CSCI 340: Computational Models

Recursive Definitions

Chapter 3

Department of Computer Science

A New Method for Defining Languages

Recursive Definitions allow us to define sets in a unique way

1. Specify some basic objects in the set.
2. Give rules for constructing additional objects in the set.
3. Declare that no objects except those constructed are allowed

Example

Standard Definition:

EVEN is the set of all positive whole numbers divisible by 2

Alternative Definition:

EVEN is the set of all $2n$ where $n = 1\ 2\ 3\ 4\ \ldots$

Recursive Definition:

1. 2 is in EVEN
2. if x is in EVEN, then so is $x + 2$.
3. The *only* elements in EVEN are those produced by (1) and (2)

Fun with EVEN

Question

Why would we ever want to use the recursive definition for EVEN?

Example

Prove 14 is in set EVEN

Proof by Standard Definition.

Divide 14 by 2 and find there is no remainder □

Proof by Alternative Definition.

Somehow come up with the number 7
Since (2)(7) = 14, 14 is in EVEN □

Fun with EVEN

Proof by Recursive Definition.

By *Rule 1*, 2 is in EVEN.
By *Rule 2*, we know $2 + 2 = 4$ is also in EVEN.
By *Rule 2*, we know $4 + 2 = 6$ is also in EVEN.
By *Rule 2*, we know $6 + 2 = 8$ is also in EVEN.
By *Rule 2*, we know $8 + 2 = 10$ is also in EVEN.
By *Rule 2*, we know $10 + 2 = 12$ is also in EVEN.
By *Rule 2*, we know $12 + 2 = 14$ is also in EVEN. □

Aside: This is completely disgusting

Can we come up with a better recursive definition?

Fun with EVEN

A Better Recursive Definition for EVEN

1. 2 is in EVEN
2. If x and y are both in EVEN, then so is $x + y$
3. The *only* elements in EVEN are those produced by (1) and (2)

Proving 14 is in EVEN by our new Definition.

By *Rule 1*, 2 is in EVEN.
By *Rule 2*, $x = 2, y = 2 \longrightarrow 4$ is also in EVEN.
By *Rule 2*, $x = 2, y = 4 \longrightarrow 6$ is also in EVEN.
By *Rule 2*, $x = 4, y = 4 \longrightarrow 8$ is also in EVEN.
By *Rule 2*, $x = 6, y = 8 \longrightarrow 14$ is also in EVEN. □

Why is this definition better?

INTEGERS

Example

1. 1 is in INTEGERS
2. If x is in INTEGERS, then so is $x + 1$.

Note: we will omit rule 3 from now on

If we wanted the set INTEGERS to include positive and negative integers, we need to change our definition:

Example

1. 1 is in INTEGERS
2. If x and y are both in INTEGERS, then so are $x + y$ and $x - y$.

POSITIVE

Example

1. x is in POSITIVE
2. If x and y are both in POSITIVE, then so are $x + y$ and xy.

Problem: there no base for x

Example (An Attempted Variant)

1. x is in INTEGERS, "." is a decimal point, and y is any finite string of digits, even one starting with 0's, then $x.y$ is in POSITIVE

Problem 1: doesn't generate all real numbers e.g. π.
Problem 2: definition is not recursive

Example (A Better Definition)

1. 1 is in POSITIVE

POLYNOMIAL

A *polynomial* is a finite sum of terms, each of which is of the form: a real number times a power of x (that may be $x^0 = 1$)

Example

1. Any number is in POLYNOMIAL
2. The variable x is in POLYNOMIAL
3. If p and q are in POLYNOMIAL, then so are $p + q$, $p - q$,(p), pq

Problem

Show $3x^2 + 7x - 9$ *is in POLYNOMIAL*

POLYNOMIAL

Problem

Show $3x^2 + 7x - 9$ *is in POLYNOMIAL*

Proof.

By *Rule 1:* 3 is in POLYNOMIAL.
By *Rule 2:* x is in POLYNOMIAL.
By *Rule 3:* $(3)(x)$ is in POLYNOMIAL; call it $3x$.
By *Rule 3:* $(3x)(x)$ is in POLYNOMIAL; call it $3x^2$.
By *Rule 1:* 7 is in POLYNOMIAL; call it $3x^2$.
By *Rule 3:* $(7)(x)$ is in POLYNOMIAL; call it $7x$.
By *Rule 3:* $3x^2 + 7x$ is in POLYNOMIAL.
By *Rule 1:* -9 is in POLYNOMIAL.
By *Rule 3:* $3x^2 + 7x - 9$ is in POLYNOMIAL. □

Advantages and Disadvantages of POLYNOMIAL

Advantages

- sums and products of polynomials are obviously polynomials
- if we have a proof for differentiable, we can show all polynomials are differentiable
- **AND** we don't need to give the best algorithm for it

Disadvantages

- Tedious building blocks
-

Reminder: this is computer theory – we are interested in proving that tasks are possible, not necessarily knowing the best algorithm in *how* to do it

Examples

Example (x^+)

1. x is in L_1
2. If w is any word in L_1, then xw is also in L_1

Example (x^*)

1. λ is in L_2
2. If w is any word in L_2, then xw is also in L_1

Example (x^{odd})

1. λ is in L_3
2. If w is any word in L_3, then xxw is also in L_3

Examples

Example (INTEGER)

1. 1 2 3 4 5 6 7 8 9 are in INTEGERS
2. If w is any word in INTEGERS, then
   $w0$ $w1$ $w2$ $w3$ $w4$ $w5$ $w6$ $w7$ $w8$ $w9$ are also in INTEGERS

Example (Kleene Closure)

1. If S is a language, then all the words of S are in S^*
2. λ is in S^*
3. If x and y are in S^*, then so is the concatenation xy

Note: this definition of Kleene Closure is *easier* to understand.

Arithmetic Expressions

What is a valid arithmetic expression?

Alphabet

$$\Sigma = \{\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -\ *\ /\ (\ )\ \}$$

Invalid Strings?

$(3+5)+6)$ $\quad 2(/8+9)$ $\quad (3+(4-)8)$ $\quad 2)-(4$

Problem

What makes a valid string?

Solution

Recursive Definition...

Recursive Definition for Arithmetic Expressions

1. Any number (positive, negative, or zero) is in AE
2. If x is in AE, then so are:
   1. (x)
   2. $-x$ (provided x does not already start with a minus sign)
3. If x and y are in AE, then so are:
   1. $x + y$ (if the first symbol in y is not + or −)
   2. $x - y$ (if the first symbol in y is not + or −)
   3. $x * y$
   4. x/y
   5. $x * *y$ (notation for exponentiation)

There may be strings which we may not know the *meaning* of, but definitely argue that strings are a part of AE.

For example: 8/4/2 could mean 8/(4/2) or (8/4)/2 depending on order of operations

Examples with Arithmetic Expressions

Theorem

An arithmetic expression cannot contain the character $

Proof.

$ is not part of any number, so it cannot be induced (by *Rule 1*)

If a string, x, doesn't contain $, then neither can (x) or $-x$ (by *Rule 2*)

If neither x nor y contains $, then neither do any induced (by *Rule 3*)

The character $ can never be inserted into AE □

Examples with Arithmetic Expressions

Theorem

No AE can begin or end with symbol /

Proof.

No number begins or ends with / so it cannot occur (by *Rule 1*).

Any AE formed by *Rule 2* must begin and end with parentheses or begin with a minus sign.

If x does not already begin with / and y does not end with /, then any AE formed by any clause in *Rule 3* will not begin or end with a /.

These rules prohibit an expression beginning or ending with /. □

Examples with Arithmetic Expressions

Theorem

No AE can contain the substring //

Proof (by contradiction).

- Let us suppose there were some AE that contained the substring //. Let a shortest of these be a string w.
- w must be formed by some sequence of applying Rules 1, 2, and 3. The last rule used producing w must have been *Rule 3iv*.
- Splitting w from *Rule 3iv* would result in w_1/w_2 – meaning that w_1 would need to end with / or w_2 would need to start with /.
- Since there is no rule that possibly yields a trailing / or leading /, then w_1 or w_2 must contain //.
- Since we claimed w was the shortest AE that contained the substring //, this is not feasible.
- Therefore, nothing in the set AE can have the substring //. □

Well-Formed Formulas

$$\Sigma = \{\neg \;\; \rightarrow \;\; (\;)\; a\; b\; c\; d\; \ldots\}$$

1. Any single Latin letter is a WFF.
2. If p is a WFF, the so are (p) and $\neg p$.
3. If p and q are WFFs, then so is $p \rightarrow q$

- $p \rightarrow$
- $\rightarrow p$
- $(p \rightarrow$
- $p)$
- $p) \rightarrow p($
- $p \rightarrow ((p \rightarrow p) \rightarrow q)$
- $\neg p \rightarrow p$

Homework 1b

5. Using any recursive definition of the set EVEN, show that all the numbers in it end in the digits 0, 2, 4, 6, or 8
6. Show that if n is less than 31, then x^n can be shown to be in POLYNOMIAL in fewer than eight steps
7. Give a recursive definition for the language PALINDROME. Make sure it works for even and odd length strings. $\Sigma = \{a\ b\}$
8. Give recursive definitions for the following languages. $\Sigma = \{a\ b\}$
    - i. the language EVENSTRING of all words of even length
    - ii. the language ODDSTRING of all words of odd length
    - iii. the language AA of all words containing substring *aa*
    - iv. the language NOTAA of all words **not** containing substring *aa*