Topics in IT 1

Parsing and Pattern Recognition

Week 01
Course Introduction

College of Information Science and Engineering
Ritsumeikan University
# course information

| Course title          | Topics in IT 1  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parsing and Pattern Recognition</td>
</tr>
<tr>
<td>Course language</td>
<td>English</td>
</tr>
<tr>
<td>Course web site</td>
<td><a href="http://www.ritsumei.ac.jp/~piumarta/topics1">www.ritsumei.ac.jp/~piumarta/topics1</a></td>
</tr>
<tr>
<td>Instructor’s name</td>
<td>(Prof.</td>
</tr>
<tr>
<td>Instructor’s e-mail</td>
<td><a href="mailto:piumarta@cs.ritsumei.ac.jp">piumarta@cs.ritsumei.ac.jp</a></td>
</tr>
<tr>
<td>Instructor’s office</td>
<td>CC301</td>
</tr>
</tbody>
</table>
| Office hours          | see my web page for availability  
|                       | [www.ritsumei.ac.jp/~piumarta](http://www.ritsumei.ac.jp/~piumarta) |
course assumptions

1: you can speak English (or something approximating it)
   • University policy *requires* this course be taught in English
     – to facilitate access for foreign students
     – to prepare everyone for ‘global community’ participation
   • is there anyone here who cannot speak English?
     – you might want to find a mentor

2: you can understand C programs
   • many examples in this course use C for illustration
   • is there anyone here who cannot read C programs?
     – you might want to find a mentor
course support material

course web site: www.ritsumei.ac.jp/~piumarta/topics1

this web site is essential reading
– copies of lecture slides (a few days before class)
– support material for homework assignments

text books

• are extremely expensive (at least those worth reading)

• therefore there are no required text books
  – all necessary material will be provided (in class or on the course web site)

• there might be suggested (optional) reading from a book
  – a copy of which will always be available for reference in my office, during office hours
individual support

something not clear?
difficulty with English?
problems with practical or reading assignments?

come and ask me questions in CC301
  • exiting Creation Core elevator, 3rd floor: turn right

without appointment:
  • weekly office hours will be posted on my web page soon

other days/times:
  • send e-mail for availability and appointment
course evaluation

mid-term examination

continual weekly evaluation by:

- homework assignments
  - you may be asked to submit your work by e-mail for evaluation
  - you may be asked to demonstrate your work in class
  - you will be tested on your understanding of the homework assignments in quizzes and in the mid-term exam

- quizzes and tests
  - about 10–15 minutes, at the start or end of class
  - one quiz/test almost every week
  - the tests will count towards your final grade!
quizzes and tests

fill in your

- **name** and
- **student ID number**

at the top of the answer sheets

hand in your answer sheet at the end of every quiz/test

- the answer sheet is your attendance record

during tests:

- no talking
- no copying

or you risk obtaining zero credit for the test
suggestions

make *hand-written* lecture notes
  • writing makes learning easier

this is especially important because
  • no laptops
  • no ‘smart’phones

allowed during tests/exams
course philosophy

class attendance

• is expected, every week
• weekly quizzes and homework assignments count towards final score
• no talking or communicating during tests/exams
  – text books or lecture notes may be allowed
• copying/cheating in tests or exams = automatic “F”
  – this is University policy

class format

• interactivity encouraged (except in tests/exams ;-) 
  – tell me if I’m talking too quickly or too slowly
  – tell me if anything is not clear
    ∗ it is highly likely many other people find it unclear
  – interrupt and ask questions

any questions so far?
horribly expensive, but: search for this on Springer’s web site

- from within Rits, it should be a free download
- I will hand out just those sections you need
horribly expensive: ¥10000 at amazon.co.jp

- I have a copy of the 2nd edition
- I will hand out just those sections you need
course contents

1. Overview and introduction
2. String searching and finite choice parsing
3. Classification of grammars
4. Principles of parsing
5. Regular languages and finite state machines
6. Non-deterministic finite-state automata
7. Deterministic finite-state automata
8. Mid-term review and big test
9. Lexical analysis
10. Recursive-descent parsing
11. Context-free parsing
12. Top-down table-driven parsing
13. Nullable rules in LL(1)
14. Grammar transformations
15. Advanced topics

we will try to fit this into 15 weeks
  • slowing down, or speeding up, as necessary

homework:
  • reading
  • report
or:
  • reading
  • presentation
or:
  • programming
  • evaluation
today’s topics

applications of pattern matching and parsing

the parts of language:
  • words and vocabulary: lexemes, lexicons
  • sequences of words: sentences
  • systems of sentences: grammars
  • the structure of grammars

homework assignment
applications of pattern matching

*pattern matching* in computing:

- selecting more than one file for an operation
- searching for files
- finding/counting words in documents
- choosing behaviour based on the structure of data
  - some kinds of functional programming
- etc...

and more generally:

- base sequencing in DNA
- feature extraction (face recognition in images, etc.)
- media repair (audio/visual scratch removal, etc.)
- etc...
applications of parsing

“structuring a linear representation in accordance with a given grammar”

linear representation:

- natural language sentence, computer program
- knitting pattern, musical composition
- sequence of geological strata, actions in ritual behaviour

⇒ any linear sequence in which the preceding elements in some way restrict the next element

grammar:

- well-known (programming languages, many natural languages)
- an object of research (some natural languages, music)
- a notion that is only just taking shape (behavioural patterns)

structure:

- parse tree (derivation)
concrete elements describe what we can write:

- characters
- words
- sentences

character

word

tanaka eats sushi

sentence, phrase (sometimes "word")
anatomy of languages

*abstract* elements describe the structure of sentences:

- categories
- rules (phrase structure)

categories: sentences, noun phrases, verb phrases verbs

rules:

- a **Sentence** is a **Noun Phrase** (subject) and **Verb Phrase**
- a **Verb Phrase** is a **Verb** plus a **Noun Phrase** (object)
anatomy of languages

characters = primitive units of writing or data
words = sequences of (zero or more) characters
categories = families of words playing the same role
phrases = legal sequences of words, categories and phrases
sentences = phrases containing only primitives (characters/words)
rule = permitted way of converting one phrase into another phrase
grammar = set of rules generating all legal sentences of a language

lexical analysis: identifying lexemes in a stream of characters
• the concrete words in a language are its lexemes
• the set of all lexemes in a language are its lexicon
• in programming languages, lexemes are often called tokens

syntactic analysis: finding structure in the sequence of lexemes
• the order in which rules are applied to generate the sentence tells us what role each lexeme plays in its meaning
The rules tell us how to convert one phrase into another. In our English example, the simplest phrase is a S-V-O sentence $S$:

- Made of a noun phrase $NP$ and a verb phrase $VP$

To show that $S$ is made from $NP$ followed by $VP$, we write

$$S \rightarrow NP \ VP$$

Which is read:

$$S \textit{ produces } NP \ VP$$

(Rules are therefore sometimes called *production rules*).

When we see $S$ in any context, we can *rewrite* (replace) it with $NP \ VP$:

- The meaning (and structure) of the phrase does not change.
- But its representation may become a little less abstract.
the structure of grammars

to make a useful grammar, a family of related rules is formed

\[
S \rightarrow NP \ VP \\
NP \rightarrow N \quad (a \ subject \ noun) \\
VP \rightarrow V \ NP \quad (a \ verb \ and \ an \ object \ noun)
\]

to make sentences, we must eventually produce concrete words

\[
N \rightarrow jack \\
N \rightarrow jane \\
V \rightarrow likes \\
V \rightarrow knows
\]

(two productions for the same category do not conflict, they define alternatives when replacing the left-hand side to the right-hand side)

starting with \( S \) we can now produce many concrete sentences, e.g:

jane knows jack
jack likes jane
the structure of grammars

rules can be *recursive*; e.g., adding adjectives before nouns

\[
\begin{align*}
S & \rightarrow NP \text{ } VP \\
NP & \rightarrow N \\
NP & \rightarrow A \text{ } NP \quad \text{(adjectives before a noun)} \\
VP & \rightarrow V \text{ } NP \\
N & \rightarrow jack \quad N \rightarrow jane \\
V & \rightarrow likes \quad V \rightarrow knows \\
A & \rightarrow tall \quad A \rightarrow short \\
A & \rightarrow handsome \quad A \rightarrow pretty
\end{align*}
\]

giving

\[
\text{tall handsome jack likes pretty jane} \\
\text{short jane knows jack} \\
\text{pretty handsome jane knows tall short jack} \\
\text{etc...}
\]

(syntactic correctness does not imply semantic correctness)
grammar specifications

the production rules describe *rewrites*
  - the left-hand side can be rewritten as the right-hand side

various different types of grammar exist
  - with different restrictions on left- and/or right-hand sides

(we will investigate several of these later)

one important type of grammar is restricted by:
  - the left-hand side must be a single category
    (which is how we wrote our trivial model of English)

this type of grammar can describe most programming languages

computer scientists have particular ways of writing these grammars
  - *Backus-Naur Form* (BNF) was the first, and is still popular
our trivial English grammar expressed in BNF:

\[
\begin{align*}
\langle \text{sentence} \rangle & \ ::= \ \langle \text{noun-phrase} \rangle \ \langle \text{verb-phrase} \rangle \\
\langle \text{noun-phrase} \rangle & \ ::= \ \langle \text{noun} \rangle \ | \ \langle \text{adjective} \rangle \ \langle \text{noun-phrase} \rangle \\
\langle \text{verb-phrase} \rangle & \ ::= \ \langle \text{verb} \rangle \ \langle \text{noun-phrase} \rangle \\
\langle \text{noun} \rangle & \ ::= \ "jack" \ | \ "jane" \\
\langle \text{verb} \rangle & \ ::= \ "likes" \ | \ "knows" \\
\langle \text{adjective} \rangle & \ ::= \ "tall" \ | \ "short" \ | \ "pretty" \ | \ "handsome"
\end{align*}
\]

(note the recursion, allowing any number of adjectives before a noun)
extended notations

various pragmatic extensions to BNF (and other grammar notations) are often used

one of them makes repetition explicit

recursions that are used for repetition can then be eliminated

\[
\text{<item>}^? \quad \text{means “zero or one <item>”}
\]

\[
\text{<item>}^* \quad \text{means “zero or more <item>s”}
\]

\[
\text{<item>}^+ \quad \text{means “one or more <item>s”}
\]

⇒ fewer alternatives in productions, and a less confusing grammar

\[
\text{<noun-phrase>} ::= \text{<adjective>} \ \text{<noun-phrase>}
\]

\[
| \quad \text{<noun>}
\]

can be written more concisely as

\[
\text{<noun-phrase>} ::= \text{<adjective>}^* \ \text{<noun>}
\]

(i.e., zero or more \text{<adjective>}s followed by one \text{<noun>})
download the first two handouts

study the following sections of the first handout (about this week’s class):

- Section 2.1.4 (how grammars are constructed)
- Section 2.2.1 (why grammars describe entire languages)
  (the rest is optional, but recommended as background material)

study the following sections of the second handout (in preparation for next week’s class):

- Section 2.3 (the five types of grammar)
glossary

abstract — something that cannot be written down: an idea, or role.

alternative — one of several right-hand sides that can replace a given left-hand side in a production.

Backus-Naur Form — a grammar writing language used to describe computer programming languages.

category — an abstract name representing a sequence of categories or words.

correct — a letter of an alphabet that can be written down.

concrete — something that is written: character, word, or sentence containing only concrete characters/words.

grammar — a collection of related production rules that together define a language.
**language** — the set of all legal sentences that can be produced by a given grammar.

**lexeme** — the smallest unit of concrete information dealt with by a grammar. Grammars that describe complex sentences might have single words as their lexemes. Grammars that describe how words are constructed from characters would have characters as their lexemes.

**lexical analysis** — the process of turning a sequence of characters into a sequence of words, for a grammar in which lexemes are words.

**lexicon** — the collection of lexemes (words) that can appear in the sentences of a particular language.

**linear representation** — a sentence described as its sequence of lexemes.

**parsing** — the process of discovering the role/category of each lexeme and group of lexemes in a given sentence.
pattern matching — the process of identifying a sequence of items (characters or words) within a larger sequence of items.

phrase — a sequence of words (most often in a natural language) that together form a useful category for reasoning about sentences in which they appear. Most English sentence can be broken into a ‘subject’ noun phrase and a verb phrase (which contains an ‘object’ noun phrase).

production rule — a description of how to rewrite a sequence of categories and/or words with an equivalent sequence that does not change the meaning of the phrase.

production — one rewrite, according to a single production rule, that transforms the rule’s left-hand side into its right-hand side within a phrase.

recursive — a rule (or any function in general) that invokes itself, directly or indirectly.
**rewrite** — replacing a sequence of category names and/or words with another sequence.

**rule** — see *production rule*.

**sentence** — a sequence of words belonging to some language.

**syntactic analysis** — the process of converting a sequence of words into a structure representing the meaning of those words according to the grammar that generated them. The structure is often a tree reflecting the productions that are needed to generate the sentence, and the order in which they must be applied.

**token** — the term in computer science often used to mean the same thing as *word*. When the word is a category name, the token usually also includes some data describing which concrete value within the category is being represented. For example, a token might represent the category *Noun* with the specific value “*elephant*”, or the category *Number* with the specific value 21, etc.
**word** — a single, indivisible unit of lexical information within a sentence.