# Execution Traces as a Powerful Data Representation for Intelligent Tutoring Systems for Programming

*Benjamin Paaßen* Joris Jensen Barbara Hammer

Educational Datamining Conference, 30-06-2016, Raleigh, USA

Funding by the DFG under grant number HA 2719/6-2 and the CITEC center of excellence (EXC 277) is gratefully acknowledged.

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction.*

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction. [...] Significant potential for improving the quality of programming education and for reducing instructor work load lies in **computer-based tutoring systems** that can coach students in **solving introductory programming problems**.*

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction. [...] Significant potential for improving the quality of programming education and for reducing instructor work load lies in **computer-based tutoring systems** that can coach students in **solving introductory programming problems**. - Anderson and Skwarecki (1986)*

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction. [...] Significant potential for improving the quality of programming education and for reducing instructor work load lies in **computer-based tutoring systems** that can coach students in **solving introductory programming problems**. - Anderson and Skwarecki (1986)*

- We need to **do** programming to **learn** programming

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction. [...] Significant potential for improving the quality of programming education and for reducing instructor work load lies in **computer-based tutoring systems** that can coach students in **solving introductory programming problems**. - Anderson and Skwarecki (1986)*

- We need to **do** programming to **learn** programming
- Insufficient faculty time to tutor programming attempts

# ITSs for Programming (1)

*With the **growing number** of high-school and college students currently taking introductory computer programming courses, sizeable amounts of **faculty time** are spent on course development and instruction. [...] Significant potential for improving the quality of programming education and for reducing instructor work load lies in **computer-based tutoring systems** that can coach students in **solving introductory programming problems**. - Anderson and Skwarecki (1986)*

- We need to **do** programming to **learn** programming
- Insufficient faculty time to tutor programming attempts

⇒ ITSs are desirable

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

student state

$x_t$

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

student state

$x_t$

$m$

database

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

student state

$$\begin{tikzcd} & x_t \arrow[d] \\ m \arrow[r] & \mathcal{N}(x_t) \end{tikzcd}$$

database similar states

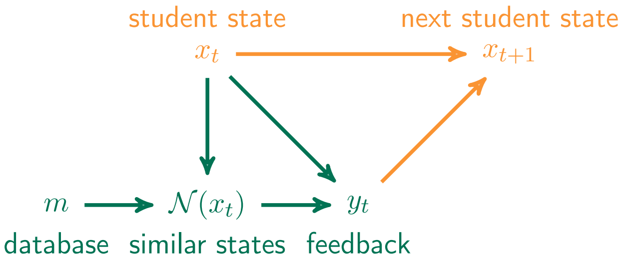
# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

$$\begin{tikzcd}
 & \text{student state} & \\
 & x_t \arrow[d] \arrow[dr] & \\
m \arrow[r] & \mathcal{N}(x_t) \arrow[r] & y_t \\
\text{database} & \text{similar states} & \text{feedback}
\end{tikzcd}$$

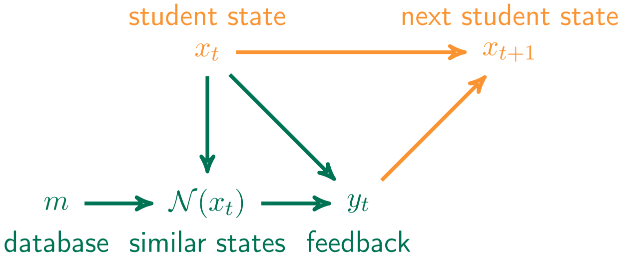
# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)



# Data-Driven ITSs

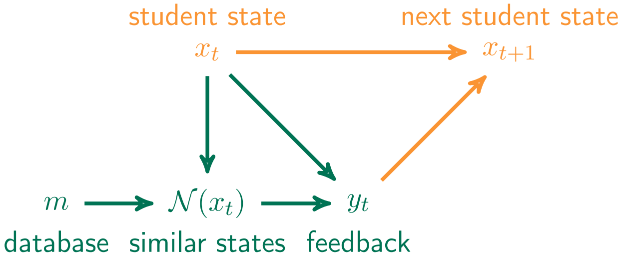
(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)




Main Research Questions:

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

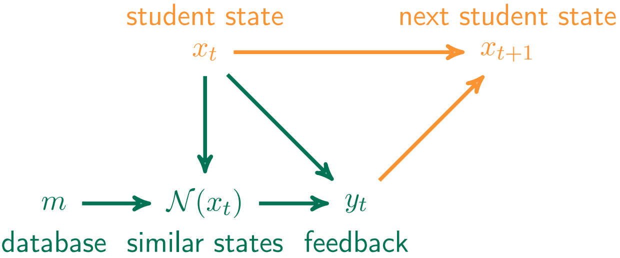



Main Research Questions:

- How to retrieve **similar states**?

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

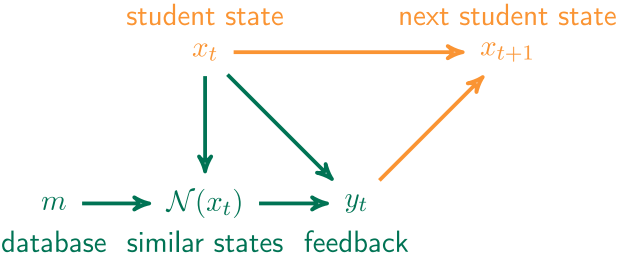



Main Research Questions:

- How to retrieve **similar states**?
- How to generate **feedback**? In particular: Error detection

# Data-Driven ITSs

(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)




Main Research Questions:

- How to retrieve **similar states**?
- How to generate **feedback**? In particular: Error detection

⇒ Depends on **data representation**

# Data Representations: Syntax and Traces

# Syntax (1)

```
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Syntax (1)

```java
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

```java
public static int[] insertionSort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = l; i < r; i++) {
    for (int j = i - 1; j >= l; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Syntax (1)

```
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

```
public static int[] insertionSort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = l; i < r; i++) {
    for (int j = i - 1; j >= l; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Syntax (2)

```java
public static int[] insertionSort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = l; i < r; i++) {
    for (int j = i - 1; j >= l; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

```java
public static int[] insertionSort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  insertionSort(A, l, r);
  return A;
}
private static void insertionSort(int[] A, int l, int r) {
  if (l < r) {
    insertionSort(A, l, r - 1);
    insert(A, l, r);
  }
}
private static void insert(int[] A, int l, int r) {
  if (l < r) {
    if (A[r - 1] > A[r]) {
      final int tmp = A[r - 1];
      A[r - 1] = A[r];
      A[r] = tmp;
    }
    insert(A, l, r - 1);
  }
}
```

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| |
|---|
| |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l |
|---|---|
| [4,7,2,1] | 0 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r |
|---|---|---|
| [4,7,2,1] | 0 | |
| [4,7,2,1] | 0 | 3 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i |
|---|---|---|---|
| [4,7,2,1] | 0 | | |
| [4,7,2,1] | 0 | 3 | |
| [4,7,2,1] | 0 | 3 | 3 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j |
|---|---|---|---|---|
| [4,7,2,1] | 0 | | | |
| [4,7,2,1] | 0 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | |
| [4,7,2,1] | 0 | 3 | 3 | 0 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j |
|---|---|---|---|---|
| [4,7,2,1] | 0 | | | |
| [4,7,2,1] | 0 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | |
| [4,7,2,1] | 0 | 3 | 3 | 0 |
| ... | | | | |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j | tmp |
|---|---|---|---|---|---|
| [4,7,2,1] | 0 | | | | |
| [4,7,2,1] | 0 | 3 | | | |
| [4,7,2,1] | 0 | 3 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | 0 | |
| ... | | | | | |
| [4,7,2,1] | 0 | 3 | 3 | 1 | 7 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j | tmp |
|---|---|---|---|---|---|
| [4,7,2,1] | 0 | | | | |
| [4,7,2,1] | 0 | 3 | | | |
| [4,7,2,1] | 0 | 3 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | 0 | |
| ... | | | | | |
| [4,7,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,2,1] | 0 | 3 | 3 | 1 | 7 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j | tmp |
|---|---|---|---|---|---|
| [4,7,2,1] | 0 | | | | |
| [4,7,2,1] | 0 | 3 | | | |
| [4,7,2,1] | 0 | 3 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | 0 | |
| ... | | | | | |
| [4,7,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,7,1] | 0 | 3 | 3 | 1 | 7 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j | tmp |
|---|---|---|---|---|---|
| [4,7,2,1] | 0 | | | | |
| [4,7,2,1] | 0 | 3 | | | |
| [4,7,2,1] | 0 | 3 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | 0 | |
| ... | | | | | |
| [4,7,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,7,1] | 0 | 3 | 3 | 1 | 7 |

# Traces (1)

```
int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

| A | l | r | i | j | tmp |
|---|---|---|---|---|---|
| [4,7,2,1] | 0 | | | | |
| [4,7,2,1] | 0 | 3 | | | |
| [4,7,2,1] | 0 | 3 | 3 | | |
| [4,7,2,1] | 0 | 3 | 3 | 0 | |
| ... | | | | | |
| [4,7,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,2,1] | 0 | 3 | 3 | 1 | 7 |
| [4,2,7,1] | 0 | 3 | 3 | 1 | 7 |

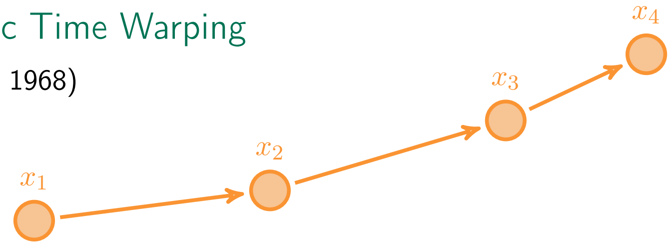
# Traces (2)

| *Bubble* | *Insertion* | recursive |
|---|---|---|
| $[4, 7, 2, 1]$ | $[4, 7, 2, 1]$ | $[4, 7, 2, 1]$ |
| $[4, 2, 7, 1]$ | $[4, 2, 7, 1]$ | $[4, 2, 7, 1]$ |
| $[4, 2, 1, 7]$ | $[2, 4, 7, 1]$ | $[2, 4, 7, 1]$ |
| $[2, 4, 1, 7]$ | $[2, 4, 1, 7]$ | $[2, 4, 1, 7]$ |
| $[2, 1, 4, 7]$ | $[2, 1, 4, 7]$ | $[2, 1, 4, 7]$ |
| $[1, 2, 4, 7]$ | $[1, 2, 4, 7]$ | $[1, 2, 4, 7]$ |

# Comparing Traces: Dynamic Time Warping

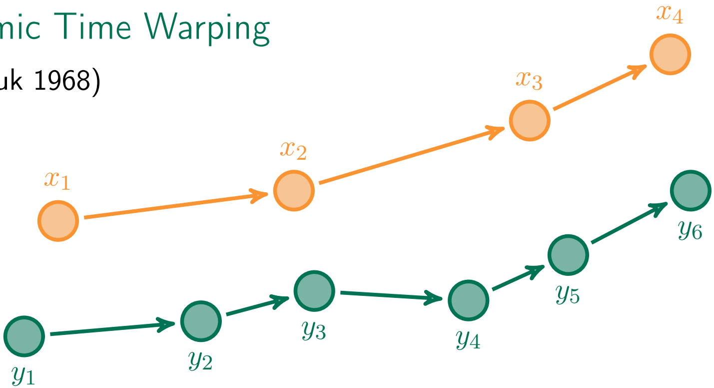
# Dynamic Time Warping

(Vintsyuk 1968)



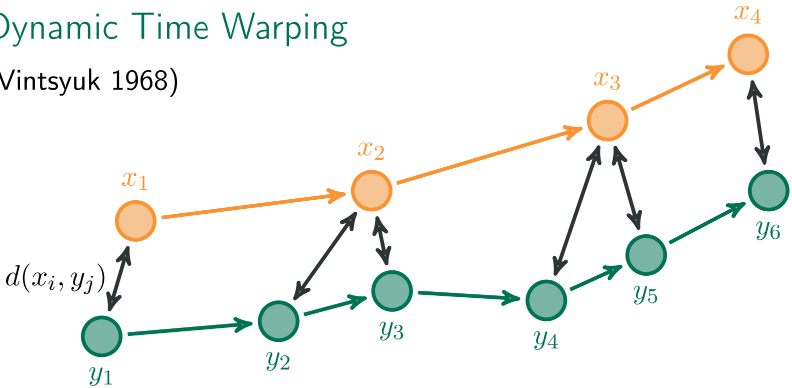
# Dynamic Time Warping

(Vintsyuk 1968)



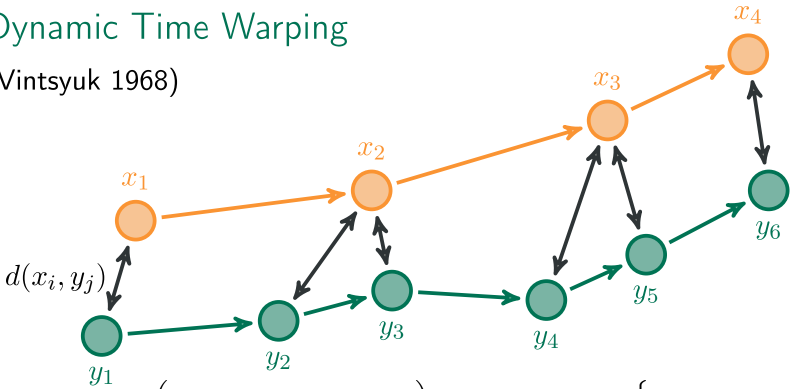
# Dynamic Time Warping

(Vintsyuk 1968)



# Dynamic Time Warping

(Vintsyuk 1968)




$$D_{\text{DTW}}\Big((x_1,\ldots,x_i),(y_1,\ldots,y_j)\Big) := d(x_i,y_j) + \min\Big\{ D_{\text{DTW}}\Big((x_1,\ldots,x_{i-1}),(y_1,\ldots,y_{j-1})\Big), D_{\text{DTW}}\Big((x_1,\ldots,x_{i-1}),(y_1,\ldots,y_j)\Big), D_{\text{DTW}}\Big((x_1,\ldots,x_i),(y_1,\ldots,y_{j-1})\Big)\Big\}$$

# Dynamic Time Warping: Example

| | |
|---|---|
| [4,7,2,1] | [4,7,2,1] |
| [4,7,2,1] | [4,2,2,1] |
| [4,7,2,1] | [4,2,7,1] |
| [4,2,2,1] | [4,2,7,1] |
| [4,2,7,1] | [2,2,7,1] |
| [4,2,7,1] | [2,4,7,1] |
| [4,2,7,1] | |
| [4,2,1,1] | |
| [4,2,1,7] | |

# Dynamic Time Warping: Example

| | | |
|---|---|---|
| [4,7,2,1] | $d(x_i, y_j) = 0$ | [4,7,2,1] |
| [4,7,2,1] | | [4,2,2,1] |
| [4,7,2,1] | | [4,2,7,1] |
| [4,2,2,1] | | [4,2,7,1] |
| [4,2,7,1] | | [2,2,7,1] |
| [4,2,7,1] | | [2,4,7,1] |
| [4,2,7,1] | | |
| [4,2,1,1] | | |
| [4,2,1,7] | | |

# Dynamic Time Warping: Example

| | | |
|---|---|---|
| [4,7,2,1] | $d(x_i,y_j) = 0$ | [4,7,2,1] |
| [4,7,2,1] | $d(x_i,y_j) = 0$ | [4,2,2,1] |
| [4,7,2,1] | | [4,2,7,1] |
| [4,2,2,1] | | [4,2,7,1] |
| [4,2,7,1] | | [2,2,7,1] |
| [4,2,7,1] | | [2,4,7,1] |
| [4,2,7,1] | | |
| [4,2,1,1] | | |
| [4,2,1,7] | | |

# Dynamic Time Warping: Example

| | | |
|---|---|---|
| [4,7,2,1] | $d(x_i,y_j) = 0$ | [4,7,2,1] |
| [4,7,2,1] | $d(x_i,y_j) = 0$ | [4,2,2,1] |
| [4,7,2,1] | $d(x_i,y_j) = 0$ | [4,2,7,1] |
| [4,2,2,1] | | [4,2,7,1] |
| [4,2,7,1] | | [2,2,7,1] |
| [4,2,7,1] | | [2,4,7,1] |
| [4,2,7,1] | | |
| [4,2,1,1] | | |
| [4,2,1,7] | | |

# Dynamic Time Warping: Example

[4,7,2,1] $d(x_i,y_j) = 0$ [4,7,2,1]

[4,7,2,1] $d(x_i,y_j) = 0$ [4,2,2,1]

[4,7,2,1] $d(x_i,y_j) = 0$ [4,2,7,1]

[4,2,2,1] [4,2,7,1]

[4,2,7,1] $d(x_i,y_j) = 0.5$ [2,2,7,1]

[4,2,7,1] [2,4,7,1]

[4,2,7,1]

[4,2,1,1]

[4,2,1,7]

# Dynamic Time Warping: Example

[4,7,2,1] — $d(x_i,y_j) = 0$ — [4,7,2,1]

[4,7,2,1]

[4,7,2,1]

[4,2,2,1] — $d(x_i,y_j) = 0$ — [4,2,2,1]

[4,2,7,1] — $d(x_i,y_j) = 0$ — [4,2,7,1]

[4,2,7,1] — [4,2,7,1]

[4,2,7,1] — $d(x_i,y_j) = 0.5$ — [2,2,7,1]

[4,2,1,1] — $d(x_i,y_j) = 1$ — [2,4,7,1]

[4,2,1,7]

# Frame Dissimilarity

- DTW relies on a **frame dissimilarity**

# Frame Dissimilarity

- DTW relies on a **frame dissimilarity**
- Requires knowledge about **relevant variables** (e.g. count unequal elements in array)

# Frame Dissimilarity

- DTW relies on a **frame dissimilarity**
- Requires knowledge about **relevant variables** (e.g. count unequal elements in array)
- More generic approach: Compare **type histograms** (drawback: ignores variable content)

# Frame Dissimilarity

- DTW relies on a **frame dissimilarity**
- Requires knowledge about **relevant variables** (e.g. count unequal elements in array)
- More generic approach: Compare **type histograms** (drawback: ignores variable content)

$\Rightarrow$ Subject to ongoing research

# Experiments

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

Methodological setup:

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

Methodological setup:

- Experiments 1 and 2: Classification task with 1-nearest neighbor accuracy

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

Methodological setup:

- Experiments 1 and 2: Classification task with 1-nearest neighbor accuracy
- Experiment 3: Pattern recognition task with precision and recall

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

Methodological setup:

- Experiments 1 and 2: Classification task with 1-nearest neighbor accuracy
- Experiment 3: Pattern recognition task with precision and recall
- Comparison with state-of-the-art syntactic representation (Paaßen, Mokbel, and Hammer 2016)

# Overview

1. Do we recognize **strategic differences** and ignore **stylistic differences**?
2. Do we recognize **erroneous programs**?
3. Do we detect the **location of errors**?

Methodological setup:

- Experiments 1 and 2: Classification task with 1-nearest neighbor accuracy
- Experiment 3: Pattern recognition task with precision and recall
- Comparison with state-of-the-art syntactic representation (Paaßen, Mokbel, and Hammer 2016)
- TCS Alignment Toolbox (Paaßen, Mokbel, and Hammer 2015a)

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each
- `http://doi.org/10.4119/unibi/2900666`

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each
- `http://doi.org/10.4119/unibi/2900666`

## Sorting Programs

(Paaßen, Mokbel, and Hammer 2015b)

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each
- `http://doi.org/10.4119/unibi/2900666`

## Sorting Programs

(Paaßen, Mokbel, and Hammer 2015b)

- 126 (correct) sorting programs from the web

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each
- `http://doi.org/10.4119/unibi/2900666`

## Sorting Programs

(Paaßen, Mokbel, and Hammer 2015b)

- 126 (correct) sorting programs from the web
- 6 different sorting algorithms (Bubble, Insertion, Selection, Shell, Quick, Merge)

# Datasets

## Palindrome Detection Programs

(Mokbel et al. 2013)

- 48 (correct) artificial programs
- 8 different approaches with 6 small variations each
- `http://doi.org/10.4119/unibi/2900666`

## Sorting Programs

(Paaßen, Mokbel, and Hammer 2015b)

- 126 (correct) sorting programs from the web
- 6 different sorting algorithms (Bubble, Insertion, Selection, Shell, Quick, Merge)
- `http://doi.org/10.4119/unibi/2900684`

# Strategy Classification: Results

| method | palindromes | sorting |
|---|---|---|
| syntax | 0.8750 (0.1581) | 0.8118 (0.0676) |
| traces | 0.9792 (0.0510) | 0.9535 (0.0403) |

mean classification accuracy (and standard deviation) across cross-validation trials

# Strategy Classification: Results

| method | palindromes | sorting |
|---|---|---|
| syntax | 0.8750 (0.1581) | 0.8118 (0.0676) |
| traces | 0.9792 (0.0510) | 0.9535 (0.0403) |

mean classification accuracy (and standard deviation) across cross-validation trials

$\Rightarrow$ Significant difference for **sorting** dataset ($p < 0.01$).

# Artificial Errors

Introduce **erroneous counterparts** for sorting programs, such that:

# Artificial Errors

Introduce **erroneous counterparts** for sorting programs, such that:

- program still compiles and executes without exception

# Artificial Errors

Introduce **erroneous counterparts** for sorting programs, such that:

- program still compiles and executes without exception
- produces the wrong (e.g. unsorted) output

# Artificial Errors

Introduce **erroneous counterparts** for sorting programs, such that:

- program still compiles and executes without exception
- produces the wrong (e.g. unsorted) output
- still is plausible

# Error Classification: Results

| method | accuracy |
|---|---|
| syntax | 0.2112 (0.1073) |
| traces | 0.8608 (0.0856) |

mean classification accuracy (and standard deviation) across cross-validation trials on the sorting dataset with artificial errors.

# Error Detection: Syntax

```
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j >= 0; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Error Detection: Syntax

```java
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j >= 0; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

```java
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Error Detection: Syntax

```
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j >= 0; j--) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

```
public static int[] bubblesort(int[] A) {
  final int l = 0;
  final int r = A.length - 1;
  for (int i = r; i > l; i--) {
    for (int j = l; j < i; j++) {
      if (A[j] > A[j + 1]) {
        final int tmp = A[j];
        A[j] = A[j + 1];
        A[j + 1] = tmp;
      }
    }
  }
  return A;
}
```

# Error Detection: Traces

```
[4,7,2,1]
[4,7,2,1]
[4,7,2,1]
[4,7,7,1]
[4,2,7,1]
[4,2,7,1]
[4,2,7,1]
[4,2,1,1]
[4,2,1,7]
```

# Error Detection: Traces

| | |
|---|---|
| [4,7,2,1] | [4,7,2,1] |
| [4,7,2,1] | [4,2,2,1] |
| [4,7,2,1] | [4,2,7,1] |
| [4,7,7,1] | [4,2,7,1] |
| [4,2,7,1] | [2,2,7,1] |
| [4,2,7,1] | [2,4,7,1] |
| [4,2,7,1] | |
| [4,2,1,1] | |
| [4,2,1,7] | |

# Error Detection: Traces

| | |
|---|---|
| [4,7,2,1] | [4,7,2,1] |
| [4,7,2,1] | [4,2,2,1] |
| [4,7,2,1] | [4,2,7,1] |
| [4,7,7,1] | [4,2,7,1] |
| [4,2,7,1] | [2,2,7,1] |
| [4,2,7,1] | [2,4,7,1] |
| [4,2,7,1] | |
| [4,2,1,1] | |
| [4,2,1,7] | |

# Error Detection: Traces

| | |
|---|---|
| [4,7,2,1] | [4,7,2,1] |
| [4,7,2,1] | [4,2,2,1] |
| [4,7,2,1] | [4,2,7,1] |
| [4,7,7,1] | [4,2,7,1] |
| [4,2,7,1] | [2,2,7,1] |
| [4,2,7,1] | [2,4,7,1] |
| [4,2,7,1] | |
| [4,2,1,1] | |
| [4,2,1,7] | |

# Error Detection: Results

| method | precision | recall |
|---|---|---|
| traces | 0.1833 (0.0714) | 0.5202 (0.2105) |
| syntax | 0.1031 (0.0859) | 0.1335 (0.1004) |

mean precision and recall (with standard deviation in brackets) on the **sorting** dataset

# Error Detection: Results

| method | precision | recall |
|---|---|---|
| traces | 0.1833 (0.0714) | 0.5202 (0.2105) |
| syntax | 0.1031 (0.0859) | 0.1335 (0.1004) |

mean precision and recall (with standard deviation in brackets) on the **sorting** dataset

$\Rightarrow$ Significant difference in F1 score ($p < 0.0001$)

# Conclusion

# Conclusion

- Trace representation outperforms purely syntactic representation.

# Conclusion

- Trace representation outperforms purely syntactic representation.
- Requires more domain knowledge and only works on compilable programs.

# Conclusion

- Trace representation outperforms purely syntactic representation.
- Requires more domain knowledge and only works on compilable programs.

$\Rightarrow$ Best used in conjunction with syntax representation to improve data-driven ITSs.

# Conclusion

- Trace representation outperforms purely syntactic representation.
- Requires more domain knowledge and only works on compilable programs.

$\Rightarrow$ Best used in conjunction with syntax representation to improve data-driven ITSs.

Thank you for your attention!

# Literature I

Anderson, J. R and E. Skwarecki (1986). "The Automated Tutoring of Introductory Computer Programming". In: *Commun. ACM* 29.9, pp. 842–849. DOI: 10.1145/6592.6593.

Gross, Sebastian et al. (2014). "Example-based Feedback Provision Using Structured Solution Spaces". In: *International Journal of Learning Technology* 9.3, pp. 248–280. ISSN: 1477-8386. DOI: 10.1504/IJLT.2014.065752.

Hicks, Andrew et al. (2015). "BOTS: Selecting Next-Steps from Player Traces in a Puzzle Game". In: *Workshops Proceedings of EDM 2015 8th International Conference on Educational Data Mining, EDM 2015, Madrid, Spain, June 26-29, 2015.* URL: http://ceur-ws.org/Vol-1446/GEDM_2015_Submission_10.pdf.

# Literature II

Mokbel, Bassam et al. (2013). "Domain-Independent Proximity Measures in Intelligent Tutoring Systems". In: *Proceedings of the 6th International Conference on Educational Data Mining (EDM)*. Ed. by S. K. D'Mello, R. A. Calvo, and A. Olney. Memphis, Tennessee, USA.

Paaßen, Benjamin, Bassam Mokbel, and Barbara Hammer (2015a). "A Toolbox for Adaptive Sequence Dissimilarity Measures for Intelligent Tutoring Systems". English. In: ed. by Olga Christina Santos et al. Proceedings of the 8th International Conference on Educational Data Mining. Madrid, Spain: International Educational Datamining Society, pp. 632–632. ISBN: 978-84-606-9425-0. URL: http://www.educationaldatamining.org/EDM2015/uploads/papers/paper%5C_257.pdf.

# Literature III

Paaßen, Benjamin, Bassam Mokbel, and Barbara Hammer (2015b). "Adaptive structure metrics for automated feedback provision in Java programming". English. In: *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. Ed. by Michel Verleysen. Bruges, Belgium, pp. 307–312.

Paaßen, Benjamin, Bassam Mokbel, and Barbara Hammer (2016). "Adaptive structure metrics for automated feedback provision in intelligent tutoring systems". In: *Neurocomputing* 192. DOI: doi:10.1016/j.neucom.2015.12.108.

# Literature IV

Rivers, Kelly and Kenneth R. Koedinger (2015). "Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor". In: *International Journal of Artificial Intelligence in Education*, pp. 1–28. DOI: 10.1007/s40593-015-0070-z.

Vintsyuk, T.K. (1968). "Speech discrimination by dynamic programming". English. In: *Cybernetics* 4.1, pp. 52–57. ISSN: 0011-4235.