SigmaPie for subregular grammar induction

Formal language theory explains how potentially infinite stringsets, or *formal languages*, can be generalized to *grammars* encoding the desired patterns and what properties those grammars have. It also allows one to compare different grammars with respect to parameters such as expressivity.

Grammar induction is the process of building a grammar corresponding to a given stringset. The type of grammar must be specified a priori because the formalization of the same pattern is different if expressed via grammars of different complexity. For example, regular vs. context-free.

Subregular grammars produce languages that are formally weaker than regular. Multiple patterns in phonology and morphology exhibit properties of subregular languages, therefore, this way of formalization can give an interesting perspective on the complexity of natural languages.

Grammars @ SigmaPie

- **Strictly k-piecewise** grammars prohibit occurrence of sequences of k symbols at an arbitrary distance from each other.
- **Strictly k-local** grammars prohibit occurrence of consecutive substrings consisting of up to k symbols.
- **Tier-based strictly k-local** grammars have freely distributed *non-tier* symbols and a *tier* that is a set of crucial symbols for describing the current pattern. The grammar then restricts consecutive k-long substrings consisting of the tier symbols, whereas the non-tier symbols are ignored, i.e. they can occur in any position of the string.
- **Multiple tier-based strictly k-local** grammars can implement more than a single tier.

Tools @ SigmaPie

- **Learners** extract grammars from stringsets.
- **Scanners** evaluate strings with respect to a given grammar.
- **Sample generators** generate stringsets for a given grammar.
- **FSM constructors** translate subregular grammars to finite state machines.
- **Polarity converters** switch negative grammars to positive, and vice versa.