Reshaping Unplugged Computer Science Workshops for Primary School Education

Martina Landman(✉), Sophie Rain, Laura Kovács, and Gerald Futschek

TU Wien, Vienna, Austria
{martina.landman,sophie.rain,laura.kovacs, gerald.futschek}@tuwien.ac.at

Abstract. Through meticulous analysis and adaptation, we reshaped unplugged computer science activities to align with the developmental needs and capabilities of primary school children. Our approach focuses on distilling the essence of computer science topics while tailoring their content and delivery methods to suit the younger audience. We describe our efforts and report on our experiences implementing our framework for eight primary school classes, turning our unplugged computer science workshops for secondary school classes into an educational playground for 192 primary school children. Our work contributes to the general societal mission of supporting more and more children to become interested in STEM, ensuring that our technological future is as diverse as possible.

Keywords: Primary Education · CS Unplugged · Girls Empowerment

1 Introduction

In today's highly autonomous world, Computer Science (CS) has become a subject everyone should learn about. Within the work presented in this paper, we advocate that computer science should be accessible to everyone, starting with our society's youngest generation. We therefore designed an unplugged CS workshop framework to ease the integration of CS topics within primary school education. We focus on adjusting a set of CS unplugged activities [1] adapted by us, initially designed for an informal learning environment of a 90 min workshop targeting early secondary school students, into engaging activities suitable for primary school students.

With the aim of reshaping unplugged CS activities, we carefully selected three CS topics, forming the backbone of our unplugged CS workshops for primary schools. Through the engaging scenarios of (Task 1) sorting, (Task 2) searching, and (Task 3) planning, we provide young children the opportunity to explore and experiment with various CS concepts playfully. This way, we empower primary school children to create their own solutions (i.e. algorithms) without compromising the conveyed CS content. In particular, we address (Task 1) the role of fast-moving (secret) data points within sorting algorithms; (Task 2) the concept of divide and conquer algorithms during searching/playing cards; and (Task 3) the need for optimal planning while solving path-finding puzzles, allowing us to ultimately expose children to basic ideas of programming (algorithms, programming languages).

To make our workshop tasks as accessible as possible, especially for first-graders, our CS activities are developed without requiring reading or writing skills (Sect. 4). Further, we carefully integrated age-appropriate content with CS tasks (Sect. 5) and paid special attention to the children's social-emotional development during our unplugged CS tasks (Sect. 6).

We performed eight iterations of our unplugged CS workshops for primary schools. In each such iteration, one primary school class of 22–27 students (ages 8–9) visited our workshop at our institute, as this gives the children the opportunity to experience a scientific institution. In total, 192 primary school children have participated in our CS workshops so far (Sect. 6). Based on our empirical evaluation and the teachers'/students' feedback, we conclude that reshaping CS workshop tasks for primary schools is a worthy and successful effort, which we aim to further strengthen at TU Wien.

2 Background and Related Work

2.1 In-House Outreach Programmes

At TU Wien, we have already conducted outreach programmes for teaching CS concepts to secondary school students. Within this existing framework, we offer (i) online programming courses to ease individual learning; (ii) in-class programming courses at interested schools, and (iii) "unplugged" CS workshops at our university. Each event from (i)-(iii) is conducted by the authors, based on our experience as university students/assistants/professors. The work described in this paper carefully reshapes item (iii) and proposes "unplugged" CS workshops as a supplementary educational programme for primary schools.

2.2 Related Approaches and Initiatives

A recent report by the Brookings Institution [9] points out the need for computing education "early on", that is, as early as primary and secondary schools, and identifies the key challenges for doing so. In her blog, Sue Sentance [7] highlights the main lessons learned from [9], importantly to "start early" with CS education. While [9] finds that there is a worldwide interest in the early promotion of CS programmes, it also stresses the need to catch up with up-to-date CS developments and improve the overall integration of CS in various curricula.

In addition to coding and programming initiatives[1], there are creative approaches to teaching CS without a computer, most notably CS Unplugged [1], Abenteuer Informatik [5] and Bebras Challenge[2]. Our own outreach programs are mainly based on tasks inspired by the former two programs. We also

[1] e.g. https://code.org/.
[2] e.g. https://bebras.org/.

extend these initiatives with further ideas, such as including tasks involving simple robotics. In particular, we use Ozobots[3] as our main robotics device because of their ability to be programmed through colour codes. Ozobots are thus immediately accessible to students of all ages and allow their programming via "paper and pencil", which fits nicely with our unplugged CS activities.

It has been proposed that curricula and long-term initiatives are most effective when adhering to the concept of a spiral curriculum [3]. Extending and revising our CS tasks to younger students, as proposed within our current paper, should thereby ideally offer an iteration of our existing program for a younger age group, allowing them to be exposed repeatedly to the same core ideas, each time in a format that is most appropriate for their respective age and stage. Two of the most significant advantages of such an approach are increased retention through repetition, as well as the possibility to continually increase task complexity and therefore reach technical depth.

Instructional design models support structured and traceable developments of teaching materials and tasks. Among others, the ADDIE model [2] implements such a structured process within 5 main phases "analysis", "design", "development", "implementation" and "evaluation".

To ensure high-quality (unplugged) CS workshops, design-based research (DBR) [6] can be used, characterized by the cyclical repetition of a learning intervention to improve it. The Action Research Model [8], consisting again of cyclical, repetitive phases in which the teacher herself reflects on and evaluates her action, is particularly suitable. There are different interpretations of the phases, such as "plan", "act", "observe" and "reflect" [4], which provide a framework for collecting data from one's own experience of delivering a learning intervention and building on this to improve.

3 Methodology

In this section we present the workflow for reshaping our existing unplugged CS workshop for secondary schools, allowing us to introduce unplugged CS workshops for primary school children (ages 6–10, school grades 1–4). Within this workflow, we systematically answer the following research question:

RQ: How can secondary school tasks be simplified in an informal learning setting (workshop) to be used in primary school level?

We address this RQ by analyzing the difficulties and needs of the target group; revising and designing new learning tasks; and evaluating and improving our learning tasks based on first-hand experience collected within eight workshops implementation of RQ, allowing us to attract 192 primary school children (ages 8–9). Doing so, we rely upon the ADDIE instructional design model [2]. We restructure learning materials from secondary schools and integrate our solutions

[3] e.g. https://ozobot.com/.

to the RQ within the five main ADDIE phases, as summarized in Fig. 1 and discussed next.

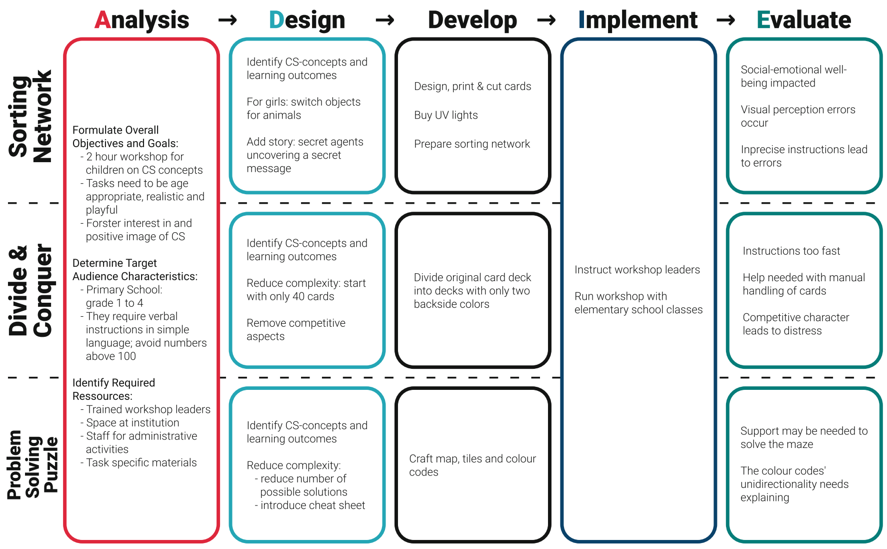
Analysis → Design → Develop → Implement → Evaluate
Sorting Network
Divide & Conquer
Problem Solving Puzzle
Formulate Overall Objectives and Goals:
- 2 hour workshop for children on CS concepts
- Tasks need to be age appropriate, realistic and playful
- Forster interest in and positive image of CS
Determine Target Audience Characteristics:
- Primary School: grade 1 to 4
- They require verbal instructions in simple language; avoid numbers above 100
Identify Required Ressources:
- Trained workshop leaders
- Space at institution
- Staff for administrative activities
- Task specific materials
Identify CS-concepts and learning outcomes
For girls: switch objects for animals
Add story: secret agents uncovering a secret message
Design, print & cut cards
Buy UV lights
Prepare sorting network
Social-emotional well-being impacted
Visual perception errors occur
Inprecise instructions lead to errors
Identify CS-concepts and learning outcomes
Reduce complexity: start with only 40 cards
Remove competitive aspects
Divide original card deck into decks with only two backside colors
Instruct workshop leaders
Run workshop with elementary school classes
Instructions too fast
Help needed with manual handling of cards
Competitive character leads to distress
Identify CS-concepts and learning outcomes
Reduce complexity:
- reduce number of possible solutions
- introduce cheat sheet
Craft map, tiles and colour codes
Support may be needed to solve the maze
The colour codes' unidirectionality needs explaining

Fig. 1. Summary of the main steps within each ADDIE phase of our approach.

Analysis. Through close observation of the previous practical implementation of our CS tasks for secondary school classes, we identified learning difficulties. We analyzed these challenges by focusing on our new target group of primary school children (Sect. 4).

Design. We identified and revised CS tasks to be presented in our unplugged CS workshops. Each CS task is motivated and related towards complementing primary school education with CS topics. Since we aim for a diverse environment, we considered the findings of [10] in the design of our CS tasks. For example, as [10] argues that girls are more interested in a problem framed about humans/animals instead of using objects, whereas boys are indifferent to such problem-framing choices. We therefore designed our CS tasks using visual material involving animals, thus favouring girls-friendly content while not harming boys' perceptions (Sect. 5). The conveyed CS topics within our workshops are abstracted on a high level to make the concepts accessible to primary school children.

Development. We formalized the learning objectives for each CS task and created a prototype of each CS unplugged task. We make this prototype accessible also to future workshop leaders (Sect. 5).

Implementation. We instruct team members and university students to serve as workshop leaders and promote the workshops to schools.

Evaluation. After each workshop, we conduct a joint group reflection inspired by the reflection phase of action research (Sect. 6). This information is used to continuously improve our CS tasks and our unplugged CS workshop setting.

4 Task Analysis and Evolution

We now detail the evolution of our tasks, originally introduced for secondary schools and reshaped for primary schools (see Sect. 5). We provide the rationale for our changes while relying on the ADDIE model [2].

Within our first ADDIE phase of "Analysis", the following three main points have been considered: analyzing the characteristics of our target group, formalizing our learning objectives, and identifying required resources for implementation. With the aim of reshaping existing unplugged CS workshops from the secondary level to the primary school level, we set the following aspects.

Overall Goal. We offer unplugged CS workshops, with each workshop instance taking place at our university for a duration of two hours. Our overall goal is for young children in primary schools to learn core computer science concepts. Looking through the possible tasks and concepts of our existing unplugged CS workshops for secondary schools, we decided on choosing the following three workshop tasks for primary schools: *(Task 1)* using sorting networks; *(Task 2)* sort and search via the CS principle of Divide and Conquer concept; and *(Task 3)* planning via solving puzzles based on computational thinking and experimenting using Ozobots. We believe these tasks can be made age-appropriate, realistic, and playful (see Sect. 5), sparkling genuine interest in computer science at a very young age. We also paid special attention to making these tasks as diverse as possible, in particular friendly to young girls.

Target Audience Characteristics. Our workshops target primary school children of any age without any prior knowledge. We therefore provide easy-to-adjust workshop tasks, for example, by using small number ranges of 1–9 for first grade children vs larger number ranges of 1–100 for third graders in Task 1. Similarly, we initially use card stacks of 4×10 cards in Task 2, instead of starting immediately with all $4 \times 4 \times 10$ cards. Whereas in Task 3, we reduced the possibilities of different paths in the map added additional tiles for path planning and introduced a "cheat" sheet that children could use to identify colour codes.

Similarly, reading comprehension should not be a hard requirement in our primary school workshops. We used visual material, such as different-sized animals for sorting in Task 1 instead. To make our tasks more playful for primary school children, we use secret messages and UV light-pens in Task 1 and obstacles as animals/ponds/forests in Task 3.

Finally, instead of using written instructions (as is the case for secondary schools), we verbally explain our workshop tasks to primary school children, using simple, age-appropriate terms.

Resources. To run our workshops at TU Wien, skilled workshop leaders are needed. Therefore, we dedicated ourselves to training computer science undergraduate/graduate students, helping us as workshop leaders. This way the school teachers could focus on the children's social-emotional well-being. We also implemented adequate framework conditions for our workshops, including reserving suitable rooms; establishing and maintaining contact with school classes; and organizing task-specific material and equipment.

5 Task Design and Development

Within the design of our CS workshops and their respective tasks, we placed special emphasis on creating a suitable learning environment for young children, where the framework focuses on embedding the learning content within an age-appropriate scene. It is essential to assist children in adjusting to their new surroundings; we therefore relate our university setting to the daily school life scenario of students, while drawing similarities between primary school and university activities.

For each CS task, we considered the following design process. First, we **(M)** motivate the task by explaining why the task is relevant to daily life. Second, we let children **(E)** experiment with the task, develop different solutions, and implement their own strategies (i.e. algorithms). Finally, we **(R)** relate the task to CS topics by providing them answers on how such CS topics (and solutions) are used on a daily basis. In conclusion of this design process, we developed an effective task prototype and set the **learning outcomes** of our workshop tasks.

5.1 Task 1 – Sorting Network

(M) Motivate. This task is inspired by the unplugged CS task of sorting networks[4], enabling children to sort items by following a small number of instructions. We motivate the use of sorting networks, and in general sorting, with real-life sorting scenarios of clothes (by size), videos (by length), and groceries (by price).

(E) Experiment. We use two large sorting networks taped to the floor, supporting six children to sort items according to some predefined property. Six children *start* sorting by one child standing in one of the starting nodes (represented as squares) and randomly choosing, for example, one number card from a collection (numbers 1–100[5]). The chosen cards are unsorted, with the cards becoming sorted upon the children *finish* traversing the sorting network and arriving on the other side of the room in correctly sorted order in the designated boxes on the floor.

For traversing the sorting network, the children *initialize* the sorting process by moving from their initial position along lines, arriving at the first node of the network, which is represented as an ellipse on the floor where two children fit in.

[4] https://www.csunplugged.org/en/topics/sorting-networks/reinforcing-numeracy-through-a-sorting-network-junior/.

[5] For younger children, we use numbers 1–9 or simple geometric figures (e.g. triangles) of different sizes.

Within the ellipse node, *decisions* between two children are made by comparing the values of the property of interest: for example, when using number cards, the child with the smaller (resp. bigger) number moves along a red (resp. blue) line into the next ellipse node. Such decisions between pairs of children are *repeated until* each child reaches their (final) spot, resulting in a sequence of sorted numbers. In such a sorting process, children thus implement four instructions (start, initialize, decide&repeat, stop) within the sorting network; moreover, children are encouraged to experiment with "runs" of the sorting network, trying to become faster while correctly executing the sorting instructions.

We further experiment with sorting secret cards, each card hides a secret key that is only visible when a UV light shines on the cards. Here, we used animals in different sizes on each card. The children sort secret cards as before, yet, to make "decisions" in the black ellipse nodes, the children need to wait for a UV light pen to process and compare the secret keys. While experimenting with this, children realize that more UV light pens significantly speed up the sorting process by parallelizing sorting instead of waiting for one single central pen (processor).

(R) Relate. Children are introduced to sorting networks and algorithms, enabling them to sort items parallel, similarly to sorting engines on the web operate. Within this process, children learn about (fast) algorithms, allowing our workshop task to explain foundation concepts from the CS areas of Algorithms and Data Structures, as well as Parallel Computing.

Learning Outcomes. As the underlying CS concepts of this task are parallelization and rigorous algorithmic execution, we set the following learning objectives, as learners can:

- understand that the more people make comparison decisions simultaneously, the faster the sorting network/algorithm will (correctly) terminate;
- solve the algorithmic aspect of sorting networks by following the given instructions on traversing sorting networks.

5.2 Task 2 – Divide and Conquer

(M) Motivate. During this task, we motivate the CS principle of Divide and Conquer by instrumenting children to playfully develop their own set of instructions that yield sorting algorithms for a shuffled stack of cards. We exemplify the need for such sorting algorithms for finding items in large collections, such as identifying that a particular book with 2–3 characteristics is missing in large library collections.

(E) Experiment. We split children into groups of five–seven participants, urging collaborative teamwork. We use Ligretto cards[6] as task workhorses since these cards exhibit easily visible, three different properties: each card has one of four colours on the card back and a number between 1–10 on its front side. A full set of Ligretto cards thus consists of $4 \times 4 \times 10 = 160$ cards.

[6] https://www.schmidtspiele.de/detail/product/ligretto-blue.html.

Each group of children receives only one shuffled stack of Ligretto cards[7], with one side of the cards being of the same (front) colour. The task is to sort these cards based on (back) colour and numbers as quickly as possible. After visually illustrating the desired sorting outcome (e.g. red cards from 1–10, followed by blue cards from 1–10, etc.), we give no further instructions on how sorting should be achieved; yet, if children struggle with structuring their ideas, further hints may be given by workshop leaders. We encourage children to discuss their ideas together and have everyone's role determined before the sorting process starts. After the groups complete sorting their shuffled stack of cards, each group's solution is discussed, with the ultimate goal that groups learn from other groups' solutions. Within this discussion, we highlight the key steps children used in their solutions: e.g. divided the big stack of cards into smaller ones, sort the smaller stacks, and then merge sorted card stacks; in other words, children intuitively implemented the CS principle of Divide and Conquer.

A second round of card sorting is started, without changing groups, but only shuffling cards. The revised sorting solutions are again discussed, again focusing on whether sorting performances have improved compared to the previous round. In case of significant improvements, e.g. sorting times fit within a short time window (1–2 min), another round of card sorting is initiated with increased difficulty: a second set of 4×10 shuffled cards of another front-side colour are additionally used, asking children to sort based on two colours and numbers 1–10. Children are likely to adapt their previous solutions to fit the new challenge. Nevertheless, Divide and Conquer remained the crux of their solutions.

Finally, we also initiate a card sorting process where workshop leaders silently remove one card from a complete stack. When asking children to sort the shuffled stack of cards, children identify in a relatively short amount of time that one card is missing and precisely characterize the colour and number of the missing card. However, again, all this while playfully executing their own solutions based on Divide and Conquer.

(R) Relate. When sorting large data sets or searching for an item in a big database (i.e. Ligretto cards), children naturally work collaboratively in order to find their fastest solutions: divide a given problem into subproblems and merge the sorted subproblems into the sorted version of the given problem. Such a structured approach is the basis of Divide and Conquer, and hence children are naturally introduced to key concepts within the CS areas of Distributed Computing and Resource Optimization.

Learning Outcomes. While parallelization is a practical application of Divide and Conquer, its key benefit comes from dividing an intractable problem into more manageable parts. The learning outcomes of this task are therefore that learners can:

– identify/recognize subtasks;
– divide a task into subtasks and distribute the solving workload among group participants;

[7] we only use 4×10 cards per group initially.

– describe a (Divide and Conquer) strategy to sort a shuffled stack of cards.

5.3 Task 3 – Planning in Puzzle Solving

(M) Motivate. During this task, children create (optimal) paths within a (geographic) map given limited resources. We motivate the need for planning via route finding from school to home, or in general for other routes, frequently used by automotive navigation systems. To showcase the impact of human/children intelligence vs computer intelligence, students also discover how to steer a small line-follower robot using colour codes within the path created by the children.

(E) Experiment. We split the whole group into groups of three–five participants. Each group receives an A2 sheet with a printed tile grid, as shown in Fig. 2(a). Each printed tile is either empty, contains a picture of an object, or has a partial path, i.e. a black line drawn from one side to another. The pictures used within the tile include a school, a house (home), and the obstacles of a forest, a pond, and a tiger. Children are further given a limited set of additional path tiles with either a straight or a curved path. Each group starts with finding a path from the school to the house while avoiding the a priori given obstacles. The children may use only the given set of straight/curved path tiles, as shown in Fig. 2(b). Once the students have found a suitable path, they are given an Ozobot placed at the school. This Ozobot will follow the developed path. Hence, children already know what the Ozobot should do: do exactly what the children's solutions tell them to do and thus reach the house (home).

In a further step, a new tile grid with intersections is introduced. When encountering an intersection, an Ozobot may randomly choose which direction to take. To have full control over the Ozobot, children are encouraged to instrument the Ozobot by laying down a colour code (over the path) right before the intersection, determining the specific direction to be taken. In other words, children "program" the Ozobot to follow their chosen path.

Once the children find solutions, the workshop instructors visualize possible solutions with live coding using the block-programming language "Scratch"[8]. Children are thus introduced to the art of programming only after they know what a computer scientist should do: understand and create a path-finding solution (algorithm) just like the children did.

(R) Relate. When planning a path from home to school, children ensure that their solutions are safe, e.g. obey a priori given rules and omit forbidden/unsecure items in their planned trajectory. As such, the children are playfully introduced to the CS areas of Formal Methods and Computer Security, and in general to Artificial Intelligence.

Learning Outcomes. This task corresponds to solving a logical puzzle: finding a way through a maze/map with a limited range of tiles. The possibilities are limited and only a few ways are possible, thus connecting naturally to backtracking concepts. This task also conveys elementary knowledge about computer

[8] https://scratch.mit.edu/.

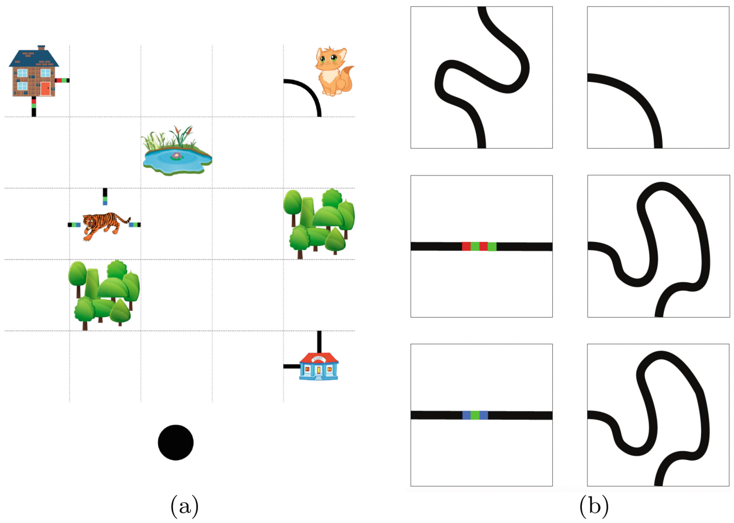

Fig. 2. (a) Finding a path from school to home, (b) using predefined tiles.

commands and languages that computer systems and machines can understand. We identified the following learning outcomes as learners can:

– plan a route given additional constraints;
– describe different possible solutions by using less/more constraints;
– understand that computer programs/machines need commands in a specific programming language (e.g. colour codes for Ozobots).

6 Task Implementation and Evaluation

For the implementation, the executing student employees were well-instructed. There were written instructions, a clear distribution of tasks, and a joint preliminary discussion. It was clearly defined in advance who is responsible for which activities during the workshop, how the tasks work, and which things they must pay special attention to when working with children.

In the final "Evaluation" phase of the ADDIE model, we conducted an action research-inspired approach through eight pilot runs of our unplugged CS workshops for primary schools with reflection sessions after each run. Altogether, we reached 192 primary school students. Each workshop instance was evaluated through the cyclical phases of "plan", "act", "observe", and "reflect" [4], allowing us to revise further and improve workshop materials and the overall workshop process, as detailed below.

Evaluation of Task 1 – Sorting Network. Based on the evaluation of our CS workshops so far, we made changes in Task 1 targeting the followings:

- *Social-emotional well-being:* As there are only 6 input nodes in the sorting network , only 6 students can experiment per network. Since class sizes may vary, with our original setup (using two sorted networks) not all students got to experiment the same number of times. Originally, only three runs of the sorting network were planned (first run with numbers, second run with invisible animal images and one UV Light "comparator", third run with three "comparators"). This initial plan resulted in having a few children that experimented with sorting networks only once, while other children twice. Due to the children's emotional reactions, we extended Task 1 with a further run with numbers so that each child tries out our sorting network at least twice. Empty inputs nodes/squares are filled with teachers/volunteering children.

- *Visual perception error:* One observed difficulty students had was reading similar numbers (e.g. 6 and 9). In the joint reflection sessions of our workshops, this topic came up several times and we kept sorting out numbers that could be misunderstood; for example, eliminating 6, 9, 66, and 99.
- *Instructional error:* We observed that children might tend to go in the wrong direction of the sorting network: though the instruction is "follow the blue line if you have the bigger number/animal", there is also a blue line leading back to the start . In particular, we observed chaotic behaviour when the starting phase of sorting was overcrowded. In our workshop reflection rounds, we agreed to pay more attention to the fact that the children go to the starting squares; we also included this as a (start) instruction for our task.

Evaluation of Task 2 – Divide and Conquer. Similarly to the reflection rounds and observations made withing Task 1, the following changes have been made on Task 2 to address:

- *Instructional error:* We spend more time explaining the task while providing an a priori fixed set of hints on-demand.
- *Social-emotional well-being:* We playfully motivate competitive efforts while measuring the times children spent on sorting cards, emphasizing that groups do not compete against each other but essentially only improve their own times (due to different solutions they use).

Evaluation of Task 3 – Planning in Puzzle Solving. Similarly to Tasks 1–2, the following changes on Task 3 have been implemented to prevent:

- *Procedural errors:* We actively interact with children and provide hints for using the tiles/planning the path.
- *Conceptual misunderstandings:* Our Ozobots need to read (i.e. drive over) a colour code for one specific direction (left/right/straight). The tiles indicate such directions by arrows next to the colour code. For example, a tile with a blue-black-red sticker is the command "go straight ahead"; yet, if the Ozobot reads this code backward, it chooses a random direction. As the sequence of colors in our tiles are not very self-explanatory, we allocate extra time so children experiment more with coloured tiles.

7 Conclusions

We showed how to reshape unplugged CS tasks from secondary to primary schools, providing an age-appropriate and diverse CS workshop environment. We rely on the design-based research approach advocated by the ADDIE structured workflow model and continuously adjust our learning framework by reflecting on our CS workshop experiences and performances.

Through our unplugged CS workshops reshaped for primary schools, we motivate and relate CS topics from and to daily life and encourage children to experiment with their own solutions. Based on our task evaluation so far, we believe that unplugged CS activities can (and should) nicely complement primary school education. We aim for further developments of our CS task by exploring the participant's background and adjusting workshop instructions/algorithmic solving challenges accordingly.

Acknowledgements. We thank Philipp Prinzinger for his prior initiatives related to our work. We are grateful for the support we received from Anja Petković Komel, Katalin Fazekas, Lukas Lehner, and Svetlana Unkovic (TU Wien). Moreover, we thank our volunteering workshop holders from the eduLAB service unit and the FORSYTE research unit of the TU Wien. We finally acknowledge partial funding from the ERC CoG ARTIST 101002685, the Let's Empower Austria - LEA project 1IP-0006, the TU Wien SecInt Doctoral College, and the FWF SFB project SpyCoDe F8504.

Appendix

Sorting Network

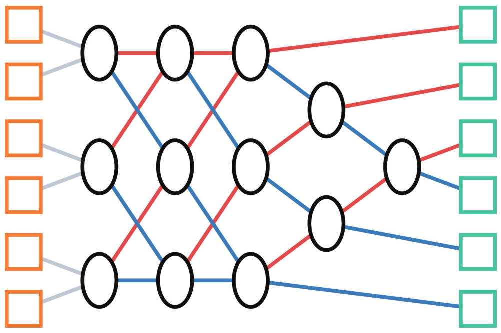

Fig. 3. Sorting network used in task 1 with input/output (orange/green squares) from the CS unplugged task

References

1. Bell, T., Witten, I., Fellows, M.: CS Unplugged: an enrichment and extension programme for primary-aged students (2015). https://www.csunplugged.org/
2. Branch, R.M.: Instructional Design: The ADDIE Approach, 1st edn. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-09506-6
3. Bruner, J.S.: The Process of Education. Harvard University Press, Cambridge (2009)
4. Dickens, L., Watkins, K.: Action research: rethinking lewin. Manag. Learn. **30**(2), 127–140 (1999). https://doi.org/10.1177/1350507699302002
5. Gallenbacher, J.: Abenteuer Informatik (2008). https://www.abenteuer-informatik.de/
6. Easterday, M.W., Lewis, D.R., Gerber, E.M.: Design-based research process: problems, phases, and applications. In: Proceedings of International Conference of the Learning Sciences, ICLS 1(January), pp. 317–324 (2014). https://www.scholars.northwestern.edu/en/publications/design-based-research-process-problems-phases-and-applications
7. Sentance, S.: Computer science education is a global challenge (2021). https://www.raspberrypi.org/blog/brookings-report-global-computer-science-education-policy/
8. Stephen, M.: Corey: action research in education. J. Educ. Res. **47**(5), 375–380 (1954). https://doi.org/10.1080/00220671.1954.10882121
9. Vegas, E., Hansen, M., Fowler, B.: Building skills for life: How to expand and improve computer science education around the world. Brookings (2021). https://www.brookings.edu/essay/building-skills-for-life-how-to-expand-and-improve-computer-science-education-around-the-world/
10. Vorvoreanu, M., Zhang, L., Huang, Y.H., Hilderbrand, C., Steine-Hanson, Z., Burnett, M.: From gender biases to gender-inclusive design: an empirical investigation. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. CHI 2019, New York, NY, USA, pp. 1–14. Association for Computing Machinery (2019). https://doi.org/10.1145/3290605.3300283