ON TEACHING PROGRAMMING WITH NONDETERMINISM

Giora Alexandron, Michal Armoni, Michal Gordon, David Harel
NONDETERMINISM (ND)

- A fundamental idea of CS
  - First introduced into CS by Rabin and Scott *

- Appears in various contexts:
  - Automata theory *
  - Nondeterministic programming (Dijkstra’s guarded commands, Logic Programming, LSC, etc.)
  - Concurrent and asynchronous systems
  - ...

- In the curriculum:
  - CC2001: Elective unit on automata theory
  - CC2013: “Given the vastly increased importance of parallel and distributed computing...identify essential concepts...promote those topics to the core.”

* D. Scott and M. Rabin (1959). "Finite Automata and Their Decision Problems".
Teaching ND

- ND is a complex concept
  - Cognitively, but also psychologically:
    - Dijkstra: “I myself had to overcome a considerable mental resistance before I found myself willing to consider non-deterministic programs seriously”
  - Known to be hard to teach and learn
- What kind of ND is usually taught?
  - Automata theory, Class NP…
    - Hige level abstraction, existential semantics, mathematical context…
- What we suggest:
  - Teach the kind of ND that appears in non-deterministic programming (operative ND)
  - Teach it in the context of a programming course
  - Using a nondeterministic language such as LSC
**Live Sequence Charts (LSC*)**

- **Visual**
- **Scenario-based**

* W. Damm and D. Harel (2001). “LSCs: Breathing Life into Message Sequence Charts”.

---

---
ND in LSC – SOME EXAMPLES

- Nondeterministic order:
  - within a chart
  - between charts

1. One chart activates the other
2. The two proceed simultaneously
ND in LSC – SOME EXAMPLES

- **Must vs. May** modality
RESEARCH SETTING

- 45 hours semestrial course on LSC
- 12th grade high-school students majoring in computer science
- Course structure: Theory + lab + final project
- Assessment: Exams + final projects
ASSESSING STUDENTS’ UNDERSTANDING

- A combined Bloom/SOLO taxonomy*:

<table>
<thead>
<tr>
<th></th>
<th>Unistructural</th>
<th>Multistructural</th>
<th>Relational</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applying</td>
<td>Quantitative + Qualitative</td>
<td>Quantitative + Qualitative</td>
<td>Qualitative</td>
</tr>
<tr>
<td>Creating</td>
<td>Quantitative + Qualitative</td>
<td>Qualitative</td>
<td>Qualitative</td>
</tr>
</tbody>
</table>

* Based on: Meerbaum-Salant et al. (2010). “Learning computer science concepts with scratch”.
The category of Applying-Multistructural:

- **Applying**: The ability to mentally simulate pieces of code that contain a non-deterministic element.
- **Multistructural**: A perspective that incorporate multiple LSC charts.
- Example of a question that falls into this category:

```
1. Write a possible execution order for the following charts.
2. Is it the only possible order? If not, write another possible order.
```
Findings (Example of)

- Quantitative:

<table>
<thead>
<tr>
<th></th>
<th>Unistructural</th>
<th>Multistructural</th>
<th>Relational</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applying</td>
<td>83%, N=26</td>
<td>76%, N = 18</td>
<td></td>
</tr>
<tr>
<td>Creating</td>
<td>100%, N=10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Qualitative (not shown here)

- Summary of findings:
  - Comprehend systems that contain ND
  - Create systems that contain ND
CONCLUSIONS

- High-school students can reach a significant understanding of operative ND, when the concept is introduced in:
  - The context of a hands-on programming course
  - Using a nondeterministic language like LSC

- Implementation:
  - Can be done by integrating a section on nondeterministic programming into an advanced high-school course
  - Using LSC achieves additional educational goals, such as introducing a new programming paradigm and developing abstract thinking

- Open issues:
  - The effect of learning operative ND on the learning of the kind of ND that appears in automata theory