Hirokawa Laboratory: Theory of Computation

Nao Hirokawa, Associate Professor

Area: Term Rewriting

- **Programming Languages**

Area: Term Rewriting

- **Programming Languages**
- **Automated Theorem Proving**

Programming Languages

quick sort (in C)

```
void qsort(int *a, int n)
{
  if (n <= 0) return;
  int i = 0, j = n - 1, x = a[n / 2];
  do {
    while (a[i] < x) i++;
    while (x < a[j]) j--;
    if (i <= j) swap(a, i++, j--);
  } while (i <= j);
  qsort(a, j);
  qsort(a + i, n - i);
}
```

Programming Languages

quick sort (in Haskell)

```
qsort []       = []
qsort (x : xs) =
  qsort [ y | y <- xs, y <  x ] ++ [x] ++
  qsort [ y | y <- xs, y >= x ]
```

Programming Languages

quick sort (in Haskell)

```
qsort []       = []
qsort (x : xs) =
  qsort [ y | y <- xs, y <  x ] ++ [x] ++
  qsort [ y | y <- xs, y >= x ]
```

- language features (pattern matching, λ, laziness, ...)

Programming Languages

quick sort (in Haskell)

```
qsort []       = []
qsort (x : xs) =
  qsort [ y | y <- xs, y <  x ] ++ [x] ++
  qsort [ y | y <- xs, y >= x ]
```

- language features (pattern matching, λ, laziness, ...)
- program transformation (optimization, parsing, ...)

Programming Languages

quick sort (in Haskell)

```
qsort []       = []
qsort (x : xs) =
  qsort [ y | y <- xs, y <  x ] ++ [x] ++
  qsort [ y | y <- xs, y >= x ]
```

- language features (pattern matching, λ, laziness, ...)
- program transformation (optimization, parsing, ...)
- program analysis (termination, complexity analysis, ...)

Programming Languages

quick sort (in Haskell)

```
qsort []       = []
qsort (x : xs) =
  qsort [ y | y <- xs, y <  x ] ++ [x] ++
  qsort [ y | y <- xs, y >= x ]
```

- language features (pattern matching, λ, laziness, ...)
- program transformation (optimization, parsing, ...)
- program analysis (termination, complexity analysis, ...)

Complexity Analysis

power of state-of-art complexity analyzers:

Complexity Analysis

power of state-of-art complexity analyzers:

- bubble sort: $O(n^2)$

Complexity Analysis

power of state-of-art complexity analyzers:

- bubble sort: $O(n^2)$
- merge sort: $O(n^2)$

actually $O(n \log n)$

Complexity Analysis

power of state-of-art complexity analyzers:

- bubble sort: $O(n^2)$
- merge sort: $O(n^2)$ actually $O(n \log n)$
- quick sort: not analyzable actually $O(n^2)$

Complexity Analysis

power of state-of-art complexity analyzers:

- bubble sort: $O(n^2)$
- merge sort: $O(n^2)$ actually $O(n \log n)$
- quick sort: not analyzable actually $O(n^2)$
- Euclidean algorithm: $O(n^2)$ actually $O(n)$

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

2 $x^5 - y^5 = 0$?

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

2 $x^5 - y^5 = 0$?

NO! counterexample: $(x, y) = (-1, 1)$

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

2 $x^5 - y^5 = 0$?

NO! counterexample: $(x, y) = (-1, 1)$

Research Topics:

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

2 $x^5 - y^5 = 0$?

NO! counterexample: $(x, y) = (-1, 1)$

Research Topics:

- deformation, lemma discovery, ...

Automated Reasoning

$$\left\{ \begin{array}{l} x^2y = 1 \\ xy^2 = x \end{array} \right\}$$

1 $x^4 - y^4 = 0$?

YES! easy if you are aware of $y^2 = 1$, $x^2 = y$

2 $x^5 - y^5 = 0$?

NO! counterexample: $(x, y) = (-1, 1)$

Research Topics:

- deformation, lemma discovery, ...
- completion, counterexample generation, ...

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) =$$

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a),$$

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa),$$

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

$$(x, y) = (zb, az) \qquad z\text{: arbitrary string}$$

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

$$(x, y) = (zb, az) \qquad z\text{: arbitrary string}$$

Research Topics:

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

$$(x, y) = (zb, az) \qquad z\text{: arbitrary string}$$

Research Topics:

- existence of solutions and solved forms, ...

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

$$(x, y) = (zb, az) \qquad z\text{: arbitrary string}$$

Research Topics:

- existence of solutions and solved forms, ...
- computation of solutions

Solving Equations

string equation $ax = yb$ admits infinitely many solutions:

$$(x, y) = (b, a), (ab, aa), (bb, ab), (aab, aaa), \ldots$$

how about general solved forms?

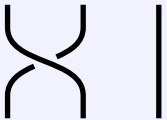
$$(x, y) = (zb, az) \qquad z\text{: arbitrary string}$$

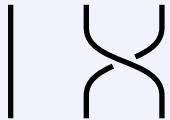
Research Topics:

- existence of solutions and solved forms, ...
- computation of solutions
- how about equation of programs? e.g., $\mathsf{qsort}([3, x, 1]) = [y, 3, 4]$

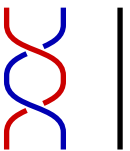
Braid Theory

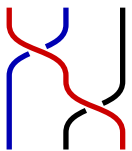
NOTATION

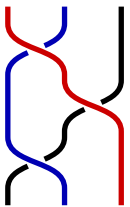
a: 

b: 

EXAMPLE

aa: 

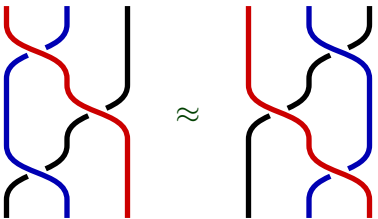
ab: 

aba: 

Equivalence of Braids

DEFINITION

$\mathcal{B} = \{\text{aba} \approx \text{bab}\}$

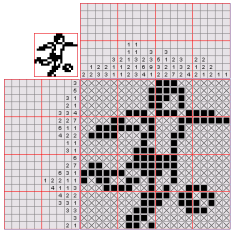


Q

aababab $\approx_{\mathcal{B}}$ ababab?

1st Year

10 —
11 —
12 —
1 **soliving puzzles**
2 **deciding research theme**
3 ⋮
4 hanami
5 ⋮
6 ⋮
7 **tool competition ...**
8 ⋮
9 **domestric meeting**



2nd Year

10 ⋮

11 ⋮

12 ⋮

1 **paper writing**

2 ⋮

3 **domestic meeting**

4 hanami

5 ⋮

6 ⋮

7 **tool competition**

8 ⋮

9 **master's defense**

Seminars

- group seminars — 0.5–1 per week

Seminars

- group seminars — 0.5–1 per week
- individual meetings — 0.5–1 per week

Seminars

- group seminars — 0.5–1 per week
- individual meetings — 0.5–1 per week
- reading group — 0.5–1 per week

Seminars

- group seminars — 0.5–1 per week
- individual meetings — 0.5–1 per week
- reading group — 0.5–1 per week
- joint group seminar — 1 per month

Research Collaborations

- **University of Innsbruck (Austria)**

Research Collaborations

- **University of Innsbruck (Austria)**

(C) Pahu, CC BY-SA 3.0

- **LORIA (France)**

(C) François Bernardin, CC BY 3.0

Our Laboratory

- rooms: **I-53, I-54**
- `http://www.jaist.ac.jp/~hirokawa/laboratory/`

join us if you are interested in

- **principle of computation**
- **programming languages**
- **logic puzzles**
- **computer algebra and theorem provers**