Mathematical String Notation

String Theory

- A mathematical model that we will use often is that of ***mathematical strings***
- A string can be thought of as a series of zero of more ***entries*** of *any* other mathematical type, say, T
  - T is called the ***entry type***
  - We will call this math type ***string of*** T

Math Notation for Strings

- Two important features of strings:
  - There may be ***duplicate*** entries
  - The ***order*** of the entries is important

The Empty String

- The ***empty string***, a string with no entries at all, is denoted by <>

Denoting a Specific String

- A particular string can be described by listing its entries between < and > separated by commas
- Examples:

  <1,2,3,2>

  <'G','o'>

  <>

Denoting a Specific String

- A particular st
  listing its entri
  separated by com
- Examples:

  `<1,2,3,2>`

  `<'G','o'>`

  `<>`

A ***string of integer*** value whose entries are the ***integer*** values *1*, *2*, *3*, and *2*.

Denoting a Specific String

- A particular st
  listing its entri
  separated by com
- Examples:

  `<1,2,3,2>`

  `<'G','o'>`

  `<>`

A ***string of character*** value whose entries are the ***character*** values `'G'` and `'o'`.

Denoting a Specific String

- A particular str
  listing its entri
  separated by com
- Examples:

  <1,2,3,2>

  <'G' 'o'>

  <>

Notation for an empty string

Concatenation

- `s * t` denotes he ***concatenation*** of strings `s` and `t`

- Examples:

```
<1,2> * <3,2> = <1,2,3,2>
<'G','o'> * <> = <'G','o'>
<> * <5,2,13> = <5,2,13>
<> * <> = <>
```

Substring

- *s* ***is substring of*** *t* iff the entries of *s* appear somewhere consecutively in *t*

Substring Examples

Given the following 3 string variables:

t = <17,3,5,2,1>

s1 = <3,5>

s2 = <5,3>

Examples:

- *s1 is a substring of t*
- *s2 is not a substring of t*

Prefix

- *s* ***is prefix of*** *t* iff the entries of *s* appear consecutively at the beginning of *t*

Prefix Examples

Given the following 3 string variables:

t = <17,3,5,2,1>

s1 = <17,3,5>

s2 = <5,2,1>

Examples:

- *s1 is a prefix of t*
- *s2 is not a prefix of t*

Suffix

- *s* ***is suffix of*** *t* iff the entries of *s* appear consecutively at the end of *t*

Suffix Examples

Given the following 3 string variables:

t = <17,3,5,2,1>

s1 = <17,3,5>

s2 = <5,2,1>

Examples:

- *s1 is not a suffix of t*
- *s2 is a suffix of t*

Length

- $|s|$ denotes he ***length*** of a string s
- That is, the number of entries in s

- Examples:

  $|\langle \text{“C343”}, \text{“C202”}, \text{“C251”}, \text{“C455”} \rangle| = 4$

  $|\langle \text{'G'}, \text{'o'} \rangle| = 2$

  $|\langle\rangle| = 0$

Concise Notation for Substrings

- `s[i, j)` denotes the substring of `s` starting at ***position*** `i` (inclusive) and ending at ***position*** `j` (exclusive)

- Where ***position*** `k` of an entry in a string is a number satisfying `0 <= k < |s|`

Concise Notation for Substrings

- `s[i, j)`

This notation is well-defined whenever

`0 <= i <= j <= |s|`;

for all other cases, the designated substring is defined to be: `<>`

Examples with *s[i, j)*

Let *s = <0,5,10,15,20,25>*

s[1,3) = *<5,10>*

s[0,|s|) = *<0,5,10,15,20,25>*

s[1,|s|-1) = *<5,10,15,20>*

s[1,1) = *<>*

*s[1,2) * s[3,4)* = *<5,15>*

s[1,0) = *<>*

Reverse

- ***rev***`(s)` denotes the ***reverse*** of a string `s`.
- That is, the string with the same entries as `s` but in the opposite order
- Examples:

```
rev(<1,2,3,4>) = <4,3,2,1>
rev(<'G','o'>) = <'o','G'>
rev(<>) = <>
```

Permutations

- ***perms***`(s1, s2)` denotes the question whether strings `s1` and `s2` are ***permutations***
- That is, whether they are simply reorderings of one another
- Examples:

```
perms(<1,2,3,4>, <3,1,4,2>) = true
not perms(<2,2,1>, <2,1>) = true
perms(<>, <>) = true
```

Occurrence Count

- ***count***`(s, x)` denotes the ***occurrence count*** of an entry `x` in a string `s`
- That is, the number of times `x` appears as an entry in `s`
- Examples:

```
count(<2,2,2,1>, 2) = 3
count(<2,2,2,1>, 4) = 0
count(<'G','o'>, 'G') = 1
count(<>, 27) = 0
```

Credits

- These slides were adapted from slides obtained from Dr. Bruce W. Weide and Dr. Paolo Bucci.

- Drs. Weide & Bucci are members of the Resolve/Reusable Software Research Group (RSRG) which is part of the Software Engineering Group in the Department of Computer Science and Engineering at The Ohio State University.