Reducing the Families

# Grimshaw 2001 against economy constraints

## OT is intrinsically an economic system – every process and structure violates some principle

## e.g. every structure built on another will have a head and a complement at least – one of these will be leftmost, so the other will violate the leftmost constraint

## every movement requires extra structure and extra structures violate leftmost constraints

## therefore there is no need for specific economy constraints

# Newson 2003 against markedness constraints

## constraints typically come in conflicting pairs

### X precedes Y – X follows Y

## these obviously cannot be satisfied at the same time unless they are vacuously satisfied

## unfaithfulness is one way to vacuously satisfy a constraint

### i.e. if X is deleted both X prec Y and X foll Y will be satisfied

## ranking faithfulness constraints above conflicting position constraints will force the element to be realised, ranking it below will force it to be unrealised.

## This is the same as markedness

# Alignment Syntax – structure and mapping reduce to alignment

## Structure, mapping and alignment constraints are all positional constraints – they place elements in certain positions

## Thus there is a redundancy involved here.

## Both structure and mapping can be satisfied by specific alignments

### ObHd = finite element must immediately follow wh-element

### OpSpec = operator precedes every element it has scope over

## Alignments cannot be satisfied by structure and mapping without other assumptions being made

### that X precedes Y might be able to be enforced by making X be in the specifier and Y be in the head position of some phrase (assuming that SpecFirst dominated HdFirst) but then X must be a phrase and Y must be a head, so this couldn’

## therefore it is simpler to get rid of structure and mapping constraints and reduce the redundancy by having alignment constraints

# Consequences of reduction to alignments

## If structural alignments are allowed, then structures must be part of the candidate set

### So GEN must create structure

## If non-structural alignments only are allowed, then there cannot be structure in the candidate set – something that would be unconstrained would not be possible as there would be no reason to choose one candidate over a potential infinite number of different candidates

### GEN only needs to generate orders over input elements

### There are only a finite number of orderings over a finite number of elements and hence the candidate set is finite under this assumption.

### GEN is very simple – the only relevant syntactic processes concern ordering and deletion