Mathematical Structures in Programming

Zhenjiang Hu

The Graduate University for Advanced Studies

April 11, 2011

Course Summary

This course

- discusses the mathematical structures in programs, and

Course Summary

This course

- discusses the mathematical structures in programs, and
- explains how mathematical reasoning plays an important role in designing efficient and correct algorithms (programs).

Course Summary

This course

- discusses the mathematical structures in programs, and
- explains how mathematical reasoning plays an important role in designing efficient and correct algorithms (programs).

$\Uparrow$

A new programming style: calculational programming

Course Summary

This course

- discusses the mathematical structures in programs, and
- explains how mathematical reasoning plays an important role in designing efficient and correct algorithms (programs).

$\Uparrow$

A new programming style: calculational programming

Course page:

http://research.nii.ac.jp/~hu/pub/teach/msp11_nii/

Calculation is widely used in solving our daily problems, but its importance in programming has not been fully recognized.

An Arithmetic Problem: Tsuru-Kame-Zan

Crane and Tortoise Calculation Problem

Calculate how many tsuru (crane which has 2 legs) or kame (tortoise which has 4 legs) there are, if we know that there are 12 legs and 5 heads.

An Arithmetic Problem: Tsuru-Kame-Zan

Crane and Tortoise Calculation Problem

Calculate how many tsuru (crane which has 2 legs) or kame (tortoise which has 4 legs) there are, if we know that there are 12 legs and 5 heads.

Children in Kindergarden: solving problems by enumeration

- Crane 0, Tortoise 5: No
- Crane 1, Tortoise 4: No
- Crane 2, Tortoise 3: No
- Crane 3, Tortoise 2: No
- Crane 4, Tortoise 1: Yes
- Crane 5, Tortoise 0: No

An Arithmetic Problem: Tsuru-Kame-Zan

Crane and Tortoise Calculation Problem

Calculate how many tsuru (crane which has 2 legs) or kame (tortoise which has 4 legs) there are, if we know that there are 12 legs and 5 heads.

Students in Primary School: solving problems using rules

$$\begin{aligned} \mathit{tortoise} &= (\mathit{numberOfLegs} - \mathit{numberOfHeads} \times 2)/2 \\ \mathit{crane} &= \mathit{numberOfHeads} - \mathit{tortoise} \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} \mathit{tortoise} &= (12 - 5 \times 2)/2 = 1 \\ \mathit{crane} &= 5 - 1 = 4 \end{aligned}$$

An Arithmetic Problem: Tsuru-Kame-Zan

Crane and Tortoise Calculation Problem

Calculate how many tsuru (crane which has 2 legs) or kame (tortoise which has 4 legs) there are, if we know that there are 12 legs and 5 heads.

Students in Middle School: solving problem using equation theories

$$\begin{aligned} x + y &= numberOfHeads \\ x \times 4 + y \times 2 &= numberOfLegs \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} x &= 1 \\ y &= 4 \end{aligned}$$

An Arithmetic Problem: Tsuru-Kame-Zan

Crane and Tortoise Calculation Problem

Calculate how many tsuru (crane which has 2 legs) or kame (tortoise which has 4 legs) there are, if we know that there are 12 legs and 5 heads.

Laws and theories are useful for solving problems easily and systematically.

- Problems are declaratively specified.
- Implementation is hidden.

This course studies laws and theories in programming for obtaining both correct and efficient algorithms (programs).

A Programming Problem: Maximum Segment Sum

- Given a list of numbers, find the maximum sum of a *consecutive* segment.
  - $[-1, 3, 3, -4, -1, 4, 2, -1] \quad \Longrightarrow \quad 7$
  - $[-1, 3, 1, -4, -1, 4, 2, -1] \quad \Longrightarrow \quad 6$
  - $[-1, 3, 1, -4, -1, 1, 2, -1] \quad \Longrightarrow \quad 4$

A Programming Problem: Maximum Segment Sum

- Given a list of numbers, find the maximum sum of a *consecutive* segment.
  - $[-1, 3, 3, -4, -1, 4, 2, -1] \quad \Longrightarrow \quad 7$
  - $[-1, 3, 1, -4, -1, 4, 2, -1] \quad \Longrightarrow \quad 6$
  - $[-1, 3, 1, -4, -1, 1, 2, -1] \quad \Longrightarrow \quad 4$
- Can you design a correct linear time algorithm?

A Simple Solution

1. Enumerating all segments (*segs*);

A Simple Solution

1. Enumerating all segments (*segs*);
2. Computing sum for each segment(*sums*);

A Simple Solution

1. Enumerating all segments (*segs*);
2. Computing sum for each segment(*sums*);
3. Calculating the maximum of all the sums (*max*).

A Simple Solution

1. Enumerating all segments (*segs*);
2. Computing sum for each segment(*sums*);
3. Calculating the maximum of all the sums (*max*).

A Simple Solution

1. Enumerating all segments (*segs*);
2. Computing sum for each segment(*sums*);
3. Calculating the maximum of all the sums (*max*).

Exercise

How many segments does a list of length n have?

A Simple Solution

1. Enumerating all segments (*segs*);
2. Computing sum for each segment(*sums*);
3. Calculating the maximum of all the sums (*max*).

Exercise

How many segments does a list of length n have?

Exercise

What is the time complexity of the above simple solution?

There indeed exists a clever solution!

```
mss=0; s=0;
 for(i=0;i<n;i++){
     s += x[i];
     if(s<0) s=0;
     if(mss<s) mss= s;
 }
```

$x[i]$	-1	3	1	-4	-1	1	2	-1
s	0	3	4	0	0	1	3	2
mss	0	3	4	4	4	4	4	4

There is a big gap between the simple and clever solutions!

There is a big gap between the simple and clever solutions!

- Can we calculate the clever solution from the simple solution?

There is a big gap between the simple and clever solutions!

- Can we calculate the clever solution from the simple solution?
- What laws and theorems are necessary to do so?

There is a big gap between the simple and clever solutions!

- Can we calculate the clever solution from the simple solution?
- What laws and theorems are necessary to do so?
- How to apply the laws and theorems to do so?

There is a big gap between the simple and clever solutions!

- Can we calculate the clever solution from the simple solution?
- What laws and theorems are necessary to do so?
- How to apply the laws and theorems to do so?

$$\Downarrow$$

There problems will be addressed in this course.

Aims

1. Fully understand why algorithm design and programming can be viewed as a mathematical activity.

Aims

1. Fully understand why algorithm design and programming can be viewed as a mathematical activity.
2. Can apply mathematical reasoning to solve practical programming problems.

Course Contents

1. Algorithm Description (in Haskell)

Course Contents

1. Algorithm Description (in Haskell)
2. Specification (compositions of specific functional units)

Course Contents

1. Algorithm Description (in Haskell)
2. Specification (compositions of specific functional units)
3. Laws and Theories for Algorithm Calculation (in Coq)

Course Contents

1. Algorithm Description (in Haskell)
2. Specification (compositions of specific functional units)
3. Laws and Theories for Algorithm Calculation (in Coq)
4. Applications (dynamic programming, parallelization, inversion)

Course Contents

1. Algorithm Description (in Haskell)
2. Specification (compositions of specific functional units)
3. Laws and Theories for Algorithm Calculation (in Coq)
4. Applications (dynamic programming, parallelization, inversion)
5. Development of Calculation Laws and Theorems (option)

Course Contents

1. Algorithm Description (in Haskell)
2. Specification (compositions of specific functional units)
3. Laws and Theories for Algorithm Calculation (in Coq)
4. Applications (dynamic programming, parallelization, inversion)
5. Development of Calculation Laws and Theorems (option)
6. Students' Presentation (introduction of a relavant paper)

References

- Roland Backhouse, *Program Construction: Calculating Implementation from Specification*, Wiley, 2003.
- Richard Bird and Oege de Moor, *The Algebra of Programming*, Prentice-Hall, 1996.
- Anne Kaldewaij, Programming: *The Derivation of Algorithms*, Prentice Hall, 1990.

References

- Roland Backhouse, *Program Construction: Calculating Implementation from Specification*, Wiley, 2003.
- Richard Bird and Oege de Moor, *The Algebra of Programming*, Prentice-Hall, 1996.
- Anne Kaldewaij, Programming: *The Derivation of Algorithms*, Prentice Hall, 1990.
- My lecture notes in University of Tokyo:
  - Mathematical Structures in Computer Programs: http://research.nii.ac.jp/~hu/pub/teach/pm08/
  - Mathematical Structures in Programming: http://research.nii.ac.jp/~hu/pub/teach/msp08/

Course Grade

- Your activity in the class (50%)
- Your final presentation and report (50%)

Your questions?

My Questions (in-class discussion)

- What would you like to obtain from this course?
- How to improve programming skills?
- Do you agree that programming is a science?
- Do you like mathematics?
- Should the class be taught in English or in Japanese?