

Real-time Grammar-based Parsing and Restructuring of Musical Streams

ICMC 2011

Dale E. Parson and Ryan R. Panuski

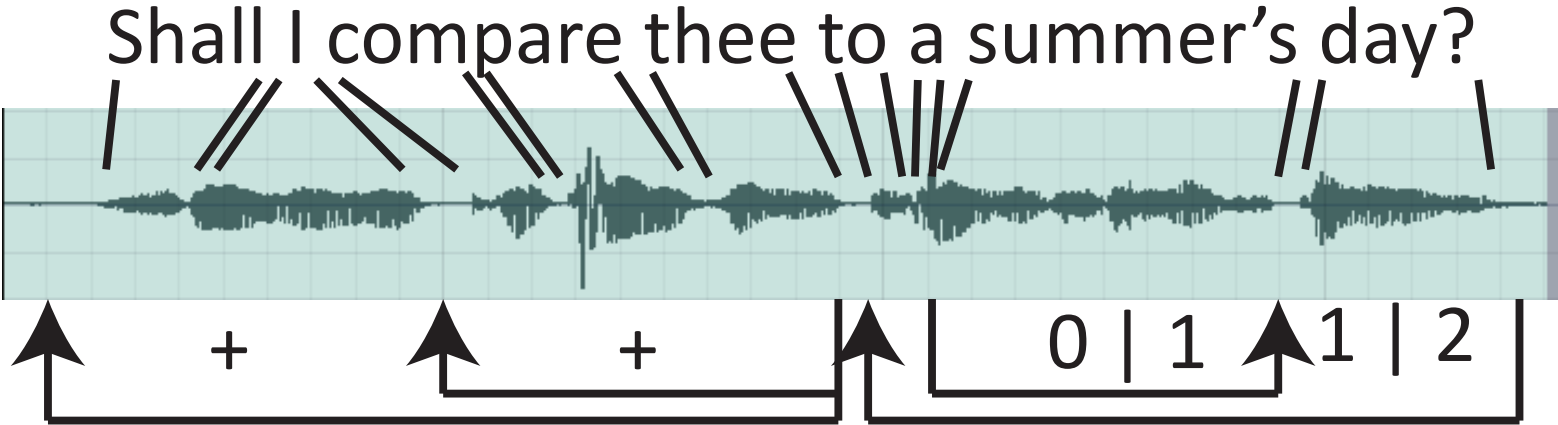
Kutztown University of PA

<http://faculty.kutztown.edu/parson>

Summary

- Looping and conventional sequencing are forms of *finite automata* specified by *regular expressions*.
- Deeper analysis and restructuring of spoken word and instrumental phrase structures require *context-free grammars* and their generated *parsers* for tagging audio streams.
- Real-time, tag-based parsing and restructuring can be incorporated in a software instrument.

Looping: A Finite State Automaton

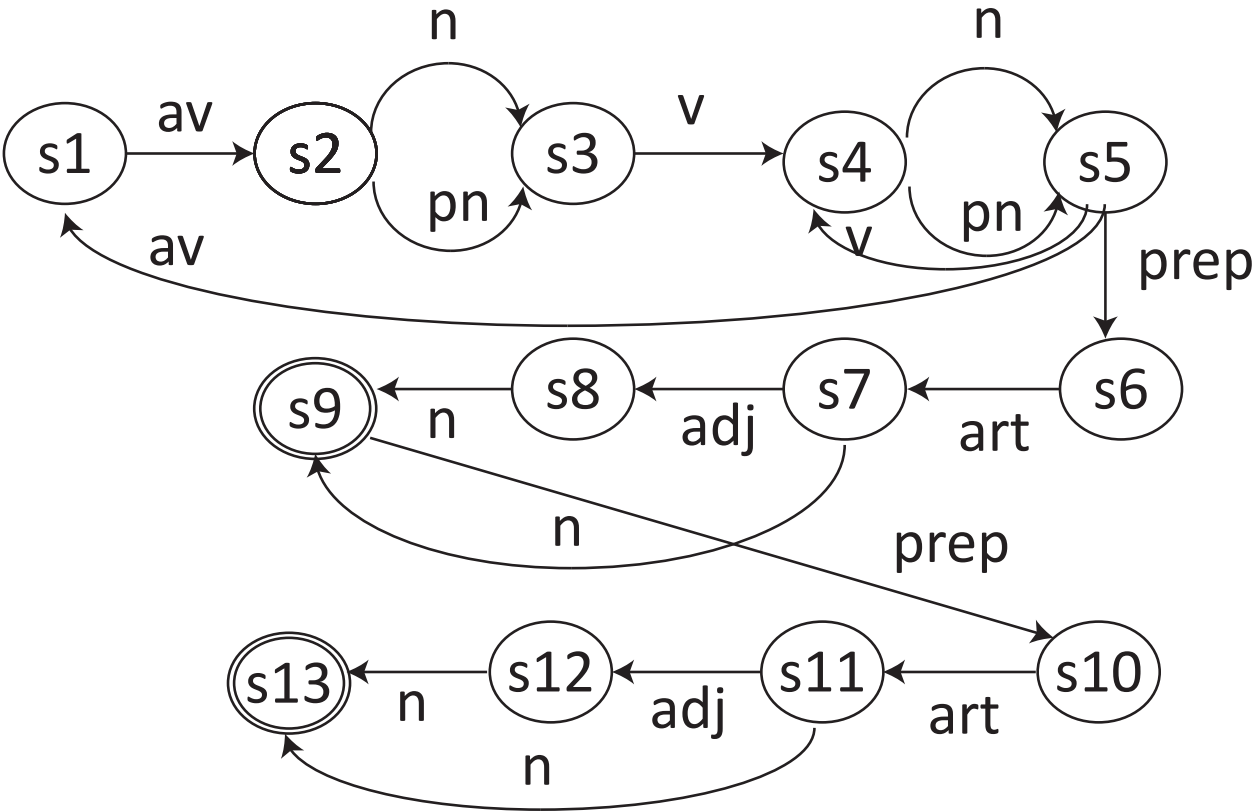


(aux-verb (noun | pronoun)(verb(noun | pronoun)))++(prep art adj₀₋₁ noun)₁₋₂

A regular expression supports sequencing, alternatives (|), repetition (* + i-j), and grouping.
A finite automaton is a machine that recognizes a sentence that conforms to a regular expression.

States, transitions, inputs, outputs

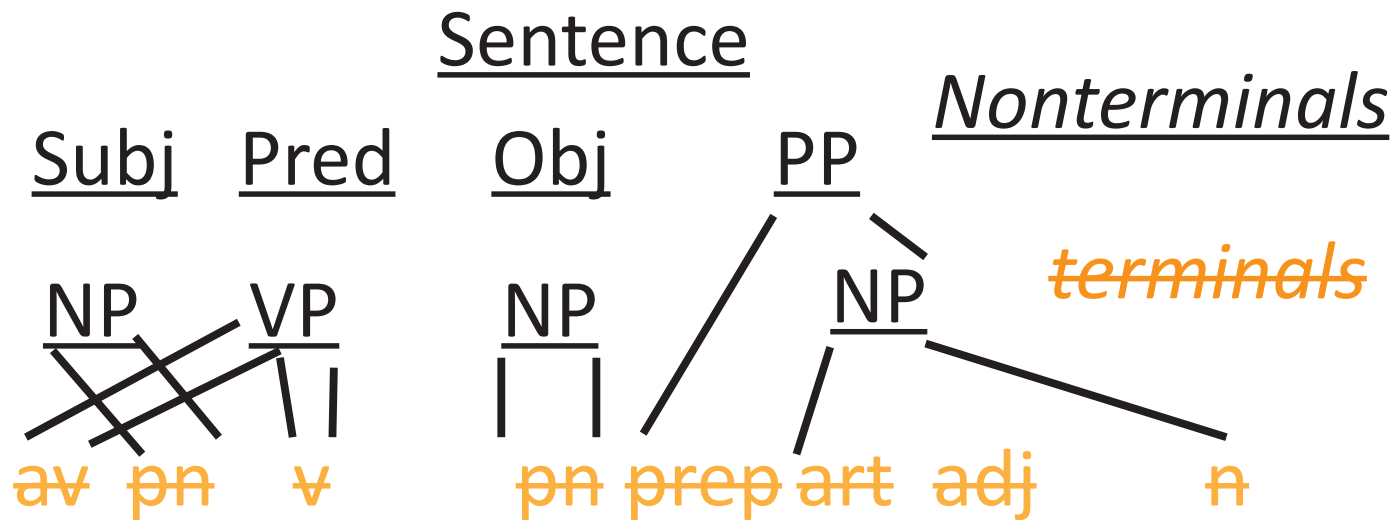
$(\text{aux-verb (noun | pronoun)(verb(noun | pronoun))}^+)^+ + (\text{prep art adj}_{0-1} \text{ noun})_{1-2}$



The Limits of Looping

- Regular expressions and their finite state automata cannot parse or generate nested phrase structures.
- Context free grammars and their parsers can!
- Organizing an audio stream as a parse tree allows tree-structured reorganization.
 - AABA -> ABAA
- Transformations may be non-contiguous.
- Phrases can nest to arbitrary depth.

A Context Free Parse of a Sonnet

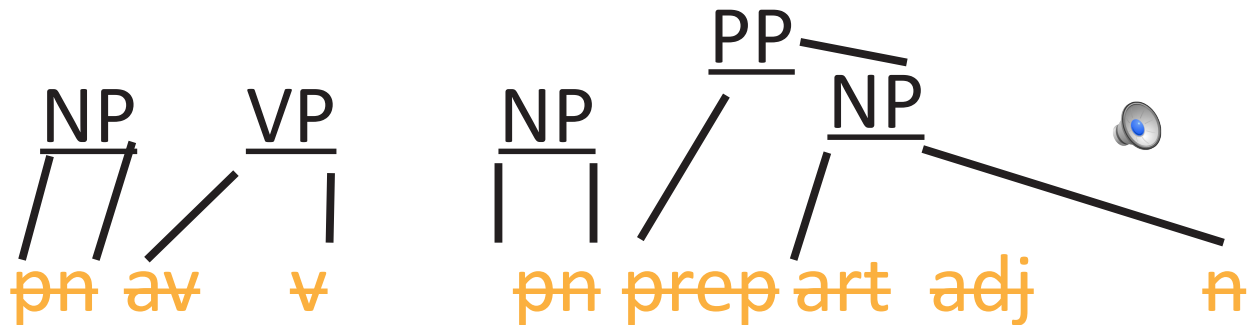


Shall I compare thee to a summer's day?

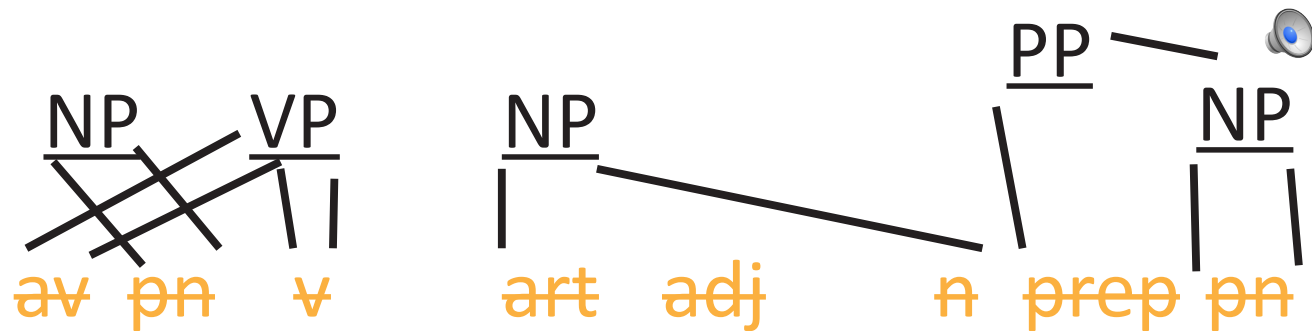
A Context Free Grammar Describes a Parser

- Sentence \rightarrow Subj Pred Obj PP
- Subj \rightarrow NP
- Obj \rightarrow NP
- Pred \rightarrow VB
- NP \rightarrow art? adj* (n | pn)
- PP \rightarrow prep NP
- PP \rightarrow *empty*

Grammar-based transforms



I shall compare thee to a summer's day?



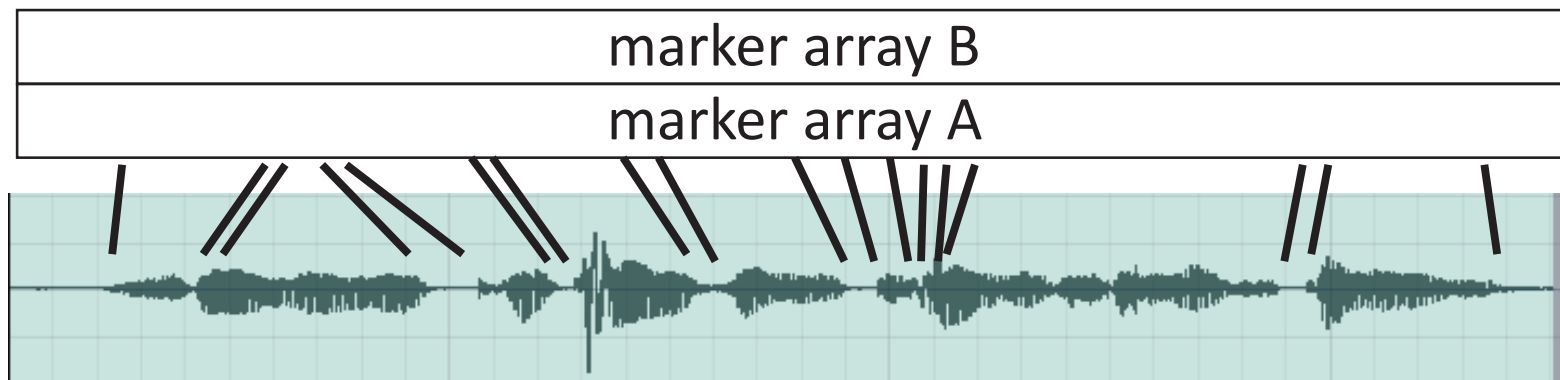
Shall I compare a summer's day to thee?

Python or Java, Chuck for Parsing

A *Python or Java* program accepts performance token marker clicks from a keyboard, controller, or GUI, builds an internal parse tree, and sends “set marker” and “rearrange marker” *Open Sound Control (OSC)* messages to Chuck.



A *Chuck* program records audio streams and stores markers to token boundaries as directed by *OSC* messages from Python.



Related and Future Work

- Grammar-based work starting in the 70's concentrates on composition and analysis.
 - Finite state looping remains the norm for performance.
- We are continuing UI work for triggering token markers (iPad) and hope to start work on a grammar visualization GUI.
- Parsing and restructuring of sounds and microsounds, parsed via sonic properties, is a promising area for exploration.