viksəl ~ vigzəl | | | W | W |
| **f**. ægde → ægde ~ ækte | | W | L | W |

For step 3, we fuse the total residue — here, just **S**$_1$ — to enable entailment checking.

$$f\text{TR}(\mathbf{S}) = f\mathbf{S}_1 = (\text{e,W,L,L})$$

It is evident that $f$**S**, the fusion of the original set, namely (W,L,L,L), is unentailed.

(81) **FRed(S), step 3**. Entailment Check. $f$TR(**S**) $\nrightarrow$ $f$**S**. No action taken.

We execute the recursion step:

(82) **Recurse**. FNF(**S**) := {$f$**S**} ∪ FNF(**S**$_1$)

We must now start all over again with **S**$_1$. As always, we begin with fusion:

(83) **FRed(S$_1$). Step 1: Fuse All.**

| **$\mathbf{S_1}$** | **1**:AGREE | **2**:IDLAR | **3**:*LAR | **4**:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | ***e*** | W | L |
| **b**. skuːg → skuːg ~ skuːk | | W | L | |
| **d**. vigsəl → viksəl ~ vigzəl | | ***e*** | W | W |
| **f**. ægde → ægde ~ ækte | | W | L | W |
| $f$**S$_1$** | | ***W*** | *L* | *L* |

Only constraint **2** shows info loss. This gives rise to the info loss residue of **S**$_1$, namely **S**$_{12}$.

(84) **FRed(S$_1$). Step 2**: **Collect Info Loss Residue** S$_1$: S$_{12}$

| **$\mathbf{S_{12}}$** | **1**:AGREE | **2**:IDLAR | **3**:*LAR | **4**:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | W | L |
| **d**. vigsəl → viksəl ~ vigzəl | | | W | W |

To perform the entailment check, we fuse the total residue of **S$_1$**, which is mercifully just **S$_{12}$**.

$$f\text{TR}(\mathbf{S}_1) = f\mathbf{S}_{12} = (\text{e,e,W,L}) \nrightarrow (\text{e,W,L,L}) = f\mathbf{S}_1$$

(85) **FRed(S$_1$). Step 3.** Entailment Check. $f$TR(**S**$_1$) $\nrightarrow$ $f$**S**$_1$. No action taken

We advance to the recursion step:

(86) **FRed(S1), Recurse**. FNF(**S$_1$**) := {*f***S**$_1$} ∪ FNF(**S**$_{12}$).

Recursive application of FRed to **S**$_{12}$ begins with its fusion, which is clearly unentailed, and furthermore indicates that the residue is at last null.

(87) **FRed(S$_{12}$). Fuse All.**

| **S**$_{12}$ | **1**:AGREE | **2**:IDLAR | **3**:*LAR | **4**:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | W | L |
| **d**. vigsəl → viksəl ~ vigzəl | | | W | W |
| *f***S**$_{12}$ | | | *W* | *L* |

The final step is taken:

(88) FRed(**S**$_{12}$). Recursion Step. FNF(**S**$_{12}$) = {*f***S**$_{12}$} ∪ FNF(Ø)

The whole process will ends abruptly with the 0th round of the next recursion, which informs us that the FNF of the null set is itself. Putting all the pieces together, we are licensed to conclude that for S, the FNF (and therefore the MIB) consists of the fusions we have recursively calculated:

(89) FNF(**S**) = {*f***S**, *f***S**$_1$, *f***S**$_{12}$}

**Discussion.** Of note in the derivation is the deft elimination of redundant elements. On the first round, three vectors are compressed into one. Here they are:

(90) **Round 1. Fusionally reducible subset**

| | | | | |
|---|---|---|---|---|
| **c**. vigsəl → viksəl ~ vigsəl | W | L | W | |
| **g**. stekde → stekte ~ stekde | W | L | W | L |
| **h**. ægde → ægde ~ ækte | W | | L | |

This is a W-compliant set, and therefore exactly equivalent to its fusion.

On the second round, another two vectors are compressed:

(91) **Round 2**. Fusionally reducible subset.

| | | | | |
|---|---|---|---|---|
| **b**. skuːg → skuːg ~ skuːk | | W | L | |
| **f**. ægde → ægde ~ ækte | | W | L | W |

These are not W-compliant internally, but they are fusionally reducible in their context (**S**$_1$). Notice also the entailment relation between **b** and **f**. FRed eliminates a redundancy involving entailment, even though it is not specifically looking for it here.

The third round compresses two more vectors, once again W-compliant:

(92) **Round 3**. Fusionally reducible subset.

| $S_{12}$ | **1**:AGREE | **2**:IDLAR | **3**:*LAR | **4**:IDONSLAR |
|---|---|---|---|---|
| **a**. stekde → stekte ~ stegde | | | W | L |
| **d**. vigsəl → viksəl ~ vigzəl | | | W | W |

The trivial, valid vector **d** shows up in this residue and is forthwith reduced, even though the algorithm never takes explicit note of it.

There is a final property in this example worthy of note, which is utterly characteric of FRed: the *disclosure* of info loss residues as recursion progresses. When we advance to the info loss residue of the first step, the fusionally reducible subset (90) is gone, taking its L coordinates with it. Crucially, all L values now disappear from Constraint **2**, rendering it an info loss configuration, due to the *e* coordinates **a**[2] and **d**[2]. Such new, disclosed residues provide the fodder for further recursion.

We conclude with a couple of remarks on features of FRed.

**Entailment Check**. Entailment check asks whether the total residue (TR) — the union of all info loss residues — entails the fusion of the whole. In the general case, we are are only guaranteed that any entailment from a set follows from the fusion of some subset. In FRed, it suffices to fuse the entire total residue and compare that with the fusion of the whole; no prospecting around for subsets of TR is needed.[9] This special situation arises because the vector in the total residue are included the whole, so that the following result applies:

(93) **Fusion entailed by Fusion**. Let **A** be an ERC vector set, such that $f\mathbf{A}\notin \mathsf{W}^*$. Let $\mathbf{X}\subseteq\mathbf{A}$ be such that $f\mathbf{X}\notin \mathsf{L}^+$. Then, $\mathbf{X}\vDash f\mathbf{A}$ iff $f\mathbf{X}\rightarrow f\mathbf{A}$.

Proof. See Part II.

A further point of interest: conducting the check very locally suffices to determine whether entailment of the whole fusion occurs *anywhere*. This is perhaps not too surprising at the very first step of the algorithm, where the entire set under consideration is split into two parts. But after that, in the full-blown general case, there can be many, many info loss residues, each sprouting many further residues, generating a massively branching derivation tree, in which entailment could, one is tempted to imagine, lurk in some far-distant node or, worse, be spread in fragments throughout a number of them. But it turns out that any entailing set must always end up entirely represented in the total info loss residue that is sister to the fusion, and no global traverse is necessary to uncover entailment. A qualitative account of this property is given in Section 4.2 below. Part II contains the details.

[9] Because the fusion of the whole $f\mathbf{A}$ has the maximal number of L coordinates of any fused subset of **A**, this entailment arises only when $f$TR exactly matches $f\mathbf{A}$ in every L coordinate, and is less than or equal to it in disjunctivity, having a subset of its W coordinates. To check entailment, we need only compare the absolute conjunctivity (number of L coordinates) of the two fusions, which must be exactly the same to give entailment, and the absolute disjunctivity (number of W coordinates), where to have entailment the disjunctivity of $f\mathbf{A}$ must equal or be greater than that of $f$TR. Thus, there is no need for coordinate by coordinate comparison; simple numerical comparison will do the job. But see also Section 4.1 below for a further use of coordinate checking.

**Residues**. FRed can handle all kinds of relations between info loss residues. Residues can overlap, stand in subset-superset relations, or even be identical. Here's an example which shows the variety of configurations available:

(94) **Relations between Info Loss Configurations**

| **A** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **a** | W | W | **e** | L | |
| **b** | W | **e** | **e** | | |
| **c** | **e** | **e** | W | L | L |
| *f***A** | ***W*** | ***W*** | ***W*** | *L* | *L* |

| Residues | Relations |
|---|---|
| $\mathbf{A}_1 = \{c\}$ | $\mathbf{A}_1 \subseteq \mathbf{A}_2$ |
| $\mathbf{A}_2 = \{b,c\}$ | $\mathbf{A}_2 \cap \mathbf{A}_3 = \{b\}$ |
| $\mathbf{A}_3 = \{a,b\}$ | $\mathbf{A}_1 \cap \mathbf{A}_3 = \emptyset$ |

Because we must pursue every info loss configuration, we are necessarily going to encounter repetition in the branches of the process. For example, a little calculation shows these residue duplications implicit in (94):

$$\mathbf{A}_{21} = \{\mathbf{c}\} = \mathbf{A}_1$$
$$\mathbf{A}_{23} = \{\mathbf{b}\} = \mathbf{A}_{32}$$

FRed sails through all such relations smoothly, repeating itself exactly over identical residues, no matter how they are derived. Since the FNF is the *set* of collected fusions, any duplications will have no  no effect on the outcome.

# 4. FRed in Context

We consider FRed in the context of bases other than the MIB (4.1), its relation to RCD (4.2), its computational cost (4.3), and its relation to other algorithms of similar purpose or structure (4.4).

## 4.1 The Skeletal Basis

Each member of the MIB describes the entirety of ranking relations of a unique W set. Consider the MIB for $\mathbf{1} \gg \mathbf{2} \gg \mathbf{3} \gg \mathbf{4}$.

(95) **MIB ($\mathbf{1} \gg \mathbf{2} \gg \mathbf{3} \gg \mathbf{4}$)**

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **a** | W | L | L | L |
| **b** | | W | L | L |
| **c** | | | W | L |

From this, we can directly read off everything that constraint **1** must dominate, and so on, for all the others. But such maximal local informativeness also carries a cost: from **a** alone we cannot tell which of Constraint **1**'s necessary subordinates it must *immediately* dominate. A MIB vector contains all the information derivable from transitivity, lumping the dominated constraints together in one class.

The MIB is one among the potentially many different bases. Among them is one that contains *no* information derivable from transitivity. Because it represents the domination pattern so sparsely, let us call it the 'skeletal basis'. For $\mathbf{1} \gg \mathbf{2} \gg \mathbf{3} \gg \mathbf{4}$, it will look like this:

(96) **Skeletal Basis for $\mathbf{1} \gg \mathbf{2} \gg \mathbf{3} \gg \mathbf{4}$**

| | **1** | **2** | **3** | **4** |
|---|---|---|---|---|
| **a** | W | L | | |
| **b** | | W | L | |
| **c** | | | W | L |

Only the immediate domination requirements on the W-set of each component vector are represented. The characterizing quality of the Skeletal Basis that it has as many as *e*'s as possible.

We can ensure this through various definitional strategies. A relatively straightforward one is to first define a 'fusional order' on the comparative values, in which L $<_f$ W $<_f$ e. (We write '$x \geq_f y$' for '$x >_f y$ or x=y.') It earns its name because fusion of values takes the minimum value in this order (see Meyer 1975:400, ERA:55-6, and Part II.) Extend the order on values to an order on vectors in the usual coordinatewise fashion, parallel to the treatment of arrow (53). Each vector in the Skeletal Basis must then be maximal in the fusional order in the sense that no other vector entailed by that basis can be greater than it.

(97) Def. **Skeletal Basis**. **B** is the skeletal basis for **A** iff (1) **B** is a basis for **A**, and (2) for every **x** such that $\mathbf{B} \vDash \mathbf{x}$ and for every $\mathbf{b} \in$ B, if $\mathbf{x} \geq_f \mathbf{b}$, then **x**=**b**.

Unlike with the MIB, there is no operation analogous to fusion that will combine pieces of data to produce the Skeletal Basis. But FRed contains the relevant information to allow a direct computation, based on the relationship between *ƒ*TR and the fusion of the whole. To see this, consider the possible patterns of values that can occur in individual coordinates, which are highly restricted because the vectors in TR are all included the whole set.

(98) **Corresponding coordinate values** in whole and residues fusions

| | | | |
|---|---|---|---|
| *ƒ***A**[*k*] | W | e | L |
| *ƒ*TR[*k*] | e/W | e | e/W/L |

If the whole of **A** fuses to W at coordinate *k*, then no member of the TR can be L at *k*. If the whole fuses to *e* at *k*, then so must TR. From the entailment point of view, there is nothing to check at these *e* and W coordinates of *ƒ***A**. They are guaranteed to provide no problems, since e→W and X→X. Only the last requires scrutiny. If *ƒ*A[k] is L, then any coordinate in *ƒ*TR that is *not* L will disrupt entailment.

We may therefore check entailment by this procedure: go through every L coordinate of *ƒ*TR and replace the corresponding coordinate in *ƒ*A with *e*. If the end result is a vector with *no* L's, a member of W*, a valid vector, then we have *ƒ*TR→*ƒ***A**.

If not, then the result of this calculation is a member of the Skeletal Basis. We have taken away every L from $f$**A** that can be recovered from the data in the TR. What's left must be due to the subordination pattern demanded by the fusionally reducible set that $f$**A** replaces, and indeed only as much of that as is not attributable to anything in the TR.

To incorporate these remarks, we rewrite the entailment check step of FRed (76) as follows:

(99) **Entailment Check and the Skeletal Basis**

```
If $f\mathbf{A} \in L^{+}$, EXIT and announce: 'A is unsatisfiable.'    # detect inconsistency
$\mathbf{s} := f\mathbf{A}$                                             # potential Skel't'l Basis element
For every coordinate $k$, if $f$TR[$k$] = L then $\mathbf{s}[k] := e$   # eliminate TR L's from s
If $\mathbf{s} \in W^{*}$ then HoldFus:=Ø                              # omit entailed fusion from MIB
    else SKELB = SKELB ∪ {$\mathbf{s}$}                                  # save Skeletal Basis element
```

Here we construct a potential member **s** of the Skeletal basis from a corresponding potential member of the MIB. If **s** is rendered valid by the L replacement step, then we forget it and erase the fusion of the whole from the location where it is (temporarily) stored. If it is not valid, then we retain the fusion of the whole in HoldFus and add **s** to the Skeletal Basis.

Just as the Skeletal Basis is characterized by a maximum number of *e*'s, and the MIB by a maximum number of L's, so there is another extremal basis of interest: the Least Informative Basis, with a maximum number of W's. It can be defined in the same way as the MIB, reversing the informativeness demand on its elements. Although of no obvious utility in itself, it may be interestingly compared with the other two extremal bases, to determine what is shared across them all. We find that coordinates fall into two basic categories: those which vary in corresponding vectors, determining relative informativeness and skeletality; and those whose values are fixed in every basis, giving the fixed basis core of ranking requirements that each basis vector asserts with greater or less force.

## 4.2 FRed and RCD

The Ranking Problem can be addressed from different angles, with different goals. Recursive Constraint Demotion (RCD) efficiently delivers a ranking, if one exists, sufficient to satisfy all the ERCs on hand, and also discovers when there is no such ranking. With less efficiency but greater detail, FRed presents in maximally concise and maximally informative way the entirety of ranking conditions inherent in a ERC set. It is instructive in both directions to compare the two.

RCD is usually cast as a ranking algorithm, producing a concrete result — a stratified hierarchy, from which a linearized ranking may be chosen. FRed, in apparent contrast, collects information *about* rankings, presenting it not as a hierarchy of any sort but, more abstractly, as a set of ERCs. Much of the apparent conceptual distance between the two algorithms is a matter of analytic perspective that can be eliminated by re-considering RCD as an information producer.

The first action in RCD, as in FRed, is to compute the fusion of the whole (cf. ERA:21-26 on RCD). Hierarchywise, this tells us which constraints are immediately rankable: those that fuse to *e* or W; complementarily, we may think of this as identifying those constraints that must be demoted, those that fuse to L. Informationally, the fusion of the whole gives us an ERC which summarizes a set of ranking conditions. To normalize RCD to the FRed model, let us put aside the hierarchizing interpretation and proceed informationally: we want to *retain* the fusion of the whole as the output of the first step of RCD, postponing any consideration of ranking per se. In this new conception, the fusion is not a mere tool to determine rankability, but the principal goal of the calculation, as it is in FRed.

RCD in the most familiar formulations advances to its recursive step by a kind of double elimination. Those ERCs satisfied by the newly stratified constraints are omitted from further consideration; similarly omitted are those *constraints* that have been ranked. We are then left with a smaller set of ERCs *and* a smaller set of constraints to deal with. FRed, in constrast, works by eliminating only the satisfied ERCs and always keeps the entire set of constraints intact. But this constrast is illusory. The double-elimination conception is rooted in the 'mark-data pair' data structure, which is a list of the constraints crucially violated by the desired optimum and its competitor; that is: the W and L sets of the comparison. When we move to the ERC vector representation, with its explicit identity (*e*), the constraints fall into three sets, not two, with inertness formally recognized as presence (of *e*) not absence from the accounting. We need no longer remove the stratified constraints, because they award *e* to the retained ERC vectors and are therefore inert. To see this, consider the following simple example:

(100) **RCD by single elimination**

| **I** | **1** | **2** | **3** |
|---|---|---|---|
| **a** | **W** | L | e |
| **b** | e | W | L |

→

| **II** | **1** | **2** | **3** |
|---|---|---|---|
| | | | |
| **b** | e | **W** | L |

→

| **III** | **1** | **2** | **3** |
|---|---|---|---|
| | | | |
| | | | |

In Stage I we find that constraint **1** is rankable. This eliminates comparison **a**, leaving **b** to be reckoned with at stage II. At that point, the ranking of the constraint **2** satisfies **b**, leading to stage III, a kind of Optimality Theoretic nirvana. Free of ERCs, we step off the wheel of recursion. In the process, there is no need to literally remove *constraints* from consideration; at the second stage, constraint **1** would have been classically removed, but here we simply keep it, as its presence does no harm. We may now conceive of RCD as compiling a set of vectors via fusion — the 'Demotion Normal Form' of the original set — paralleling FRed. In example (100), the vectors will be these:

(101) **Demotion Normal Form**

| **DNF** | **1** | **2** | **3** |
|---|---|---|---|
| **a∘b** | W | L | L |
| **b** | e | W | L |

The DNF is exactly the same as the FNF in this case. In other cases, of course, they will diverge; for example, a set consisting of (W,e,L) and (e,W,L) has RCD Normal Form (W,W,L), but its Fusional Normal Form, and MIB, is identical to it.

These difference between FRed and RCD arise because of the differing ways that they move to recursion. To clarify these differences, we must first surmount an obstacle of perspective. In familiar RCD, we talk of *removing* ERCs (those satisfied by some rankable constraint assessing W) and in FRed we talk of *retaining* ERCs (those involved in info loss configurations), but these are complementary descriptions of the same principle of selection. In example (100), the constraint **1** is an info loss configuration. RCD proceeds by examining the info loss residue of this constraint — vector **b** — exactly as FRed does.

RCD and FRed both continue recursively with info loss residues. But where FRed explores each one separately, RCD makes a grosser calculation. RCD retains precisely those ERCs which earn *e* in *every* rankable constraint; that is, in every info loss configuration. RCD, then, continues with the *intersection* of all info loss residues. This is equivalent to removing those ERCs satisfied by some W and continuing with what's left. The following example illustrates this point:

(102) **RCD intersects info loss residues:** {d,f} = $A_1 \cap A_2 \cap A_3 \cap A_4$

| **A** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| **a** | W | e | W | W | L | e | e |
| **b** | e | W | e | e | e | L | e |
| **c** | e | e | W | e | e | L | e |
| **d** | e | e | e | e | e | W | L |
| **f** | e | e | e | e | W | e | L |
| *ƒ***A** | ***W*** | ***W*** | ***W*** | ***W*** | **L** | **L** | **L** |

The info loss residues, $A_1$ through $A_4$, are shaded: their intersection is boxed in heavy lines. RCD removes from **A** the vectors {**a,b,c**} because they are satisfied by the rankable constraints (those fusing to W); equivalently, RCD continues with the intersection of the residues, namely {**d**, **f**}.

We can now formulate RCD in the manner of FRed, as a collector of fusions. In outline, it looks like this:

(103) **RCD as Information Gatherer**

[1] Fuse all ERC vectors.
[2] If the fusion lies in $L^+$, EXIT and announce failure;
 Else, save the fusion in DNF.
[3] Identify the info loss residues.
[4] Recurse on the *intersection* of the info loss residues.

Example (102) yields the following Demotion Normal Form:

(104) **DNF(A)**

| **DNF(A)** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| *ƒ***A** | W | W | W | W | L | L | L |
| $f\cap$**A**$_i$ | e | e | e | e | W | W | L |

The logic of fusion allows to bridge between this representation and the stratifying conception, if (following ERA:22ff., 83ff.) we extend the fusion operation to constraints, applying it to columns in the tableau as well as to rows. Taking a constraint to be a column vector over {W,L,*e*}, we can use exactly the same rules for fusion that we have used throughout. A 'stratum' is then understood to be the *constraintwise* fusion of a collection of constraints — namely, those that are 'rankable', those giving rise at some stage of RCD to *e* or W in the fusion of the whole remaining ERC set.

On this view, the seven-constraint example we've been following turns out like this:

(105) **RCD with constraint fusion**

| **A** | **1∘2∘3∘4** | **5∘6** | **7** |
|---|---|---|---|
| *f***A** | W | L | L |
| *f*∩**A**$_i$ | e | W | L |

RCD, then, emerges as FRed with one essential modification: after the fusion of the whole ERC vector set, we interpolate a step where RCD contracts the constraint system by fusing together all rankable constraints. The FRed algorithm then proceeds with the reduced system. In this representation, there is at most one info loss configuration — the fused mega-constraint; everything elses fuses to L. Its info loss residue is precisely the intersection of the info loss residues of its components, because of the standard fusional rule W∘e=e∘W=W. If we add one final clean-up operation that fuses all constraints in the lowest stratum, the characteristic result would take on the following form, behind which we imagine a system of 11 constraints, and a set **U** containing various ERCs whose number we need not fix.

(106) **Typical Result of RCD**

| **DNF(U)** | **1∘7∘11** | **4∘6** | **2∘5∘9** | **3∘8∘10** |
|---|---|---|---|---|
| *f***U** | W | L | L | L |
| *f***U**′ | e | W | L | L |
| *f***U**″ | e | e | W | L |

For RCD, the entailment check step of FRed will be vacuous in satisfiable ERC sets, because there can be no entailment of the fusion of the whole from the residual vectors: the fusion will contain an L at all non-*e* coordinates, and each residual vector must contain a W somewhere.

Under this conception, RCD produces the MIB for the fusionally-contracted constraint system, which in the example consists of four derived constraints: {1∘7∘11, 4∘6, 2∘5∘9, 3∘8∘10}. When the ERCs require a linear order on the constraints, each stratum contains exactly one constraint. Constraint fusion is trivial, and RCD and FRed agree completely. Whenever fusional contraction involves distinct constraints, RCD and FRed diverge. RCD loses ranking information unrecoverably via nontrivial constraint fusion, in the usual way, by fusing W and *e*, while FRed losslessly pursues all strands of information.

Developing this approach in detail requires that the contraction operation be specified. More subtly, because contraction can change the composition of the constraint set at any recursive stage, the character of all preserved fusions will also change. After the first stage of applying RCD to some set A, we first have *f*A. When the constraint set is contracted, on the basis of L-distribution in *f*A, we will also want to be sure that the saved version of *f*A is contracted as well, so that it accords with the developing picture of the constraint set's composition. We must similarly revise the saved fusions at every stage.

The core of contractional RCD will proceed according to the following outline, which suppresses the inconsistency check and the termination condition (emptiness of the ERC set).

(107) **Contractional RCD**

[1] Fuse all ERC vectors and save the result in DNF.
[2a] Contract the constraint set, fusing all constraints projecting W or e in [1].
[2b] Contract the members of DNF according to the same scheme.
[3] Recurse on the single info loss residue.

We defer exploration of technical details until Part II. The result is a reduction of the original ERC set to a canonical form, a kind of triangular matrix, echoing the reduction of the ranking requirements to a single linear order on the fusionally-contracted constraint set.

Just as it is profitable to view RCD as a producer of information, so is it worthwhile to examine FRed as a purveyor of rankings. Consider any branch in the tree of FRed recursions, running from the root vector set to a single terminal vector. The branch begins with the entire set under consideration, call it **A**, and proceeds to some info loss residue of **A**. Since the numbering is arbitrary, let us imagine (without loss of generalization) that it is perfectly suited to our purposes, so that the residues along the branch we're discussing are $A_1$, $A_{12}$, $A_{123}$, …,$A_{123\ldots k}$.

The residue $A_1$ is comprised of those ERCs to which $C_1$ assigns *e*. Complementarily put, $A_1$ is precisely what's left of A when all the ERCs earning W in $C_1$ are removed from A. The effect of removal is precisely what is obtained by filtration when $C_1$ is ranked at the top of the hierarchy: the residue $A_1$ is made up of those ERCs that are not yet satisfied by this ranking maneuver; $A_1$ consists of the data about which $C_1$ is silent. RCD continues with those ERCs that are as-yet unsatisfied when the *entire* set of rankable constraints is placed at the top as a kind of mega-constraint. FRed separately considers what happens when each rankable constraint occupies the topmost position.

Pursuing this logic, we find that the branch terminating in $A_{123\ldots k}$ corresponds to a linearly-ordered ranking of constraints $C_1 \gg C_2 \gg C_3 \gg \ldots \gg C_k$. The terminal node is an ERC containing all the information about the possible successful continuations of the ranking into the rest of the constraint set. At each stage, the saved fusions record every possible ranking arrangement that has a chance to lead to a successful hierarchy, mentioning both the available dominators (W) and the constraints that must be dominated (L).

Starting back at the root **A**, we find that the fusion $f$**A** enumerates all possible top-ranked constraints: these provide the non-L-coordinates of $f$**A**. Ignoring the constraints that yield *e* in $f$**A** — they do no work, where the job is to outrank all L's with a W — FRed next considers what would happen if each W-supplying constraint were indeed first in a ranking order. The info loss residue of each such constraint contains all the data that is not resolved by ranking it at the top. Moving to the next stage, the fusion of any first-order residue $A_j$ identifies the constraints that may be productively ranked just below Cj, as second in the unfolding linear rankings. And so on, through the residues of residues, until none are left.

FRed does not remember the rankings its unfolds, retaining instead the fusions it creates along the way, an exact record of the conditions that it is exploring. (The entailed, redundant fusions are eliminated in the interest of conciseness.) Each fusion says: these are the possible continuations. Each residue $A_{k\ldots m}$ says: if you have continued in *this* way, adjoining $C_m$ to the end of the initial sequence $C_k \gg \ldots \gg C_{m-1}$, then here is the data unsolved by that partial ranking, the further conditions that must be met. The fusion $f A_{k\ldots m}$ is the key to what constraints may be ranked next, and identifies their next-order residues. Viewed as ranking algorithms, both RCD and FRed operate locally with respect to developing hierarchies, seeking out the set of the contraints that can be ranked *next*. RCD amalgamates them; FRed explores, for each possibility, all the further possibilities that lead to eventual success, if success is available. It is remarkable that if any of the branches so enumerated lead to success, then all of them do; and if any lead to failure, then all fail. The fact that such localism is viable — that, for example, at the very first step you can determine which constraints can stand in topmost position — is a reflection of the structure of lexicographic optimization that OT is based on.

Note that FRed does not simply enumerate all possible rankings: it is guided by the ERC set. If the collective sense of the ERCs is that a single linear ordering is required, FRed looks at only one residue in each step. (RCD, with its aggressive fusional contraction of the constraint set, reduces the ranking problem to linear ordering; hence its efficiency.) Things can be considerably more complicated — in the worst case, looking at $k-1$ residues of each $k$th residue, a condition which we will examine in the next section. But the complexity is inherent in the structure of the ranking conditions provided by the data.

We note finally that the ranking perspective on FRed is of direct use in understanding its properties. A valuable relation between ranking and ERC entailment is established in ERA:29, Prop. 4.3). Given a hierarchy, let the 'rank' of an ERC be determined by the highest-ranking constraint that assigns it a polar value. (In a successful hierarchy, this value will be W.) For non-trivial ERCs, the rank relation in a successful hierarchy *refines* the arrow/entailment relation, in the sense that, if $\mathbf{a}\rightarrow\mathbf{b}$ for two distinct ERC vectors, then the rank of **b** is not lower than than the rank of **a**. Turning this contrapositively around: if **b** is of lower rank than **a**, then we cannot have $\mathbf{a}\rightarrow\mathbf{b}$. From this, it then follows that if $f\mathrm{A}_\mathrm{K}$ is entailed, for some sequence K, it will be entailed by the constraints in the total residue of $\mathrm{A}_\mathrm{K}$, since these include all those of lower rank than $f\mathrm{A}_\mathrm{K}$ in the ranking sequences developed through $\mathrm{A}_\mathrm{K}$ and its residues. This means that the extreme locality of the entailment check step of FRed — flying in the face of the potentially daunting bushiness of the FRed tree, which has far-flung nodes of potential relevance — is fully justified. We take the matter up in more detail in Part II. For present purposes, it shows once again how the informational interpretations of FRed and RCD are mirrored in direct ranking interpretations, leading to considerable mutual illumination.

## 4.3 The worst-case complexity of FRed

As a measure of the complexity of FRed, let us count the total number of ERC sets it involves, summing over the entire computation.

To obtain the worst possible case, imagine that each of $n$ constraints produces its own info loss residue. Still looking for the worst, assume that each such first-order residue itself produces a maximum number of its own residues, and so on. Since each of these $n$ first-order residues has one constraint that is all $e$'s, each will maximally produce $(n-1)$ further residues. We therefore have $n(n-1)$ second-order residues. Continuing in this fashion, each of the $n!/(n-k)!$ $k$th-order residues produces $n-k$ further residues, adding $n(n-1)\dots(n-k)$ more residues to the total, until we reach the last possible level of residues, the $(n-1)$st. (Each residue of degree $k$ neuters out $k$ constraints, and we want one constraint left at the end, so as to obtain the absolutely maximal extension of the process.) Summing the terms, we find that the total number of ERC sets computed, including the initial one, is given by this formula:

(108) $$\text{Total} = 1 + n + n(n-1) + n(n-1)(n-2) + \dots + n!$$

To estimate this quantity, we observe first that it may be re-written as follows:

(109)
$$\begin{aligned}\text{Total} &= \frac{n!}{n!} + \frac{n!}{(n-1)!} + \frac{n!}{(n-2)!} + \ldots + \frac{n!}{1!} \\ &= n!\sum_{k=1}^{n}\frac{1}{k!}\end{aligned}$$

The summation term in this expression is recognizable as a truncation of the series expansion for *e*–1, where *e* is the base of the natural logarithms (2.718281828….).

(110)
$$e-1=\sum_{k=1}^{\infty}\frac{1}{k!}$$

Because 1/k! gets small fast, the summation term in the Total is well-approximated by the infinite series, so we have

(111) Total $\approx$ $(e–1)\, n!$

In fact, the approximation is so good that the Total is always equal to the greatest integer in the approximating expression, its integer 'floor':

(112) Total = floor $[(e–1)\, n!]$

Thus, the worst case grows factorially in the number in constraints. This situation amounts to an exhaustive listing of all possible rankings on the constraint set — with duplications.

The worst case only arises, of course, in cases with an unnaturally high degree of symmetry. We'd need *n* constraints where each row and each column contains a single W, with the rest *e*'s. (Imagine a square array with W's along the diagonal.) This particular arrangement won't present itself often in practice. More realistically, we could have *n*+1 constraints, with *n* of them as described and the $(n+1)^{st}$ all L's. The worst case is not going to be encountered ecologically, but we can still expect a certain amount of complexity in the naturalistic applications of the algorithm.

The best case arises when the ERC set demands a linear order. At each application, there is only one info loss residue. Including the original set, only *n*–1 sets are calculated, one for each required ordering relation.

## 4.4 Relation to Other Work

Here we mention two related algorithms, one with a different functionality but similar structure; the other with similar goals but a very different mode of approach.

(1) **An abstract resemblance**. In "Fundamental Properties of Harmonic Bounding," Samek-Lodovici and Prince (2002) present an algorithm that takes as input a set of violation profiles and produces as output a set of profiles that are sufficient to 'simply bound' everything

that is bounded, simply or collectively, by the original set. This algorithm, which was developed in complete independence of FRed and for very different purposes, has a structure that resembles FRed's in a number of respects. The columns of a matrix (of violations) are examined for a certain property (minimal violation value); and for each column, a summary profile is constructed with respect to the minimal values; and those rows that are implicated in the property are gathered for further recursive processing.

There are strong direct connections between harmonic bounding and ERC entailment (ERA:35-46), and, more abstractly, formal correspondences between single ERC entailment and simple bounding, set entailment and collective bounding, which arise from shared order properties, since both involve coordinatewise order in their respective vectorial domains (Prince 2005) as well as the lexicographic order basic to OT. The goals of the two related algorithms remain distinct, even viewed abstractly — not just because the bounding reduction algorithm operates in the domain of violation profiles and harmonic bounding, but because it seeks to reduce the problem entirely to simple bounding, where FRed is happy with set entailments from a basis.

(2) **A shared goal.** The matter of ridding an ERC set of logical dependencies and W-compliant sets is taken up in ERA:31-34. The technique proposed there for dealing with entailments is based on the observation that ERC logic supports a form of negation, there called the 'negative', which is obtained by permuting the terms of the comparison, so that the desired winner becomes the desired loser. The formal effect is to switch W and L (ERA:12ff). For all ERCs except the 'degenerate' one consisting of all *e*'s, the negative works exacty like classical negation. In particular, we have the familiar relationship between inconsistency/unsatisfiability and entailment:

(113) **Entailment and ERC negative**. $\mathbf{A}\vDash\mathbf{a}$ iff $\mathbf{A}\cup\{-\mathbf{a}\}$ is unsatisfiable.
Proof. ERA:13, Lemma (23).

(In ERA, this is used to prove that fusion delivers all nontrivial entailments, Prop. 2.5, ERA:14.) The tractable behavior of the ERC negative gives us a very efficient way to determine if any $\mathbf{a}\in\mathbf{A}$ is entailed by other vectors in **A**. We simply replace **a** with $-\mathbf{a}$ in **A** and ask if the result is inconsistent. If it is, then **a** is entailed; if not, then not. RCD makes this a quick calculation, since it is highly efficient and fully capable of detecting inconsistency. To rid **A** of entailments, we simply go through it, ERC by ERC, checking in this fashion and discarding the entailed ERCs that we find.

In the ERA algorithm, the entailment check is to be preceded by a winnowing of the W-compliant subsets (ERA:32-33). In many cases, the result of this two-barreled procedure will be identical to that of FRed, but there is one key case where they diverge: where a subset is *fusionally reducible* in the context of the whole, but not locally W-compliant. Example (47), repeated here for convenience, provides an example:

(114) **Fusionally reducible but not internally W-compliant**

| **A** | **1** | **2** | **3** | **4** | ERC |
|---|---|---|---|---|---|
| **a** | W | L | e | **W** | **1≫2 ∨ 4≫2** |
| **b** | W | e | L | ***e*** | **1≫3** |
| **c** | e | W | e | L | **2≫4** |
| **a∘b∘c** | ***W*** | ***L*** | ***L*** | ***L*** | **1≫{2,3,4}** |

There are no W-compliant subsets, yet {**a**,**b**} is fusionally reducible, as its members are pairwise W-compliant with the fusion of the whole.

Thus, for the goal of producing the MIB, only FRed will work. For the limited goal of determining whether an individual ERC is entailed by a specific set of vectors, the efficient ERA technique remains unrivalled.

## 5. Proleptic Conclusion

The Ranking Problem in OT asks for the necessary and sufficient ranking conditions imposed by candidate data in which the desired optima have been previously identified. This problem plays out in the realm of comparative tableaux, which display the results of the fundamental calculation that extracts the relative performance of each optimum~suboptimum pair on every constraint. At the heart of the comparative tableau are the vectors of W,L, and *e* values recording the relative successes, failures, and no-contests involving the desired optima. They support a calculus that enables computation of the desired logical properties.

Because it is the maximally concise, maximally informative set equivalent to any vector set, the MIB or 'Most Informative Basis' provides the definitive and complete answer to the Ranking Problem. In Part II, we prove its existence and uniqueness, and in the process, define and explore the ordering relations between entire sets of vectors that underly the notions of interest. The first of these comes from the arrow relation, and supports the notion of relative informativeness between bases, which leads to the Most Informative Basis. Another, crucial to uniqueness, emerges naturally from operation of fusion.

The Fusional Reduction algorithm FRed computes the Fusional Normal Form (FNF) of a set of vectors by the steps laid out and examined here. In Part II, we prove that it is correct to identify the FNF with the MIB, as we have done here. We also examine in detail the relation between FRed and entailment, showing that the local, fusion-based Entailment Check step is sufficient establish the global logical independence of any fusion retained in the FNF.

Part II explicates the relation between the tree structure generated by an application of FRed and the constraint rankings — the models — compatible with the FNF. Under this rubric, the relation between RCD and FRed is formally examined. We show that RCD is FRed recursively applied to the intersection of info residues, instead of being applied to each info residue separately, as in the production of the FNF; and we show that this is equivalent to saying that, on contractional constraint sets created by constraint fusion, as discussed in section 4.2, RCD and FRed are identical.

## References

ROA = http://roa.rutgers.edu

Anderson, Alan Ross & Nuel D. Belnap, Jr.1975. *Entailment: the logic of relevance and necessity. Volume 1*. Princeton University Press.

Brasoveanu, Adrian. 2003. Minimal Fusion Normal Form. 2003. Ms. Rutgers University, NB. http://www.rci.rutgers.edu/~abrsvn/pdfs/OT/MFNF.pdf.

Brasoveanu, Adrian and Alan Prince. 2004. Fusional Normal Form. Talk given at HUMDRUM, Rutgers University, New Brunswick.

Gigerenzer, Gerd and Daniel G. Goldstein. 1996. Reasoning the Fast and Frugal Way: Models of Bounded Rationality. *Psychological Review* 1996, Vol. 103, No. 4, 650-669.

Gigerenzer, Gerd, Peter M. Todd. and the ABC Research Group. 1999. *Simple heuristics that make us smart.*Oxford University Press. New York.

Grimshaw, Jane. 1997. Projection, Heads, and Optimality. LI 28.4, 373-422; ROA-68.

Merchant, Nazarré. 2004. FRed: a Java implemenation. Ms. Rutgers University, New Brunswick

Meyer, Robert K. 1975. Chapters 29.3 and 29.12 in Anderson & Belnap 1975.

Parks, R. Zane. 1972. A note on R-Mingle and Sobociński's three-valued logic. *Notre Dame Journal of Formal Logic* 13:227-228.

Prince, Alan. 1998. A proposal for the reformation of tableaux. ROA-288.

Prince, Alan. 2000. Comparative Tableaux. ROA-376.

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

Prince, Alan. 2002b. Arguing Optimality. ROA-562.

Prince, Alan. 2003. The logic of Optimality Theory. Invited talk, SWOT, UAz, Tucson. http://ling.rutgers.edu/gamma/SWOT.pdf.

Prince, Alan. 2005. LSA 238 Lectures. http://ruccs.rutgers.edu/~prince/ot.html

Samek-Lodovici, Vieri and Alan Prince. 2002. Fundamental Properties of Harmonic Bounding. http://ruccs.rutgers.edu/tech_rpt/harmonicbounding.pdf. Corrected 2005 as ROA-785.

Sobociński, Bolesław. 1952. Axiomatization of a partial system of three-valued calculus of propositions. *The Journal of Computing Systems*, 1:23-55.

Tesar, Bruce and Paul Smolensky. 1993. The Learnability of Optimality Theory: an Algorithm and some Basic Complexity Results. ROA-2.

Tesar, Bruce. 1995. *Computational Optimality Theory*. Ph. D. Dissertation, University of Colorado at Boulder. ROA-90.

Tesar, Bruce & Paul Smolensky. 2000. *Learnability in Optimality Theory*. MIT Press.