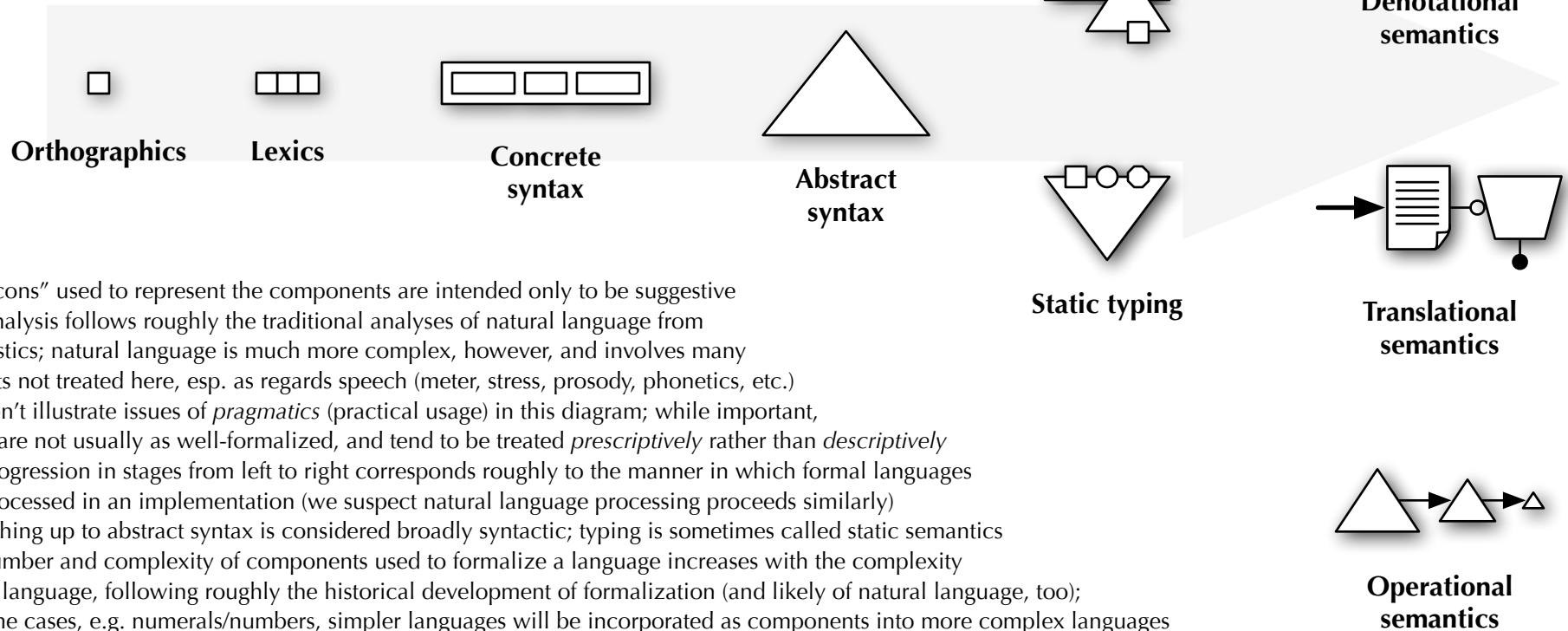


Components of formal languages

- **orthographics** — choice of symbols and their coding (e.g., Unicode encoded via UTF-8)
- **lexics** [*sic*] — (linear) combination of symbols into (unit) tokens
- **concrete syntax** — linear ordering of component parts (of abstract syntax)
- **abstract syntax** — hierarchical relations between categories and component parts
- **scoping** — delimitation of the (syntactic) region in which names have significance
- **static typing** — context-sensitive constraints on syntactic combinations
- **axiomatic semantics** — proofs of properties about abstract terms
- **denotational semantics** — mapping to conceptual values (into an intuitive meta-language)
- **translational semantics** — translation to code for interpretation by an abstract machine
- **operational semantics** — linear transition or hierarchical reduction to value terms



Notes

- the “icons” used to represent the components are intended only to be suggestive
- this analysis follows roughly the traditional analyses of natural language from linguistics; natural language is much more complex, however, and involves many aspects not treated here, esp. as regards speech (meter, stress, prosody, phonetics, etc.)
- we don’t illustrate issues of *pragmatics* (practical usage) in this diagram; while important, these are not usually as well-formalized, and tend to be treated *prescriptively* rather than *descriptively*
- the progression in stages from left to right corresponds roughly to the manner in which formal languages are processed in an implementation (we suspect natural language processing proceeds similarly)
- everything up to abstract syntax is considered broadly syntactic; typing is sometimes called static semantics
- the number and complexity of components used to formalize a language increases with the complexity of the language, following roughly the historical development of formalization (and likely of natural language, too); in some cases, e.g. numerals/numbers, simpler languages will be incorporated as components into more complex languages
- abstract syntax plays a central rôle: we might have several different semantics or concrete syntaxes for the same language (as abstract syntax); but if one abstract syntax is distinct from another (not isomorphic), we would consider this to be a distinct language
- other important concepts and dimensions: different **domains of discourse** (e.g., numbers, booleans, functions); **object language** vs. **meta-language**; **mathematical specification** (in relatively informal meta-language) versus **concrete implementation** (in an executable programming language); and **language features** (such as abstraction or imperative variables) which cut across this dimension of components orthogonally