APPLICATIONS OF CONTEXT FREE GRAMMARS

BY,
BRAMARA MANJEERA THOGARCHETTI

BRIEF INTRODUCTION

- CFG is a set of recursive writing rules used to generate patterns of strings. A CFG consists of following components

A context-free grammar is a 4-tuple $(V, \Sigma, R, S)$, where

1. $V$ is a finite set called the variables,
2. $\Sigma$ is a finite set, disjoint from $V$, called the terminals,
3. $R$ is a finite set of rules, with each rule being a variable and a string of variables and terminals, and
4. $S \in V$ is the start variable.
APPLICATIONS

- CFG parsing for high speed network applications.
- Data processing.
- Natural language processing.
- Human activities recognition.
- Neural networks.
- Multi functional Radar construction.
- Software engineering requirements documentation.

MULTI FUNCTIONAL RADAR CONSTRUCTION

- MFR is used in electronic warfare (EW) field.
  - Engage multiple targets at once.
  - Signal identification.
  - Threat assessment.
- MFR’s are threat in EW because EW signal processing algorithms are not suited for MFR’s complexity.
- Accurate modeling of rules by radar control algorithm logic.
- Comparison with priori signal intelligence.
RADAR WORDS

- Decompose dynamics into hierarchical structure i.e., radar words

![Image](a) ![Image](b)

*Figure 1. Illustration of the MFR word hierarchy.*

PROCESS

- MFR signals are read in the form of strings.
- Modelled by compact syntactic representation i.e., CFG
- Derive a finite state machine.
- Directly apply to EW signal processing.
- Stochastic CFG(measurement noise)

A stochastic grammar is a five-tuple:

$$G = (A, E, \Gamma, P, S_0).$$
VERIFICATION OF NON SELF EMBEDDING PROPERTY

- Labelled production graph.
- Each vertex of the production graph corresponds to one of the non terminal symbols in E.
- For each pair of non terminal symbols $S, T \in E$ a labelled edge is drawn from node $S$ to node $E$.
- The labelled edges are $L = \{ 'b', 'l', 'r', 'u', '0' \}$
- Certain production rules are given for these edges.

RULES

- Label ‘b’ is assigned if rules are $S \rightarrow \beta T \alpha$, $S \rightarrow \beta T$ and $S \rightarrow Ta$
- Label ‘l’ is assigned if rule is $S \rightarrow \beta T$ but not $S \rightarrow Ta$ or $S \rightarrow \beta Ta$
- Label ‘r’ is assigned if rule is $S \rightarrow Ta$ but not $S \rightarrow \beta T$ or $S \rightarrow \beta Ta$
- Label ‘u’ is assigned if rule is $S \rightarrow T \alpha$ but not $S \rightarrow \beta T \alpha$, $S \rightarrow \beta T$, $S \rightarrow Ta$
- Label ‘0’ is assigned if rule there are no rules of the form $S \rightarrow T$, $S \rightarrow \beta T$, $S \rightarrow Ta$ or $S \rightarrow \beta Ta$
- Consider the following grammar

$$A = \{ a, b, c \}, \ E = \{ S, A, B, C, D, E \}, \ S_0 = S$$
$$\Gamma = \{ S \rightarrow DA, A \rightarrow b E a B, B \rightarrow a E \mid S, C \rightarrow b D, D \rightarrow da C \mid a, E \rightarrow D \mid C C \}.$$
**PRODUCTION GRAPH AND ADJACENCY MATRIX**

Cover all the of the graph and consider the cycles.

![Production Graph](image)

\[
M(G) = \begin{bmatrix}
0 & 0 & 0 & r & 0 \\
0 & 0 & l & 0 & 0 & b \\
u & 0 & 0 & 0 & 0 & l \\
0 & 0 & 0 & 0 & 0 & l \\
0 & 0 & 0 & 0 & 0 & 0 \\
u & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

**GRAMMATICAL DECOMPOSITION**

- Cover all the of the graph and consider the cycles.

\[
A_1 = \{a, b, D, E\}, \quad E_1 = \{S, A, B\}, \quad S_{01} = S, \\
\Gamma_1 = \{S \rightarrow DA, A \rightarrow b E a B, B \rightarrow a E | S\},
\]

\[
A_2 = \{c, C, D\}, \quad E_2 = \{E\}, \quad S_{02} = E, \\
\Gamma_2 = \{E \rightarrow D | C c\},
\]

\[
A_3 = \{a, b, d\}, \quad E_3 = \{C, D\}, \quad S_{03} \in \{C, D\}, \\
\Gamma_3 = \{C \rightarrow b D, D \rightarrow da C | a\}.
\]
The image contains a slide titled "FINITE STATE AUTOMATA" with several diagrams illustrating finite state automata. Below the diagrams, it transitions into a section titled "DEVELOPMENT AND APPLICATION OF CFG FOR REQUIREMENTS." The text lists the following points:

- Basis of systems engineering lifestyle activities.
- Getting good set of requirements is always a tough job.
- Failure of the project if weak set of requirements.
- A grammar is developed by combining computer science concepts with natural language.
- BADGER—requirements writing tool.
**REQUIREMENT ACTIVITIES**

- Elicit, analyze, document, store, validate the requirements.
- Natural language is misunderstanding and ambiguous.
- Humans have contextual knowledge.
- Requirement sets are very large.
- CFG for comparing the current requirements with previous ones and identify semantic matches.
- Adaptations to different writing styles

**GRAMMAR FOR REQUIREMENTS**

- Requirement document has formal language of CS and natural language.
- Natural language has clauses.
- Independent and subordinate clauses.
- Independent are full with verb and subject.
- Subordinate are temporal, conditional, relative clauses.
CFG FOR REQUIREMENTS

- Two main restrictions for constructing a requirement set.
  - No existence of pronouns
  - Focus on active voice.
- Backus normal form. A set of given derivation rules

\[
\text{symbol} ::= \text{expression}
\]

- Symbol is a non terminal.
- And _expression_ consists of one or more sequence of symbols separated by '|' '
- Non terminals are enclosed by '<>'

REQUIREMENTS GRAMMAR IN BACKUS NAIUR FORM

```
<Requirement> ::= [<TC Clause>“,.”] <Independent Clause> [<Restrictive Relative Clause>] |
  <Independent Clause> [<Restrictive Relative Clause>] [<TC Clause> ]
<Independent Clause> ::= <Subject> <Auxiliary Verb> <Verb> <Noun Phrase>
<Restrictive Relative Clause> ::= [Preposition] <Criterion Indicator> <Value>
  <TC Clause> ::= <TC Indicator> <Noun Phrase> [Verb] <Noun Phrase>
<TC Indicator> ::= Preposition | Condition
  <Value> ::= <Number> <Units>
  [Conjunction] <Noun Phrase>
  [Determinant] <Adjective> <Noun>
  [Conjunction] <Noun Phrase>
  [Conjunction] <Noun Phrase>
<Subject> ::= [Determinant] <Adjective> <Noun>
  <Auxiliary Verb> ::= “shall” [“not”]
  <Criterion Indicator> ::= “no greater than” | “no less than” | “within”
```

- Here TC is temporal condition clause.
EXAMPLE GRAMMAR

- Consider the requirement.
  "organizational message traffic shall be transferred with no greater than 1 in 10\(^3\) BER"
- "Organizational message traffic shall be transferred" is independent clause
  - Subject="organizational message traffic"
  - Verb="be"
  - Auxiliary verb="shall"
  - Target="transferred"
- Restrictive relative clause
  - Proposition="with"
  - Criterion indicator="no greater than"
  - Value’s number="1 in 10\(^3\)"
  - Value’s units="BER"

GRAMMAR APPLICATION FOR REQUIREMENTS ELICITATION

- CFG as basis for BADGER.
- BADGER is “Built in Agent using Deterministic Grammar for Engineering of Requirements”
- TIGER is “Tool to InGest and Elucidate Requirements”
- PETS is “Prototype Educational Tools for Systems and software engineering”
- BADGER ensures adequate information and has pulldown menus.
CFG PARSING FOR HIGH SPEED NETWORK APPLICATIONS

- Processing the data to traverse over internet.
- There must be a rule based pattern matcher capable of detecting strings and/or regular expressions.
- Naïve pattern matchers are susceptible to false positive identification.
- CFG provide high level of expressiveness than strings and regular expressions.
- Goal: design and implement a high speed CFG.

PARSER ARCHITECTURE

- 4 main components.
  - Tokenizer (pattern matching).
  - Parsing structure (from grammar).
  - Error detection unit.
  - Recovery unit.
TOKENIZER

- Regular expression chain architecture
- Pipeline chain grid

GRAMMAR PARSER

- Map grammar rules on to a FPGA (field programmable gate array)
- For production:

<table>
<thead>
<tr>
<th>No.</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A → (A)</td>
</tr>
<tr>
<td>2</td>
<td>A → a</td>
</tr>
</tbody>
</table>

(a) Finite-State Automata
(b) Hardware Logic
STACKS AND COUNTERS

- Sometimes hardware logic can accept invalid strings of type “((a)))” hence we keep track of nesting depth using stack.
- Parse millions of network flows simultaneously.
- Millions of stacks on chip not possible. Hence we use counters.

CONCLUSION
THANKYOU