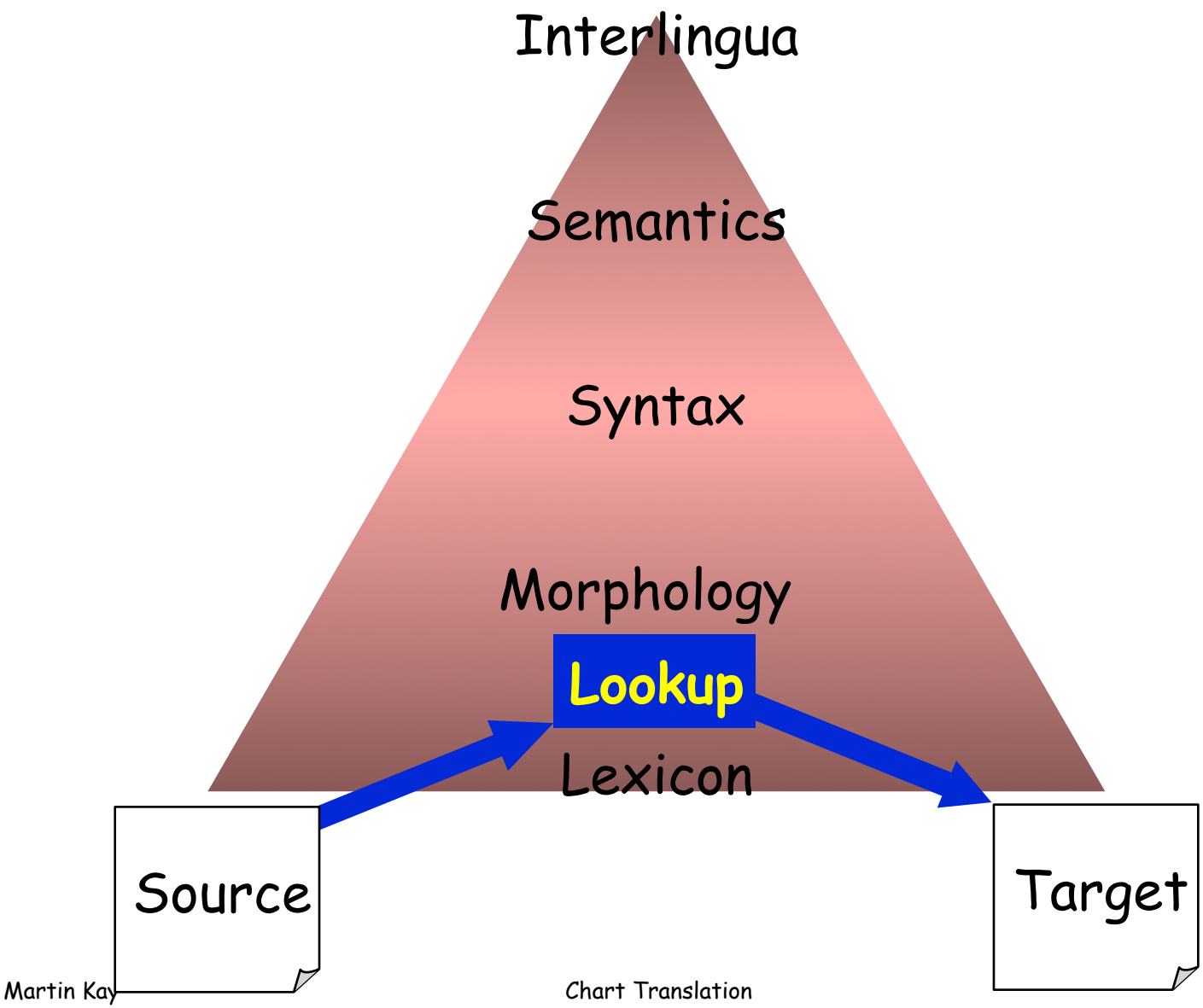
Transfer

Martin Kay

Stanford University

and the University of the Saarland

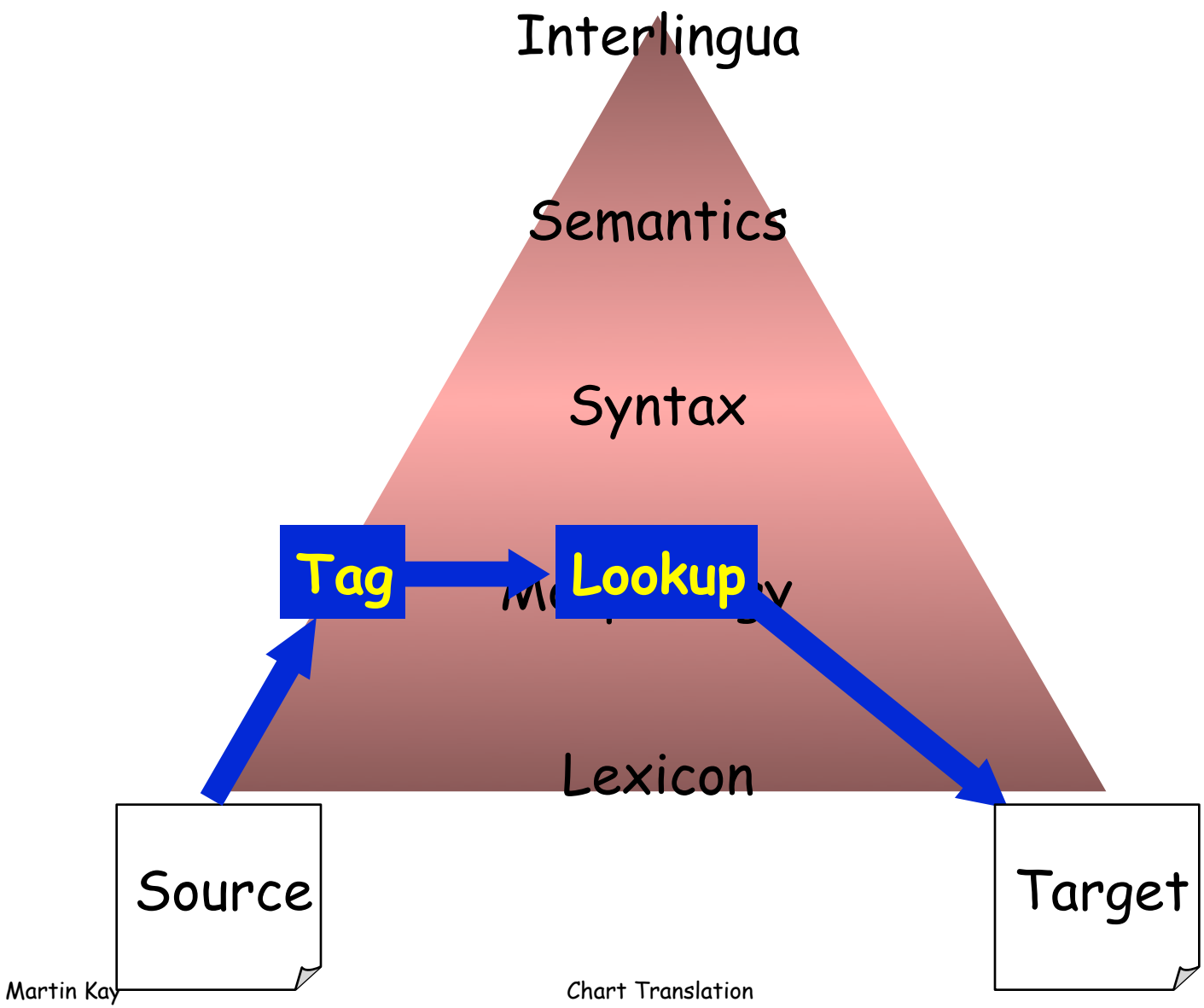

Interlingua
Semantics
Syntax
Morphology
Lookup
Lexicon
Source
Target

Word for word

- Word boundaries
- Ambiguity
- Word order

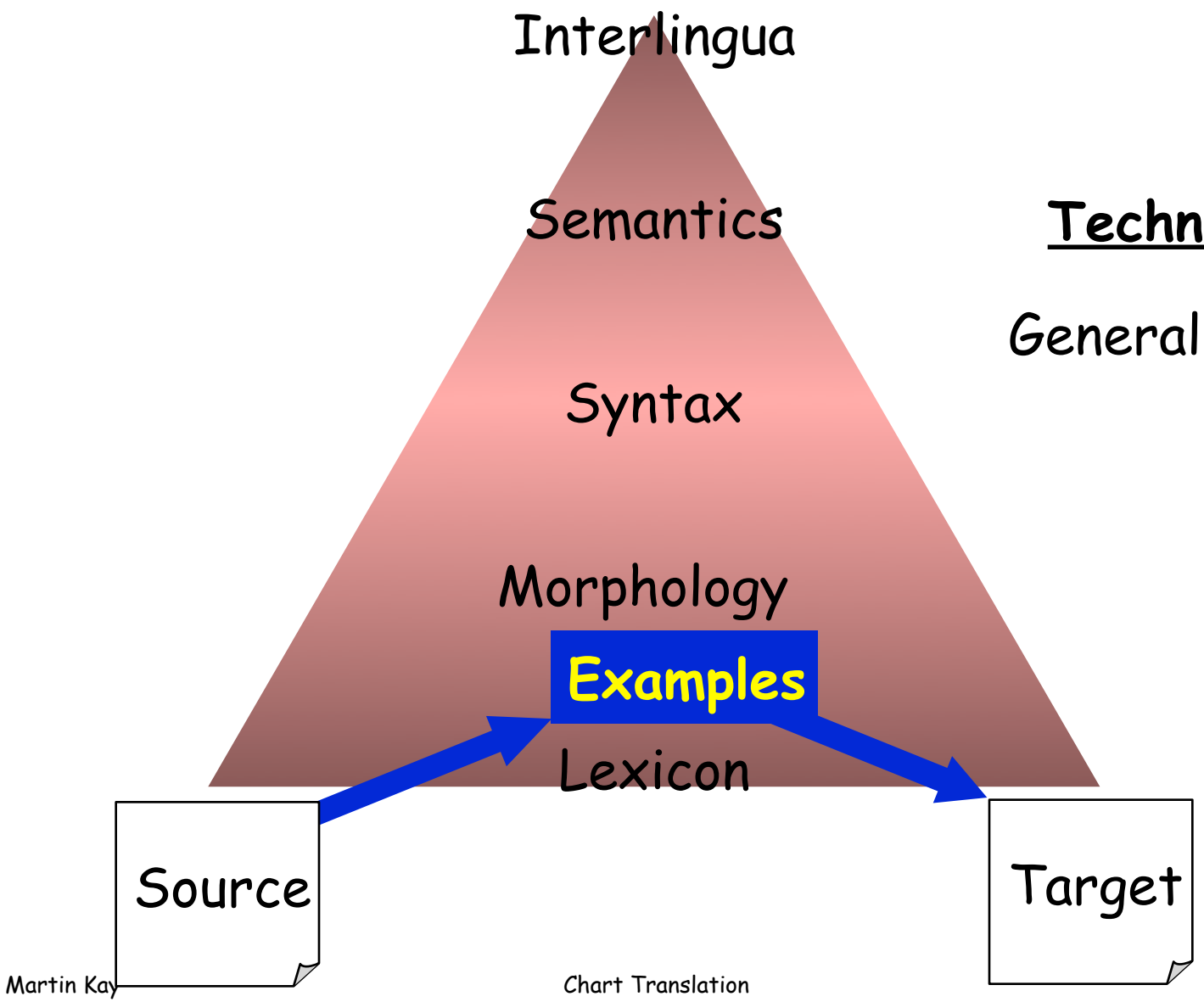
Lookup by ...

- Hashing
- Tries


Interlingua
Semantics
Syntax
Tag
Lookup
Lexicon
Source
Target

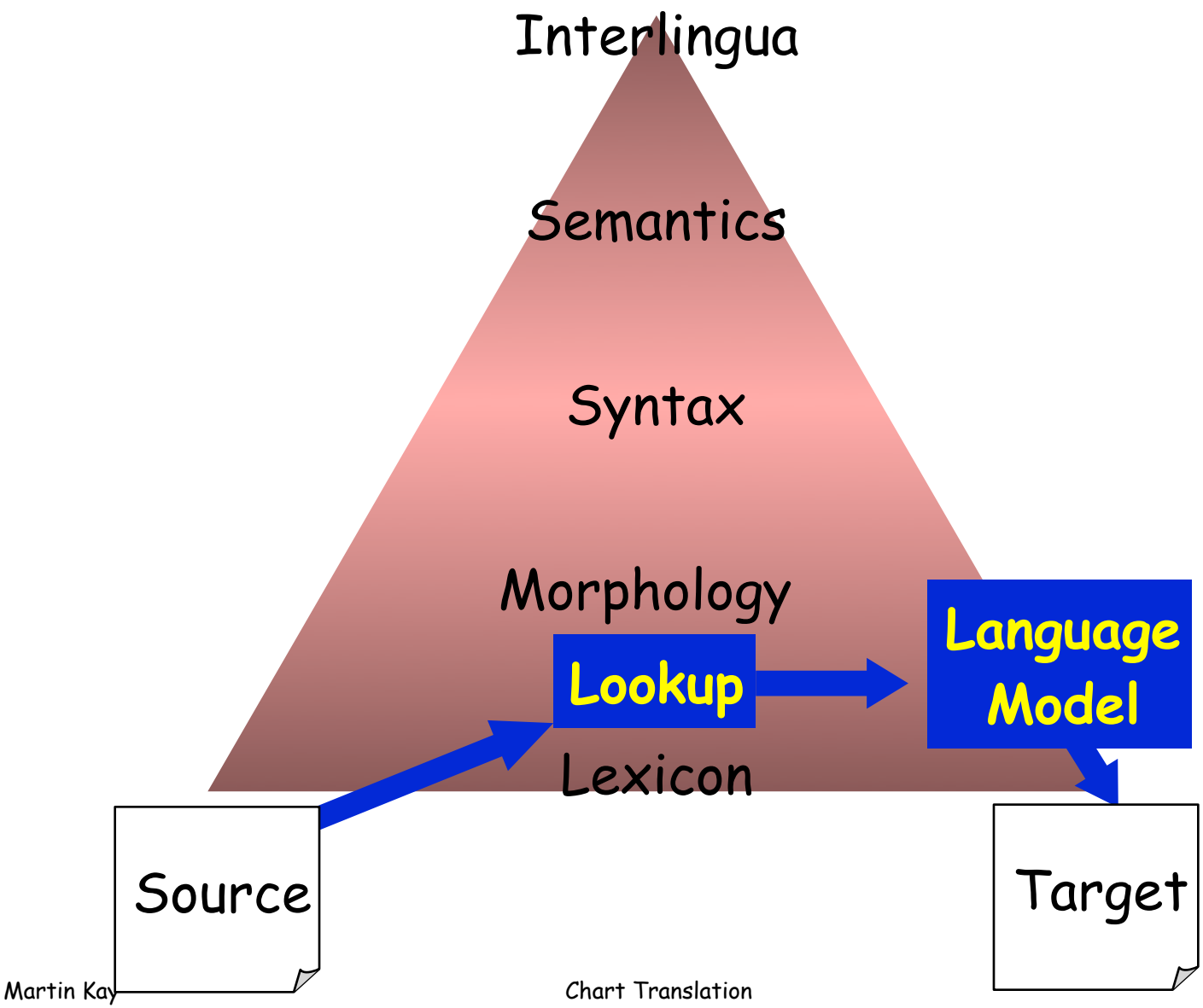
Tagging by ...

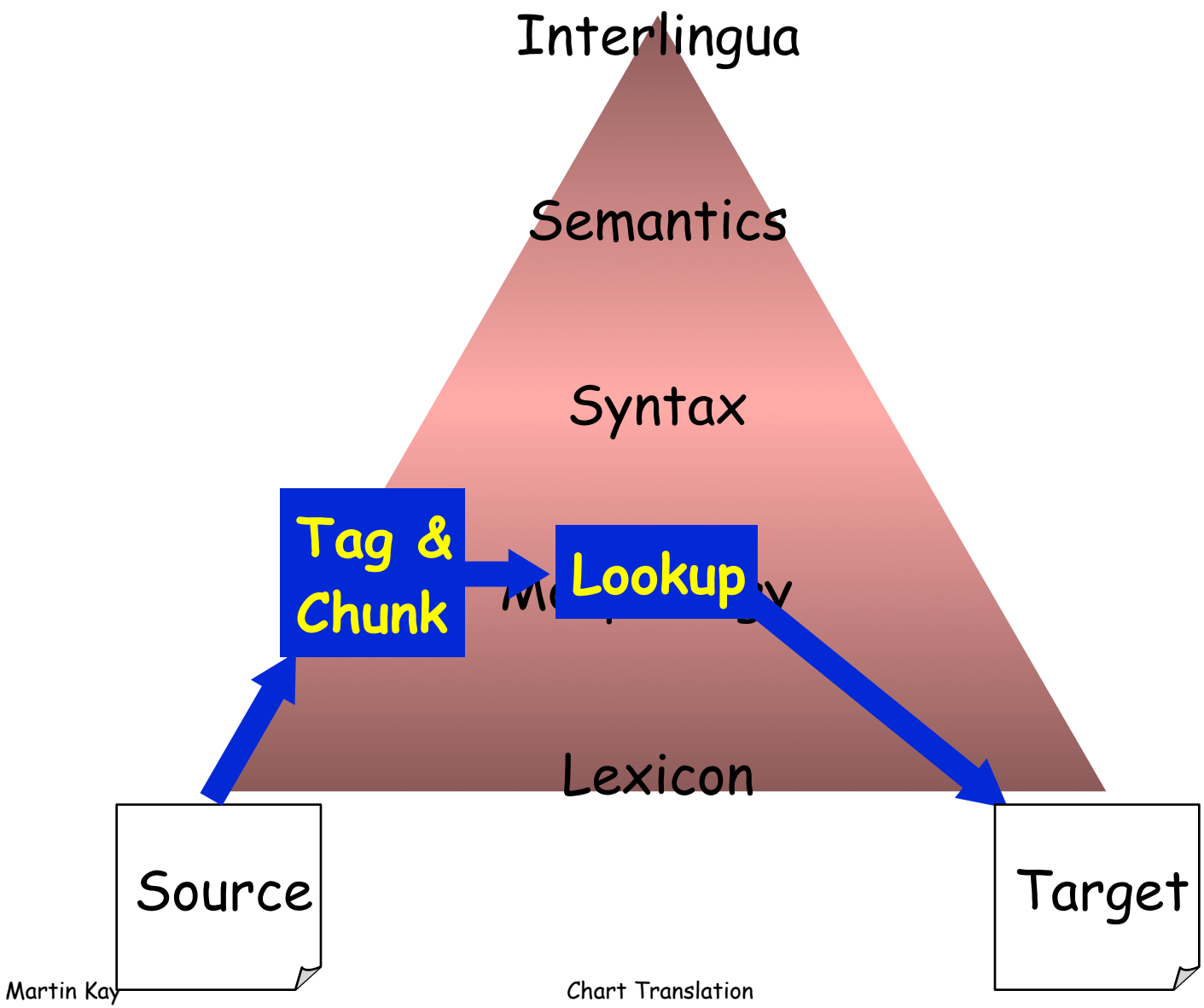
- Custom designed rules
- Machine Learning
  - Error-based learning
  - Hidden Markov Models
  - Decision Trees

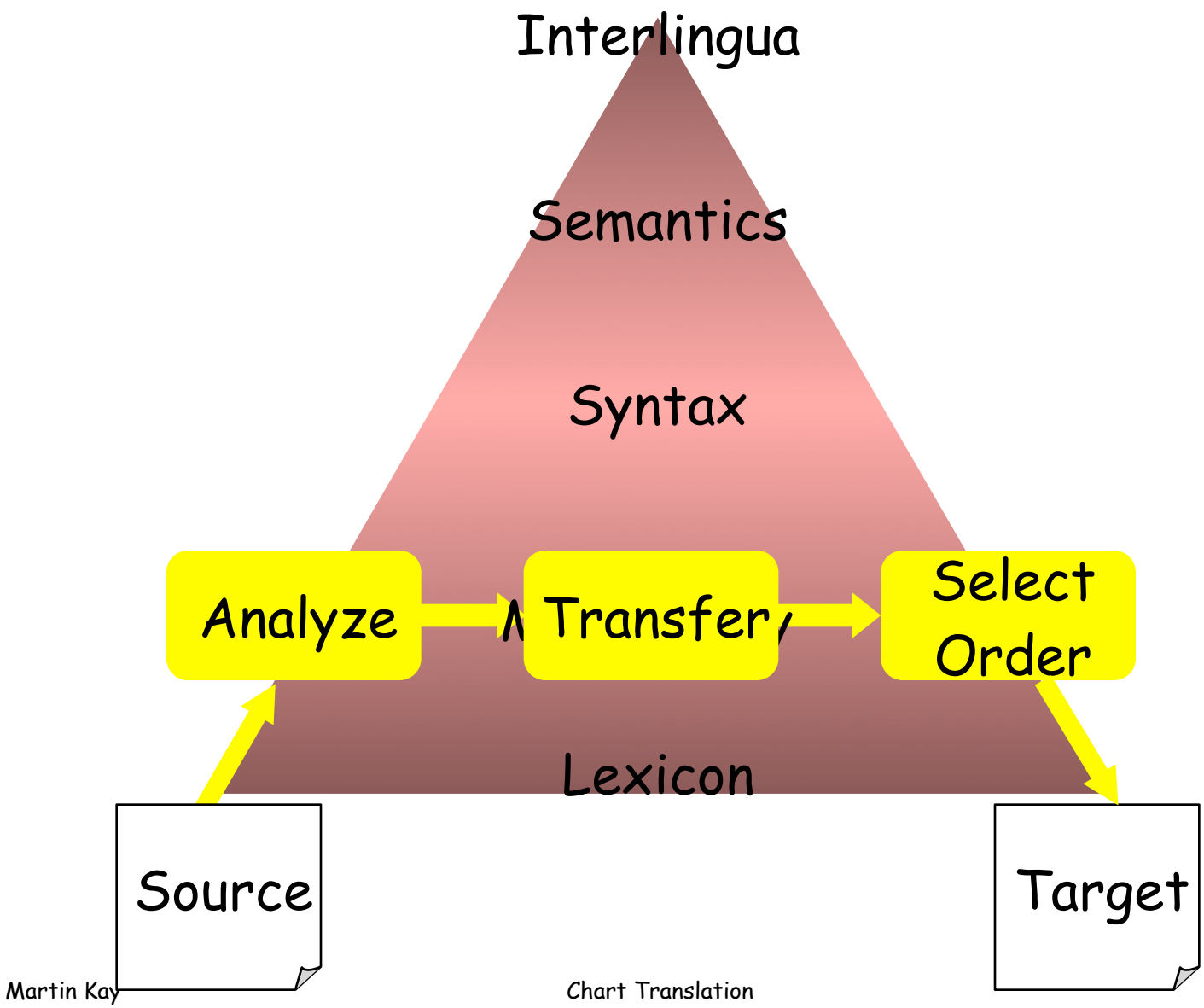


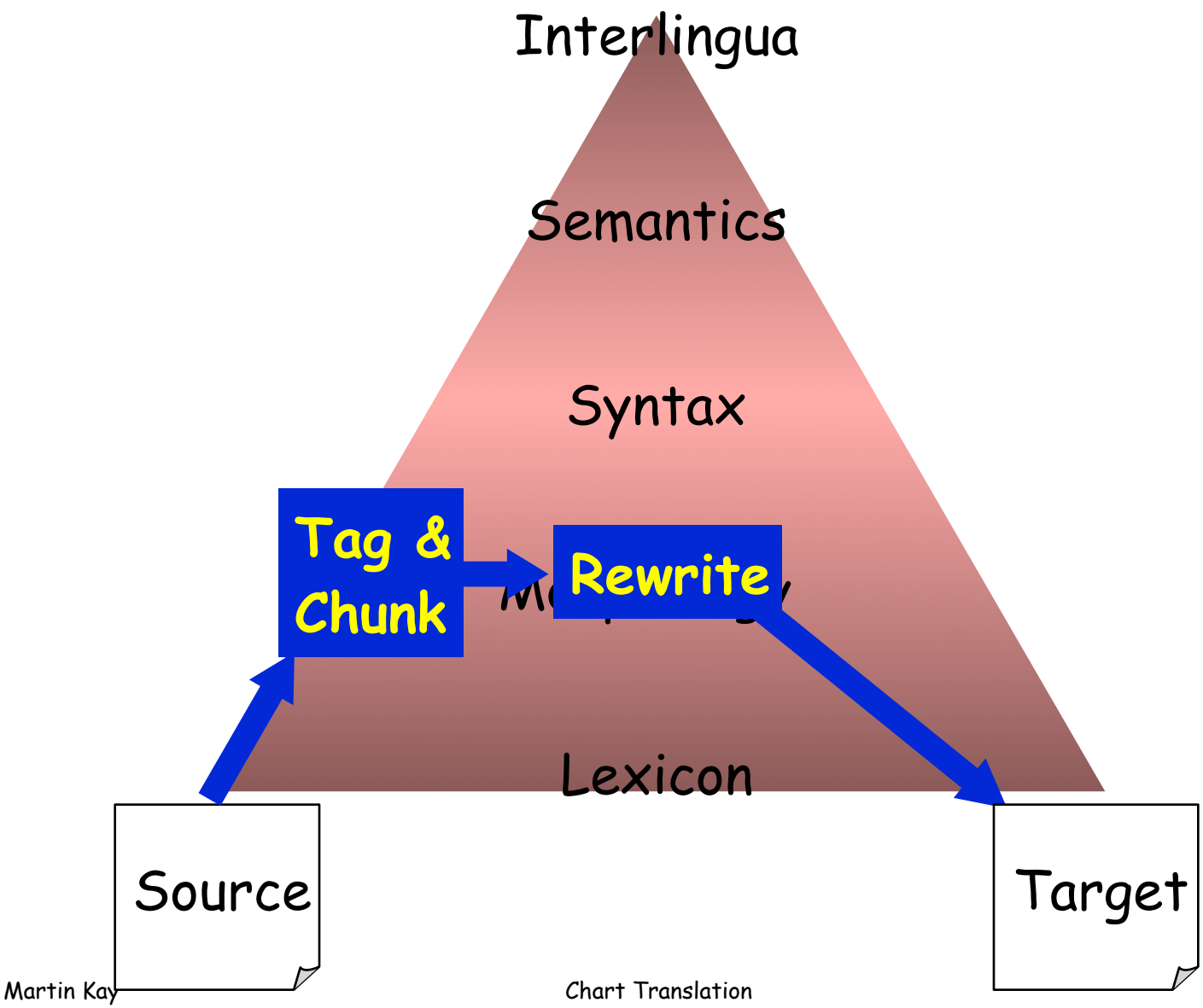
Techniques

Generalization


Interlingua
Semantics
Syntax
Morphology
Lookup
Language Model
Lexicon
Source
Target


Interlingua
Semantics
Syntax
Tag &
Chunk
Lookup
Lexicon
Source
Target


Interlingua
Semantics
Syntax
Analyze
Transfer
Select Order
Lexicon
Source
Target


Interlingua
Semantics
Syntax
Tag &
Chunk
Rewrite
Lexicon
Source
Target

Rewriting rules

String rewriting

$\alpha \rightarrow \beta \ / \ \gamma \_ \ \delta$

α, γ δ are strings / regular expressions allowing variable assignment

β is a string

Set rewriting

$\alpha \rightarrow \beta \ / \ \gamma$

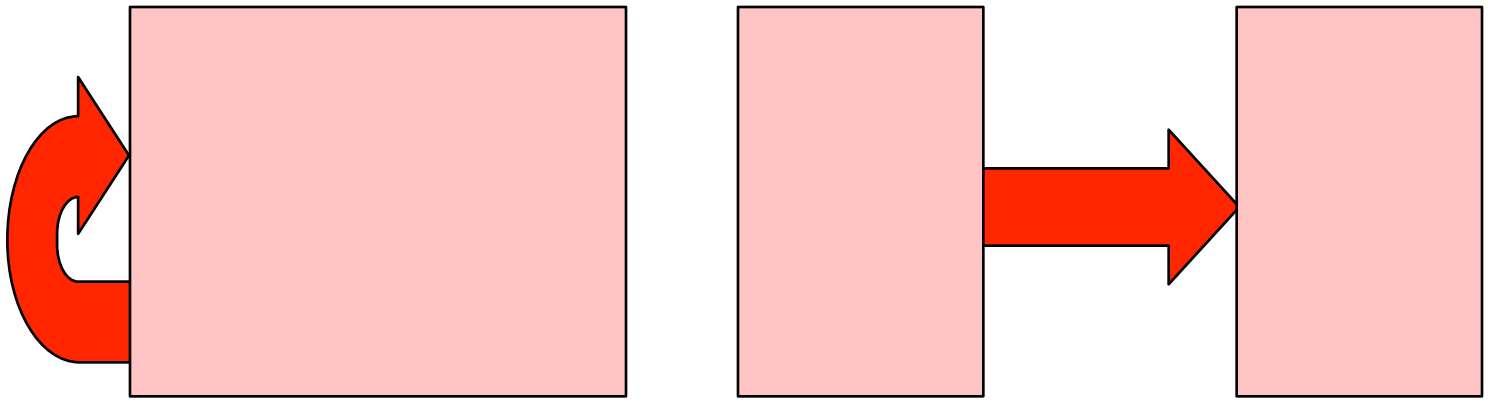
α, β are sets allowing variable assignment

γ is a set

Rewriting rules

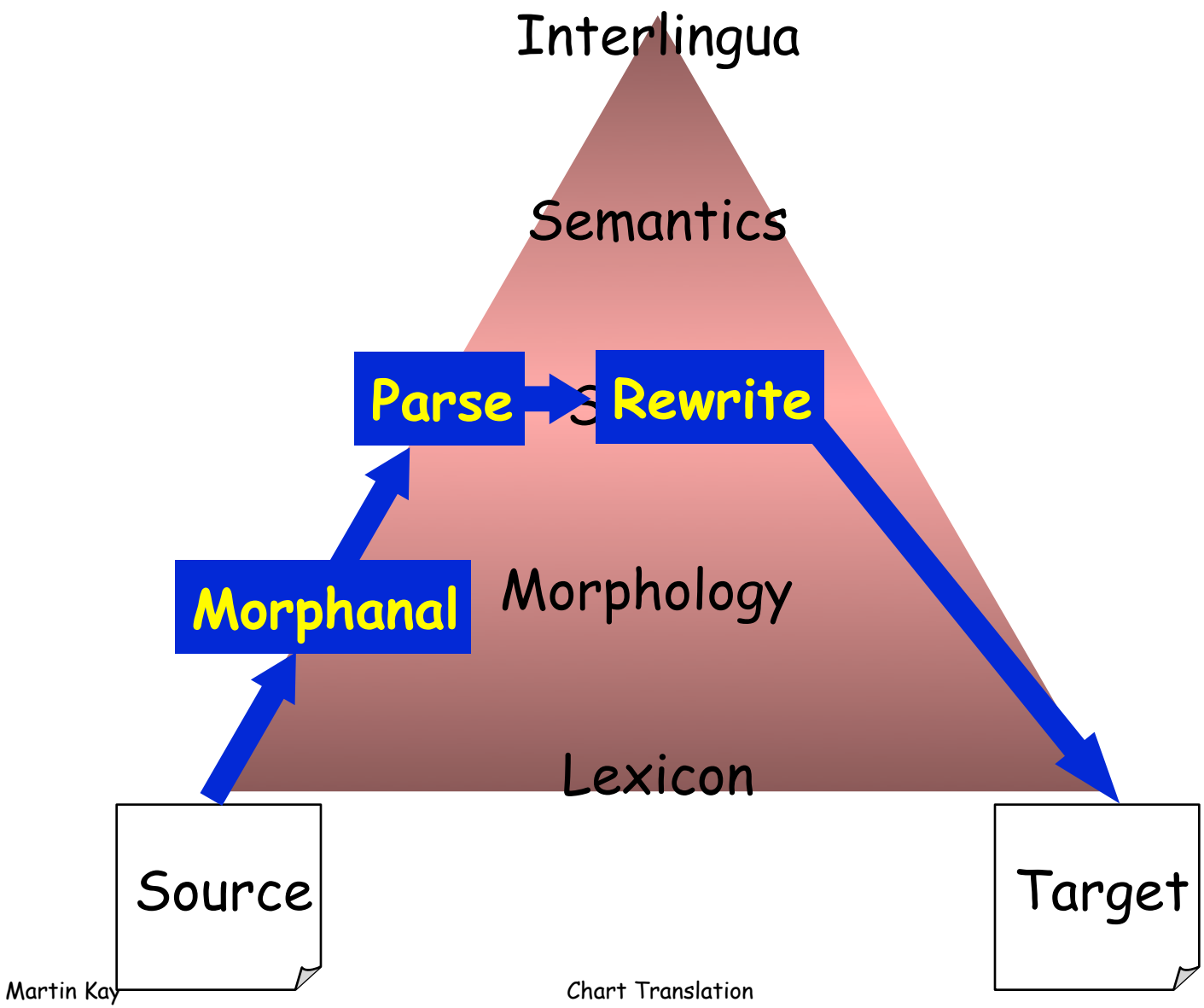
Cycling

Type Changing



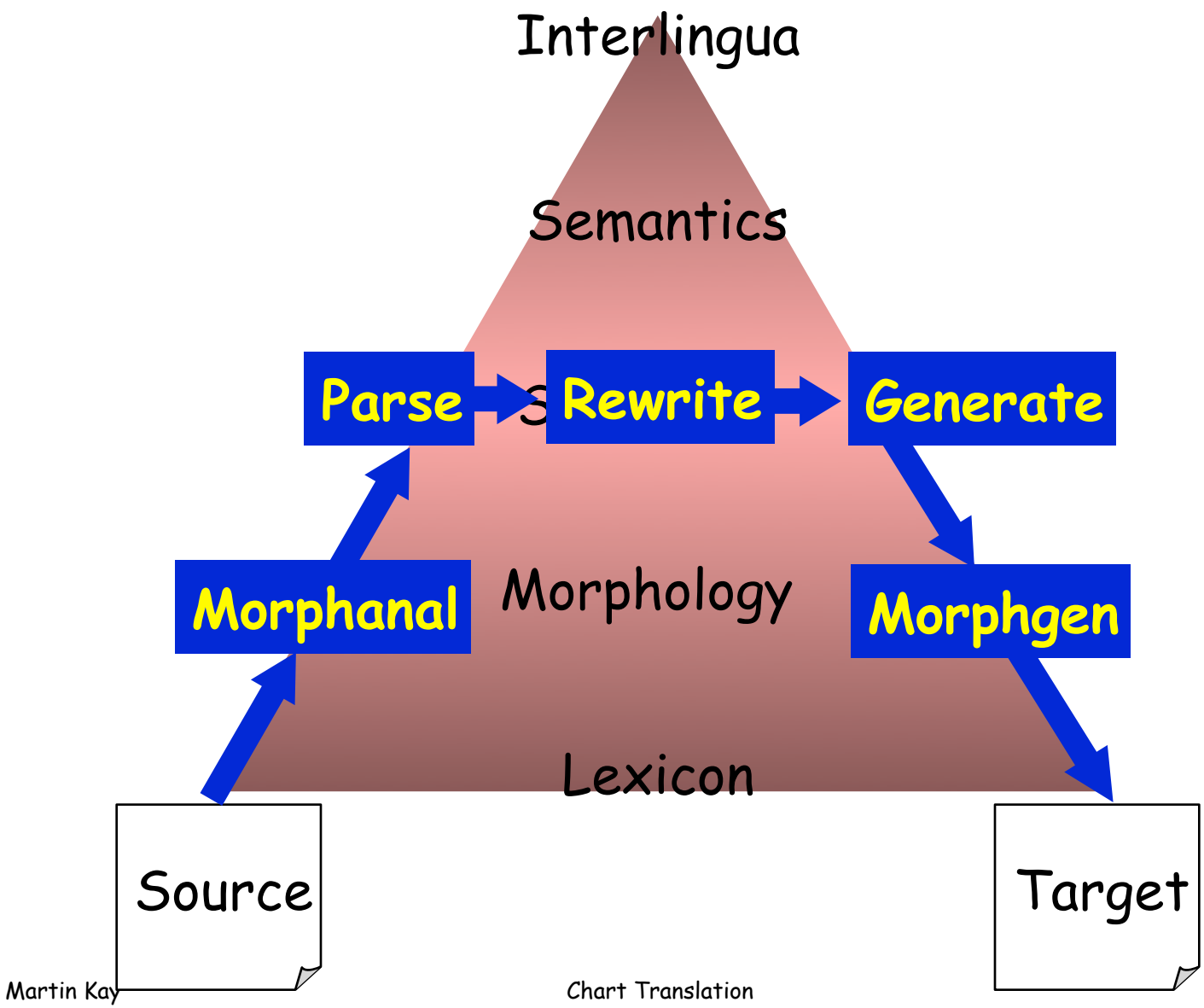
Ordered

- Feeding & bleeding
- Human stipulated order
- Order by specificity


Interlingua
Semantics
Parse
Rewrite
Morphanal
Morphology
Lexicon
Source
Target

Parsing

- **Chunks**
  - **Ad hoc rules**
  - **Finite-state**
- **Trees**
  - **Context free**
  - **Feature-based grammar**
- **Feature Structures**
  - **Unification grammar**


Interlingua
Semantics
Parse
Rewrite
Generate
Morphanal
Morphology
Morphgen
Lexicon
Source
Target

Interlingua

UNL

- **UNL is an electronic language for information exchange and transfer over the Internet, consisting of logical expressions that represent the meaning of sentences in natural languages.**
- **Mary broke the window in UNL will be represented by some logical expressions like:**

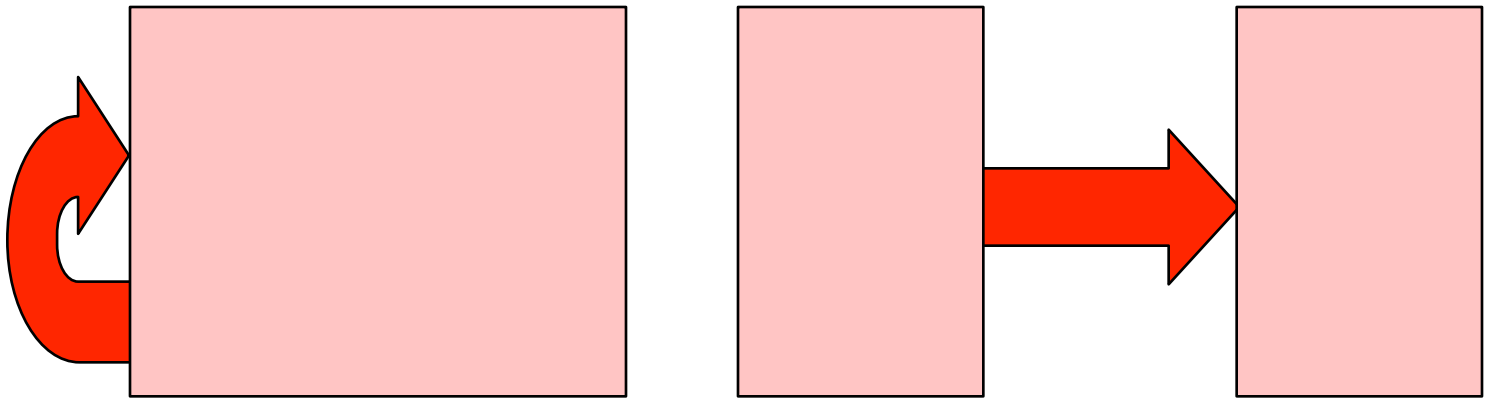
```
agt(break(icl>do).@entry.@past, Mary) obj(break
(icl>do).@entry.@past, window
(icl>thing).@def)
```

Example-based MT

Rewriting rules

Cycling

Type Changing



Rules apply to sets of predicates

The PARC Transfer System

Transfer from LFG F-Structure to

F-Structure through the intermediary of

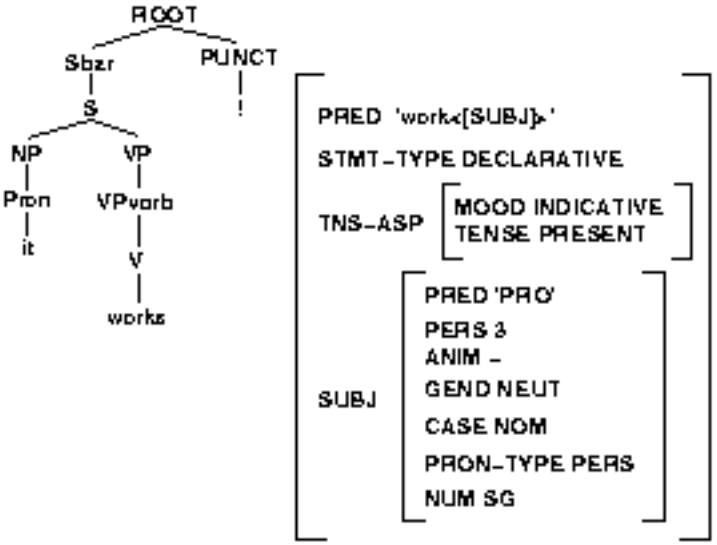
a set of Prolog predicates.

Not necessarily limited to F-Structure. Later extended to C-Structure

Characteristics

- A configuration consists of one set of predicates. Not one for each language.
- Completely deterministic.
- Ordered rules: Each rule operates on the configuration left by the preceding one.
- Anything that is not explicitly transferred to the target language remains in the source language

Flat Structure




```
pred (0, word)
smt_type(0, declarative)
tns_asp(0, 1)
mood(1, indicative)
tense(1, present)
subj(0, 2)
pred(2, pro)
pers(2, 3)
anm(2, '-')
gend(2, neut)
case(2, nom)
pron_type(2, pers)
nom(2, sg)
```

Typically the grammar begins with a bunch of deletion rules. These rules primarily delete language specific terms which are not supposed to affect the translation relation.

```
aux_select(_,_) ==> 0.
status(_,_)  ==> 0.
gend(_,_)    ==> 0.
prog(_,_)    ==> 0.
anim(_,_)    ==> 0.
ntype(_,_)   ==> 0.
grain(_,_)   ==> 0.
proper(_,_)  ==> 0.
```

Simple rules to transfer specific words or attributes, look like:

```
vtype(X,main)==> vtype(X,unspec).
pred(X, be)  ==> pred(X, être).
```

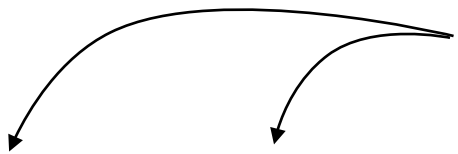
Repetitive transfer rules can be encoded using templates. A template can be defined as any prolog term, including any of the operators provided by prolog (such as '->' or '-'). Templates can call other templates and also combine with other templates and rules.

```
p2p(Source, Target) ::
   pred(X,Source) ==> pred(X,Target).

Source -> Target ::
   p2p(Source, Target).

dog    -> chien.
cat    -> chat.
saw    -> voyait.

sleep  -> dormir.
run    -> courir.
come   -> venir.
walk   -> marcher.
```

```
proper_name(Source, Target) ::
  pred(X,Source) ==> pred(X,Target).

proper_name('John', 'Jean').
proper_name('Mary', 'Marie').
```

Templates can also be redefined, which is especially useful for using the convenient and mnemonic operators.

```
Source -> Target ::
   spec_form(X,Source) ==> spec_form(X,Target).

the    -> le.

Source -> Target ::
   pron_form(X,Source) ==> pron_form(X,Target).

'I'    -> je.

Source -> Target ::
   pred(X,Source), +subj(X,Y), +pred(Y,_Noun) ==> pred(X,Target).

good   -> bon.
young  -> jeune.
left   -> gauche.
right  -> droit.
```



In addition to templates, the system provides a facility for referring to a group of recurring terms together, using macros.

```
add_adjunct(X,Z)       := adjunct(X,Y), in_set(Z,Y).
adjective(X,Adj,Subj) := pred(X,Adj), arg(X,1,Subj).

pred(X,girl) ==>
  pred(X,fille),              % girl -> jeune fille
  add_adjunct(X,J),
  adjective(J,jeune,X).
```