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.

Licensed according to CC-BY-SA 3.0
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.
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**.
**ITSSs 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 workload lies in computer-based tutoring systems that can coach students in solving introductory programming problems. - Anderson and Skwarecki (1986)
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 workload 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
ITSSs 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
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)

\[ x_t \]

student state

\[ m \]

database

Main Research Questions:
▶ How to retrieve similar states?
▶ How to generate feedback? In particular: Error detection ⇒ Depends on data representation
Data-Driven ITSs

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

student state

\[ x_t \]

\[ m \rightarrow \mathcal{N}(x_t) \]

database similar states

Main Research Questions:

▶ How to retrieve similar states?

▶ How to generate feedback? In particular: Error detection ⇒ Depends on data representation
Data-Driven ITSs

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

student state

\[ x_t \]

\[ m \rightarrow N(x_t) \rightarrow y_t \]

database  similar states  feedback

Main Research Questions:

▶ How to retrieve similar states?
▶ How to generate feedback? In particular: Error detection → Depends on data representation.
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

\[ x_t \rightarrow N(x_t) \rightarrow y_t \]

student state \( x_t \)
next student state \( x_{t+1} \)
database similar states feedback
Data-Driven ITSs
(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

\[ m \rightarrow \mathcal{N}(x_t) \rightarrow y_t \]

database  similar states  feedback

Main Research Questions:

- How to retrieve similar states?
- How to generate feedback? In particular: Error detection ⇒ Depends on data representation
Data-Driven ITSs
(Gross et al. 2014; Hicks et al. 2015; Rivers and Koedinger 2015)

Main Research Questions:
▶ How to retrieve similar states?

\[ x_t \xrightarrow{} m \xrightarrow{} N(x_t) \xrightarrow{} y_t \xrightarrow{} x_{t+1} \]

student state \→\ database \→\ similar states \→\ feedback

next student state
Data-Driven ITSs

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

\[ x_t \rightarrow m \rightarrow \mathcal{N}(x_t) \rightarrow y_t \rightarrow x_{t+1} \]

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)

student state \( x_t \) \rightarrow \text{database} \rightarrow \mathcal{N}(x_t) \rightarrow \text{similar states} \rightarrow \text{feedback} \rightarrow x_{t+1}

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
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[] 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;
}
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;
}
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;
}

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)

```java
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)

```java
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)

```java
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;
}
```

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
Traces (1)

```java
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;
}
```

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
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;
}

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
Traces (1)

```java
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)

```java
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)

```java
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;
}
```

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
<th>j</th>
<th>tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>#...#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>[4,2,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Traces (1)

```java
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;
}
```

```
<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
<th>j</th>
<th>tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[4,2,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>[4,2,7,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
```

...
Traces (1)

```java
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;
}
```

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
<th>j</th>
<th>tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>[4,2,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>[4,2,7,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Traces (1)

```java
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;
}
```

<table>
<thead>
<tr>
<th>A</th>
<th>l</th>
<th>r</th>
<th>i</th>
<th>j</th>
<th>tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,7,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>[4,2,2,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>[4,2,7,1]</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

---

www.uni-bielefeld.de
Traces (2)

<table>
<thead>
<tr>
<th>Bubble</th>
<th>Insertion</th>
<th>recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4, 7, 2, 1]</td>
<td>[4, 7, 2, 1]</td>
<td>[4, 7, 2, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td>[4, 2, 7, 1]</td>
<td>[4, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 1, 7]</td>
<td>[2, 4, 7, 1]</td>
<td>[2, 4, 7, 1]</td>
</tr>
<tr>
<td>[2, 4, 1, 7]</td>
<td>[2, 4, 1, 7]</td>
<td>[2, 4, 1, 7]</td>
</tr>
<tr>
<td>[2, 1, 4, 7]</td>
<td>[2, 1, 4, 7]</td>
<td>[2, 1, 4, 7]</td>
</tr>
<tr>
<td>[1, 2, 4, 7]</td>
<td>[1, 2, 4, 7]</td>
<td>[1, 2, 4, 7]</td>
</tr>
</tbody>
</table>
Comparing Traces: Dynamic Time Warping
Dynamic Time Warping
(Vintsyuk 1968)

\[
D_{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j) := \min \{D_{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{j-1}), D_{DTW}(x_1, \ldots, x_{i-1}), D_{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j)\}
\]
Dynamic Time Warping
(Vintsyuk 1968)

\[ d(x_i, y_j) \]

\[ \text{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j) := d(x_i, y_j) + \min \{ \text{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{j-1}), \text{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_j), \text{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_{j-1}) \} \]
Dynamic Time Warping

(Vintsyuk 1968)

\[ d(x_i, y_j) \]

\[ \text{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_j) := \min \{ \]
\[ \text{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_{j-1}), \]
\[ \text{DTW}(x_1, \ldots, x_{i-1}, y_1, \ldots, y_j), \]
\[ \text{DTW}(x_1, \ldots, x_i, y_1, \ldots, y_{j-1}) \} \]
Dynamic Time Warping

(Vintsyuk 1968)

\[ D_{DTW}\left((x_1, \ldots, x_i), (y_1, \ldots, y_j)\right) := d(x_i, y_j) + \min \left\{ \right. \]
\[ D_{DTW}\left((x_1, \ldots, x_{i-1}), (y_1, \ldots, y_{j-1})\right), \]
\[ D_{DTW}\left((x_1, \ldots, x_{i-1}), (y_1, \ldots, y_j)\right), \]
\[ D_{DTW}\left((x_1, \ldots, x_i), (y_1, \ldots, y_{j-1})\right) \left\} \right. \]
Dynamic Time Warping: Example

\[
\begin{array}{c|c}
[4, 7, 2, 1] & [4, 7, 2, 1] \\
[4, 7, 2, 1] & [4, 2, 2, 1] \\
[4, 7, 2, 1] & [4, 2, 7, 1] \\
[4, 2, 2, 1] & [4, 2, 7, 1] \\
[4, 2, 7, 1] & [2, 2, 7, 1] \\
[4, 2, 7, 1] & [2, 4, 7, 1] \\
[4, 2, 7, 1] & \\
[4, 2, 1, 1] & \\
[4, 2, 1, 7] & \\
\end{array}
\]
Dynamic Time Warping: Example

\[
\begin{align*}
\text{d}(x_i, y_j) &= 0 \\
[4, 7, 2, 1] &\quad [4, 7, 2, 1] \\
[4, 7, 2, 1] &\quad [4, 2, 2, 1] \\
[4, 7, 2, 1] &\quad [4, 2, 7, 1] \\
[4, 2, 2, 1] &\quad [4, 2, 7, 1] \\
[4, 2, 7, 1] &\quad [2, 2, 7, 1] \\
[4, 2, 7, 1] &\quad [2, 4, 7, 1] \\
[4, 2, 1, 1] &\quad [4, 2, 1, 7]
\end{align*}
\]
Dynamic Time Warping: Example

\[
\begin{array}{c}
[4, 7, 2, 1] \\
[4, 7, 2, 1] \\
[4, 7, 2, 1] \\
[4, 2, 2, 1] \\
[4, 2, 7, 1] \\
[4, 2, 7, 1] \\
[4, 2, 7, 1] \\
[4, 2, 1, 1] \\
[4, 2, 1, 7] \\
[4, 7, 2, 1] \\
[4, 2, 2, 1] \\
[4, 2, 7, 1] \\
[4, 2, 7, 1] \\
[2, 2, 7, 1] \\
[4, 2, 7, 1] \\
[2, 2, 7, 1] \\
[2, 4, 7, 1] \\
[4, 2, 7, 1] \\
[4, 2, 1, 1] \\
[4, 2, 1, 7]
\end{array}
\]

\[d(x_i, y_j) = 0\]
Dynamic Time Warping: Example

```plaintext
\begin{align*}
[4, 7, 2, 1] & \quad d(x_i, y_j) = 0 & [4, 7, 2, 1] \\
[4, 7, 2, 1] & \quad d(x_i, y_j) = 0 & [4, 2, 2, 1] \\
[4, 7, 2, 1] & \quad \quad & [4, 2, 7, 1] \\
[4, 2, 2, 1] & \quad d(x_i, y_j) = 0 & [4, 2, 7, 1] \\
[4, 2, 7, 1] & \quad \quad & [4, 2, 7, 1] \\
[4, 2, 7, 1] & \quad \quad & [2, 2, 7, 1] \\
[4, 2, 7, 1] & \quad \quad & [2, 4, 7, 1] \\
[4, 2, 1, 1] & \quad \quad & \\
[4, 2, 1, 7] & \quad \quad & \\
\end{align*}
```
Dynamic Time Warping: Example

\[
\begin{array}{cccc}
[4, 7, 2, 1] & \text{d}(x_i, y_j) = 0 & [4, 7, 2, 1] \\
[4, 7, 2, 1] & \text{d}(x_i, y_j) = 0 & [4, 2, 2, 1] \\
[4, 7, 2, 1] & \text{d}(x_i, y_j) = 0 & [4, 2, 7, 1] \\
[4, 2, 2, 1] & \text{d}(x_i, y_j) = 0 & [4, 2, 7, 1] \\
[4, 2, 7, 1] & \text{d}(x_i, y_j) = 0.5 & [2, 2, 7, 1] \\
[4, 2, 7, 1] & \text{d}(x_i, y_j) = 0.5 & [2, 4, 7, 1] \\
[4, 2, 1, 1] & & [2, 4, 7, 1] \\
[4, 2, 1, 7] & & \\
\end{array}
\]
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] | d(x_i, y_j) = 0.5 | [2, 2, 7, 1] |
| [4, 2, 7, 1] | d(x_i, y_j) = 1 | [2, 4, 7, 1] |
| [4, 2, 7, 1] |            |              |
| [4, 2, 1, 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)
- 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](http://doi.org/10.4119/unibi/2900666)
## Datasets

<table>
<thead>
<tr>
<th>Palindrome Detection Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mokbel et al. 2013)</td>
</tr>
<tr>
<td>▶ 48 (correct) artificial programs</td>
</tr>
<tr>
<td>▶ 8 different approaches with 6 small variations each</td>
</tr>
<tr>
<td>▶ <a href="http://doi.org/10.4119/unibi/2900666">http://doi.org/10.4119/unibi/2900666</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sorting Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Paaßen, Mokbel, and Hammer 2015b)</td>
</tr>
</tbody>
</table>
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](http://doi.org/10.4119/unibi/2900666)

### Sorting Programs

(Paasßen, 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

<table>
<thead>
<tr>
<th>method</th>
<th>palindromes</th>
<th>sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntax</td>
<td>0.8750 (0.1581)</td>
<td>0.8118 (0.0676)</td>
</tr>
<tr>
<td>traces</td>
<td>0.9792 (0.0510)</td>
<td>0.9535 (0.0403)</td>
</tr>
</tbody>
</table>

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

Significant difference for sorting dataset ($p < 0.01$).
Strategy Classification: Results

<table>
<thead>
<tr>
<th>method</th>
<th>palindromes</th>
<th>sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntax</td>
<td>0.8750 (0.1581)</td>
<td>0.8118 (0.0676)</td>
</tr>
<tr>
<td>traces</td>
<td>0.9792 (0.0510)</td>
<td>0.9535 (0.0403)</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>method</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>syntax</td>
<td>0.2112 (0.1073)</td>
</tr>
<tr>
<td>traces</td>
<td>0.8608 (0.0856)</td>
</tr>
</tbody>
</table>

Mean classification accuracy (and standard deviation) across cross-validation trials on the sorting dataset with artificial errors.
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;
}
```
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: 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

<table>
<thead>
<tr>
<th>[4, 7, 2, 1]</th>
<th>[4, 7, 2, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4, 7, 2, 1]</td>
<td>[4, 2, 2, 1]</td>
</tr>
<tr>
<td>[4, 7, 7, 1]</td>
<td>[4, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td>[2, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td>[2, 4, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td></td>
</tr>
<tr>
<td>[4, 2, 1, 1]</td>
<td></td>
</tr>
<tr>
<td>[4, 2, 1, 7]</td>
<td></td>
</tr>
</tbody>
</table>
Error Detection: Traces

<table>
<thead>
<tr>
<th>[4, 7, 2, 1]</th>
<th>[4, 7, 2, 1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4, 7, 2, 1]</td>
<td>[4, 2, 2, 1]</td>
</tr>
<tr>
<td>[4, 7, 2, 1]</td>
<td>[4, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 7, 7, 1]</td>
<td>[4, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td>[2, 2, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 7, 1]</td>
<td>[2, 4, 7, 1]</td>
</tr>
<tr>
<td>[4, 2, 1, 1]</td>
<td>[4, 2, 1, 7]</td>
</tr>
</tbody>
</table>
Error Detection: Traces

\[
\begin{align*}
[4, 7, 2, 1] & \quad [4, 7, 2, 1] \\
[4, 7, 2, 1] & \quad [4, 2, 2, 1] \\
[4, 7, 2, 1] & \quad [4, 2, 7, 1] \\
[4, 7, 7, 1] & \quad [4, 2, 7, 1] \\
[4, 2, 7, 1] & \quad [2, 2, 7, 1] \\
[4, 2, 7, 1] & \quad [2, 4, 7, 1] \\
[4, 2, 1, 1] & \quad [4, 2, 1, 1] \\
[4, 2, 1, 7] & \quad [4, 2, 1, 7] \\
\end{align*}
\]
Error Detection: Results

<table>
<thead>
<tr>
<th>method</th>
<th>precision</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>traces</td>
<td>0.1833 (0.0714)</td>
<td>0.5202 (0.2105)</td>
</tr>
<tr>
<td>syntax</td>
<td>0.1031 (0.0859)</td>
<td>0.1335 (0.1004)</td>
</tr>
</tbody>
</table>

mean precision and recall (with standard deviation in brackets) on the sorting dataset
## Error Detection: Results

<table>
<thead>
<tr>
<th>method</th>
<th>precision</th>
<th>recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>traces</td>
<td>0.1833 (0.0714)</td>
<td>0.5202 (0.2105)</td>
</tr>
<tr>
<td>syntax</td>
<td>0.1031 (0.0859)</td>
<td>0.1335 (0.1004)</td>
</tr>
</tbody>
</table>

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

⇒ 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.
- 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.
  ⇒ Best used in conjunction with syntax representation to improve data-driven ITSs.

Thank you for your attention!
Literature I


Literature II


Literature III


Literature IV
