Computability and Formal Languages

Chang-Gun Lee (cglee@snu.ac.kr)

Assistant Professor

The School of Computer Science and Engineering

Seoul National University

Russell's Paradox

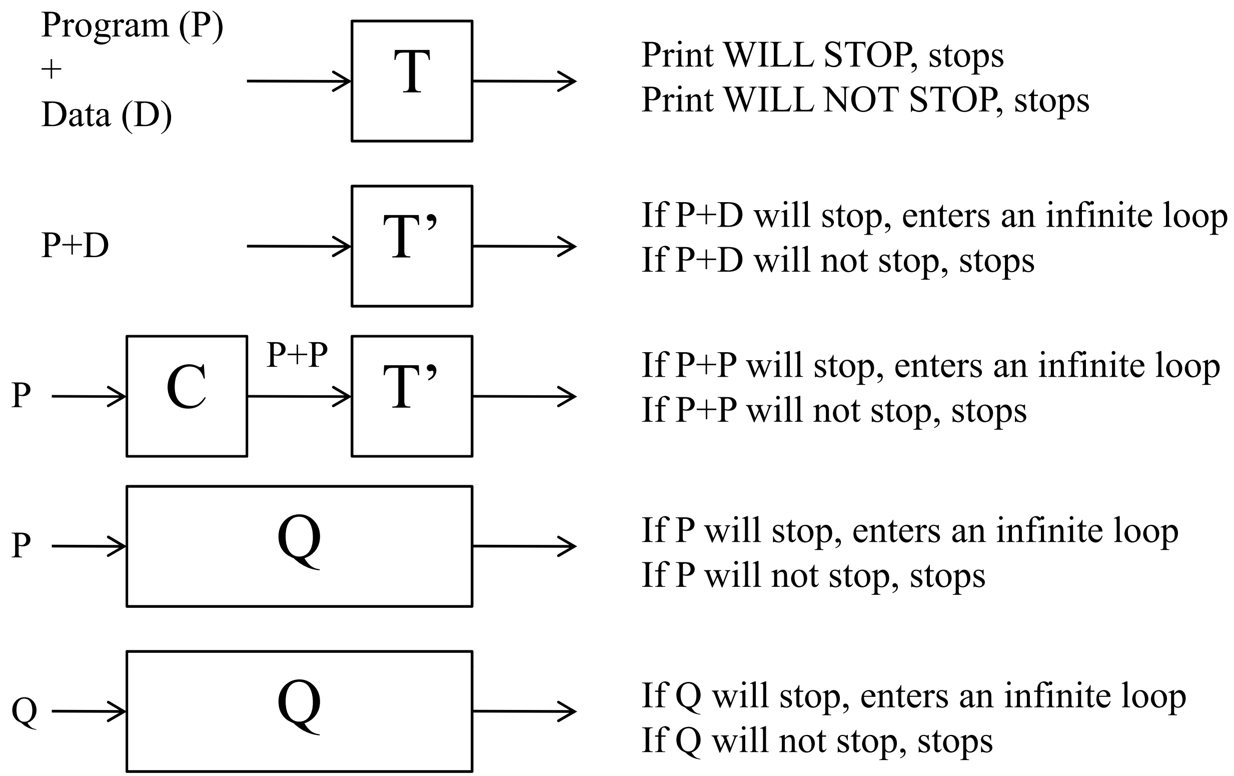
- Is it always possible to clearly specify the characteristic of elements in a set (so that a computer can enumerate them)?
- P = {x|x is a high school student in Illinois}
- Q={x|x is a perfect square}
- R={x|{a,b} $\subseteq$ x}
- S={x|x $\notin$ x}
- It is not always the case that we can precisely specify the elements of a set by specifying the properties of the elements in the set. → Russell's paradox.

Russell’s Paradox (Examples)

- There is a barber in a small village. He will shave everybody who does not shave himself.
- There were two craftsmen, Bellini and Cellini, from Florence. Whatever Bellini made, he always put a true inscription on it. On the other hand, whatever Cellini made, he always put a false inscription on it. If they were only craftsmen around, what would you say if it was reported that the following sign was discovered? “This sign was made by Cellini”

Noncomputability

- We want to show that there are tasks no computer can perform
- How?
- Can we write a computer program that checks if a program ever stops for a given program and data?



Languages in Math

- Let A={a, b, c, d, ..., x, y, z} denote the 26-letter English alphabet.
  - A n-letter word is a list (ordered set) of n letters.
  - In the context of languages, we often use the terms sequences, strings, or sentences (of letters) interchangeably with the term ordered n-tuple (list of size n).
  - A^n or $\{a,b,c,d, ..., x,y,z\}^n$: set of all sequences of n letters from A
  - A^* or $\{a,b,c,d, ..., x,y,z\}^*$: set of all sequences of letters from A
    - For example, the set of all the names in a telephone directory is a subset of A*
- Let B={a, b, ..., y, z, A, B, ..., Y, Z, ., ,, :, ;, !, ?, _}
  - A sentence in the English language is a sequence (or list) in B^*
  - Where_is_John?
- Let C = {A, B, ..., Y, Z, 0, 1, 2, ..., 8, 9, +, -, *, /, ;, ., =}
  - A statement in a programming language is a sequence in C^*

Formal Definition of Languages

- Definition: Let A be a finite set which is the alphabet of the language. A language (over the alphabet A) is a subset of the set A^*.
- For example, let A={a,b,c}. The following sets are all languages over the alphabet A.
  - L_1 = {a, aa, ab, ac, abc, cab}
  - L_2 = {aba, aabaa}
  - L_3 = { }
  - $L_4 = \{a^i c b^i \mid i \geq 1\}$
- Since languages are defined as sets of strings (or lists or sequences), all set operations can be applied to languages
  - If L_1 is the English language and L_2 is the French language, $L_1 \cup L_2$ will be the set of all sentences someone who speaks both English and French can recognize.
  - As other examples, note that

$$\{a^i b^j \mid i > j \geq 1\} \cup \{a^i b^j \mid 1 \leq i < j\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

$$\{a^i b^i c^j \mid i, j \geq 1\} \cap \{a^i b^j c^j \mid i, j \geq 1\} = \{a^i b^i c^i \mid i \geq 1\}$$

$$\{a^i b^j \mid i \geq j \geq 1\} \oplus \{a^i b^j \mid 1 \leq i \leq j\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

$$\{a^i b^j \mid i, j \geq 1\} - \{a^i b^i \mid i \geq 1\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

Formal Definition of Languages

- Definition: Let A be a finite set which is the alphabet of the language. A language (over the alphabet A) is a subset of the set A^*.
- For example, let A={a,b,c}. The following sets are all languages over the alphabet A.
  - L_1 = {a, aa, ab, ac, abc, cab}
  - L_2 = {aba, aabaa}
  - L_3 = { }
  - $L_4 = \{a^i c b^i \mid i \geq 1\}$
- Since languages are defined as sets of strings (or lists or sequences), all set operations can be applied to languages
  - If L_1 is the English language and L_2 is the French language, $L_1 \cup L_2$ will be the set of all sentences someone who speaks both English and French can recognize.
  - As other examples, note that

$$\{a^i b^j \mid i > j \geq 1\} \cup \{a^i b^j \mid 1 \leq i < j\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

$$\{a^i b^i c^j \mid i, j \geq 1\} \cap \{a^i b^j c^j \mid i, j \geq 1\} = \{a^i b^i c^i \mid i \geq 1\}$$

$$\{a^i b^j \mid i \geq j \geq 1\} \oplus \{a^i b^j \mid 1 \leq i \leq j\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

$$\{a^i b^j \mid i, j \geq 1\} - \{a^i b^i \mid i \geq 1\} = \{a^i b^j \mid i \neq j, i, j \geq 1\}$$

How to specify a language?

- A language is a set of strings, and hence two ways to specify the set
  - exhaustive listing of all strings
  - describing the properties that characterize all strings
- For any non-trivial language, the above two ways do not work!
- Furthermore, for may applications, we are interested mostly in
  - Given the specification of a language, automatically generate one or more strings in the language
  - Given the specification of a language, determine whether a given string is in the language
- Any way to describe a language that will facilitate us in solving the above problems?
- Let’s try to specify a language by a grammar!

How to specify a language?

- A language is a set of strings, and hence two ways to specify the set
  - exhaustive listing of all strings
  - describing the properties that characterize all strings
- For any non-trivial language, the above two ways do not work!
- Furthermore, for may applications, we are interested mostly in
  - Given the specification of a language, automatically generate one or more strings in the language
  - Given the specification of a language, determine whether a given string is in the language
- Any way to describe a language that will facilitate us in solving the above problems?
- Let's try to specify a language by a grammar! → A class of grammars known as "phrase structure grammars"

Grammar in English

1. A **sentence** is a **noun-phrase** followed by a **transitive-verb-phrase** and another **noun-phrase.**
2. A **sentence** is a **noun-phrase** followed by an **intransitive-verb-phrase**.
3. A **noun-phrase** is an **article** followed by a **noun**.
4. A **noun-phrase** is a **noun**.
5. A **transitive-verb-phrase** is a **transitive-verb**.
6. An **intransitive-verb-phrase** is an **intransitive-verb** followed by an **adverb**.
7. An **intransitive-verb-phrase** is an **intransitive-verb**.
8. An **article** is *a*.
9. An **article** is *the*.
10. A **noun** is *dog*.
11. A **noun** is *cat*.
12. A **transitive-verb** is *chases*.
13. A **transitive-verb** is *meets*.
14. An **intransitive-verb** is *runs*.
15. An **adverb** is *slowly*.
16. An **adverb** is *rapidly*.

Grammar in English

sentence → noun-phrase transitive-verb-phrase noun-phrase
sentence → noun-phrase intransitive-verb-phrase
noun-phrase → article noun
noun-phrase → noun
transitive-verb-phrase → transitive-verb
intransitive-verb-phrase → intransitive-verb adverb.
intransitive-verb-phrase → intransitive-verb.
article → *a*
article → *the*
noun → *dog*
noun → *cat*
transitive-verb → *chases*
transitive-verb → *meets*
intransitive-verb → *runs*
adverb → *slowly*
adverb → *rapidly*

the dog meets a cat
dog chases cat
the cat runs slowly

Phrase Structure Grammar

- It consists of four items
  1. A set of <u>terminals</u> T (like *a*, *the*, *dog*, *cat*, *slowly*, etc.)
  2. A set of <u>nonterminals</u> N (like **sentence**, **noun-phrase**, **noun**, **article**, etc.)
  3. A set of <u>productions</u> P (A production is a form of $\alpha \rightarrow \beta$)
  4. Among all the nonterminals in N, there is a special nonterminal that is referred to as the <u>starting symbol</u> (like **sentence**)

Process of generating a sentence

- Once we are given a grammar, we can generate the sentences in the language as follows:
  - Begin with the starting symbol as the current string (of terminals and non-terminals)
  - If any portion of the current string matches the left-hand side of a production, replace that portion by the right-hand side of the production
  - Any string of "only" terminals obtained by repeating step 2 is a sentence in the language.

"a dog runs rapidly"

sentence ➔ noun-phrase intransitive-verb-phrase

➔**noun-phrase intransitive-verb adverb**

➔**noun-phrase intransitive-verb** *rapidly*

➔**noun-phrase** *runs rapidly*

➔**article noun** *runs rapidly*

➔**article** *dog runs rapidly*

➔*a dog runs rapidly*

Example (1)

- We want to construct a grammar for the language
  - L = {aaaa, aabb, bbaa, bbbb}

T={a,b}, N={S}
S→aaaa
S→aabb
S→bbaa
S→bbbb

T={a,b}, N={S,A}
S→AA
A→aa
A→bb

Example (2)

- We want to construct a grammar for the language
  - $L = \{a^i b^{2i} \mid i \geq 1\}$

$T=\{a,b\}$, $N=\{S\}$

$S \rightarrow aSbb$

$S \rightarrow abb$

Example (3)

- We want to construct a grammar for the language
  - $L = \{x|x \in \{a,b\}^*$, the number of a's in x is a multiple of 3$\}$

T={a,b}, N={S,A,B}
$S \rightarrow bS$
$S \rightarrow b$
$S \rightarrow aA$
$A \rightarrow bA$
$A \rightarrow aB$
$B \rightarrow bB$
$B \rightarrow aS$
$B \rightarrow a$

Example (4)

- Suppose we are given a grammar in which T={a,b} and N={S,A,B}, with S being the starting symbol. Let the set of productions be

$S \rightarrow aB$
$S \rightarrow bA$
$A \rightarrow a$
$A \rightarrow aS$
$A \rightarrow bAA$
$B \rightarrow b$
$B \rightarrow bS$
$B \rightarrow aBB$

- What is this language?
  – all strings of a's and b's in which the number of a's equals the number of b's

Example (5)

- Let T={A,B,C,D,+,*,(,),=} and N={**asgn_stat**, **exp**, **term**, **factor**, **id**}, with **asgn_stat** being the starting symbol.

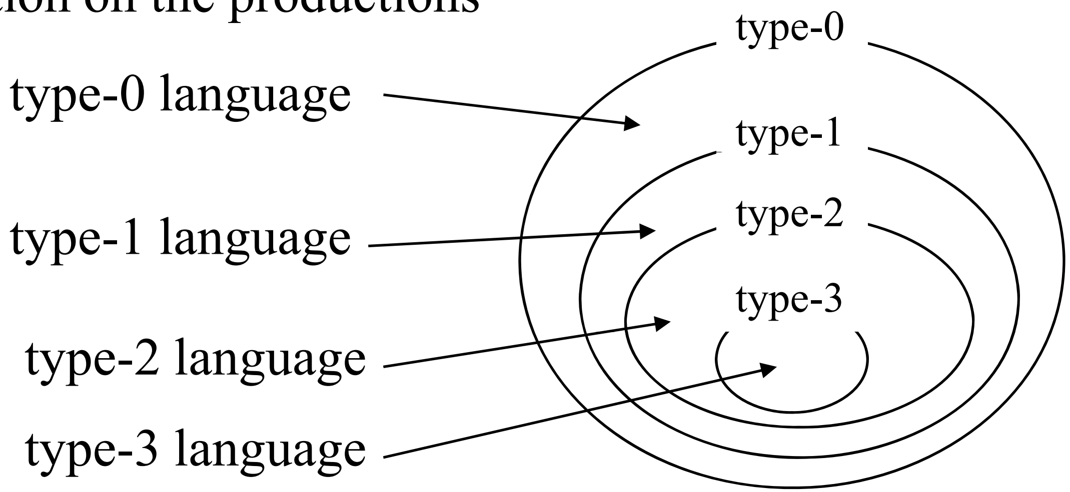
asgn_stat→id=exp
exp→exp+term
exp→term
term→term*factor
term→factor
factor→(**exp**)
factor→id
id→A
id→B
id→C
id→D

C=A+D*(D+B)

asgn_stat→id=exp→id=exp+term→
id=exp+term*factor→id=exp+term*(exp)→
id=exp+term*(exp+term)→
id=exp+term*(exp+factor)→
id=exp+term*(exp+id)→
id=exp+term*(exp+B)→
id=exp+term*(term+B)→
id=exp+term*(factor+B)→
id=exp+term*(id+B)→
id=exp+term*(D+B)→
id=exp+factor*(D+B)→
id=exp+id*(D+B)→
id=exp+D*(D+B)→
id=term+D*(D+B)→
id=factor+D*(D+B)→
id=id+D*(D+B)→
id=A+D*(D+B)→ C=A+D*(D+B)

Types of Grammars and Languages

- *A* and *B* denote arbitrary nonterminals, *a* and *b* denote arbitrary terminals, and α and β denote arbitrary strings of terminals and nonterminals.
- Type-3 grammar
  - $A \rightarrow a$
  - $A \rightarrow aB$
- Type-2 grammar
  - $A \rightarrow \alpha$
- Type-1 grammar
  - $\alpha \rightarrow \beta$ (length of β is larger than or equal to the length of α)
- Type-0 grammar
  - no restriction on the productions



Some questions?

- Are there languages that are not type-0 language?
  - affirmative
- How about all the programming languages?
  - all of them are (almost) type-2 languages
- This is how a compiler for a programming language works
  - to understand and analyze a sentence.