Linguistic Pedagogical Approaches to Transfer in Computer Science

Rosalind Owen
rowen@wested.org
WestEd
San Francisco, CA, USA

ABSTRACT

More students are encountering computer science at multiple grade levels and so are learning more than one programming language. There is an ever-growing body of research describing how students transfer knowledge from one language to another. Current research shows that transfer helps students learn a second programming language in the interim and improves attitudes and retention of students in computer science [2, 12]. While novice programmers generally struggle with the same concepts [1, 12], the exact difficulties and benefits of the transition to a second programming language differ depending on the programming languages the student is learning. In order to best serve students of different backgrounds and maintain their interest in the subject, the details of transfer for different programming language combinations must be understood. This poster surveys and analyzes the current research on transfer and provides a summary of the variety of challenges and advantages students face in learning a second programming language. Additionally, interdisciplinary pedagogical approaches are discussed, integrating perspectives from applied linguistics as novel solutions to the specific issues faced in programming language transfer.

ACM Reference Format:
Rosalind Owen. 2024. Linguistic Pedagogical Approaches to Transfer in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3626253.3635566

1 BACKGROUND AND OVERVIEW

As more students take computer science courses at different points in their education, they are likely to encounter more than one programming language. Recent research has investigated analogical transfer, or how students use shared relationships to transfer knowledge from one context to another, in this case, from one programming language to another [9]. When students make connections between surface similarities that reflect real similarities, this is called positive transfer. Negative transfer occurs when these surface similarities lead to incorrect assumptions. In order to better support students in their transition from one language to another, researchers are describing how students manage a new programming language based on their previous programming language, along with various interventions meant to support positive transfer. The specific challenges students face while learning a new programming language vary depending on their prior experience [7], as do their strengths when transferring knowledge from their first programming language [11, 12].

Applied linguistics as a field has much research on the subject of transfer, specifically from one natural language to another. Both computer science education and natural language learning share the issue of how the syntax and semantics of one language influence the acquisition and use of another language. As is the case with programming languages, natural languages often have semantic distinctions or syntactic structures not present in another language. Researchers in applied linguistics have developed pedagogical solutions to help students overcome obstacles in language acquisition and leverage previous knowledge to understand the intricacies of each language's syntax and semantics. Interventions for avoiding negative semantic transfer have been studied at the vocabulary [8] and sentence [4] levels, which can be applied to learning keywords and syntax in computer science classrooms. Additionally, contextualizing languages and raising awareness of their characteristics can promote critical thinking about how to use each language and how their distinctions may be useful [5]. This approach can be applied to programming languages and be used to empower students in their own learning.

This poster reviews studies of transfer in computer science education and presents frequently encountered issues. Approaches from natural language learning are presented as novel solutions to common problems with second programming language learning in the computer science classroom.

2 TRANSFER BY PROGRAMMING LANGUAGE

Each programming language pairing is compared in a table that summarizes the instances of positive and negative transfer. The peculiarities of each pairing are explored, as well as generalizations based on shared characteristics between programming languages. An example follows below.

Students with experience in Scratch who are learning Python struggle with understanding which characteristics of loops apply to their prior language and which apply to the new language [10]. These students use conditional statements with the opposite condition than needed for while loops, an instance of negative transfer from Scratch repeat loops. Additionally, they sometimes handle the end of a while loop with an if statement to break the loop, rather than relying on the conditional statement, which is likely a negative transfer from Scratch forever loops.

Students also struggle to grasp concepts related to Python-specific traits [10]. There is often confusion about type matching, with students using operators on variables of different types, and type

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA

ACM ISBN 979-8-4007-0424-6/24/03.
https://doi.org/10.1145/3626253.3635566

casting, with students recasting data types with still the wrong type. Considering Scratch is a weakly typed language and Python is strongly typed, students may need more information to grasp this new concept. Students also did not understand how to initialize an empty list, instead initializing a list with one item in it and then later clearing it. This misconception likely stems from the fact that lists are not explicitly initialized in Scratch.

Transitions from Alice, Snap!, and Scratch to Java, as well as Python to Java, are also included in the poster.

3 APPROACHES FROM NATURAL LANGUAGE LEARNING

This section reviews pedagogical approaches to semantic transfer in natural languages, including perspectives from translanguaging [5] and cognitive linguistics [4]. The main contribution of this poster is the application of these approaches to the transfer issues observed in different programming language pairings and how these approaches can help computer science instructors.

Pedagogical translanguaging suggests that "intentional instructional strategies that integrate two or more languages" should be used in a multilingual classroom [3]. Pedagogical translanguaging can be incorporated into the classroom by raising metalinguistic awareness, or awareness of a language system (e.g., syntactic structures or sound patterns) [5]. In natural languages, transfer is often seen at the morphological level. Negative transfer occurs in the form of "false friends," or words that mean different things but look similar. For instance, students learning Spanish used the Spanish "carpeta" (English: "folder") as if it meant "carpet" in English (Spanish: "alfombra") [6]. However, students with more metalinguistic awareness were able to use their knowledge of the morphology of Spanish to make complex changes to the suffixes and roots of known words to guess new Spanish words in their writing (e.g., English "stressed" to Spanish "estresado") [6].

In the context of a computer science classroom, students' prior knowledge of any programming language would be bolstered by instruction that raises metalinguistic awareness. Their prior coding experience would be an asset that they could combine with metalinguistic awareness to approach novel coding situations. For example, students learning Python with experience in Scratch often use an if-statement to break from a conditional loop [10]. This confusion may be akin to the issue of "false friends." Students see syntax in Python related to an infinite loop and connect it with a forever loop in Scratch, not fully aware of the semantic importance of the conditional statement. With pedagogical translanguaging, the distinctions between finite and infinite loops, as well as conditional and unconditional loops, would be explained, so that students have the vocabulary with which to think about different kinds of loops. With more explicit conceptual knowledge about the forms that loops can take, students could approach new keywords and syntax in a more informed and analytical manner.

In the poster, two more methods, transcodification and input manipulation, are applied to other examples of transfer in computer science. In transcodification activities, students are shown equivalent constructions in two languages, and their literal and figurative meanings are discussed to explore subtle semantic differences [4]. For instance, students could compare pairs of equivalent loops in two programming languages, using flow-charts or diagrams as references to discuss semantic differences. Input manipulation also addresses confusion around the semantics of a particular structure, giving students positive evidence in context for the correct interpretation [4]. While a new control structure is being introduced, students could be shown code with key words highlighted, i.e. syntax highlighting, to illustrate the crucial differences between the languages and provide positive examples of the correct, contextualized use of the structure. Example applications of both of these methods are included in the poster.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2201209. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Amjad Altadmri and Neil C.C. Brown. 2015. 37 million compilations. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015). https://doi.org/10.1145/2676723.2677258

[2] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From scratch to "real" programming. *ACM Transactions on Computing Education* 14, 4 (2015), 1–15. https://doi.org/10.1145/2677087

[3] Jasone Cenoz and Durk Gorter. 2020. Teaching English through pedagogical translanguaging. *World Englishes* 39, 2 (2020), 300–311.

[4] Paolo Della Putta. 2016. Do we also need to unlearn constructions? the case of constructional negative transfer from Spanish to Italian and its pedagogical implications. *Applied Construction Grammar* (2016), 237–268. https://doi.org/10.1515/9783110458268-010

[5] Carles Fuster. 2022. Lexical transfer as a resource in pedagogical translanguaging. *International Journal of Multilingualism* (2022), 1–21. https://doi.org/10.1080/14790718.2022.2048836

[6] Carles Fuster. 2022. *Lexical transfer in pedagogical translanguaging: Exploring intentionality in multilingual learners of Spanish.* Ph. D. Dissertation. Department of Education, Stockholm University.

[7] Ryan Garlick and Ebru Celikel Cankaya. 2010. Using Alice in CS1. *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (2010). https://doi.org/10.1145/1822090.1822138

[8] Nan Jiang. 2004. Semantic Transfer and its implications for vocabulary teaching in a second language. *The Modern Language Journal* 88, 3 (2004), 416–432. https://doi.org/10.1111/j.0026-7902.2004.00238.x

[9] Yvonne Kao, Bryan Matlen, and David Weintrop. 2022. From one language to the next: Applications of analogical transfer for programming education. *ACM Transactions on Computing Education* 22, 4 (2022), 1–21. https://doi.org/10.1145/3487051

[10] Majeed Kazemitabaar, Viktar Chyhir, David Weintrop, and Tovi Grossman. 2023. Scaffolding Progress: How Structured Editors Shape Novice Errors When Transitioning from Blocks to Text. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 556–562. https://doi.org/10.1145/3545945.3569723

[11] David Weintrop, Connor Bain, and Uri Wilensky. 2017. Blocking Progress? Transitioning from Block-based to Text-based Programming. In *Proceedings of the American Educational Research Association*. 1–8.

[12] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers amp;amp; Education* 142 (2019), 103646. https://doi.org/10.1016/j.compedu.2019.103646

Linguistic Pedagogical Approaches to Transfer in Computer Science

What can CS educators learn from the second (and third) language acquisition?

Rosalind Owen, rowen@wested.org

WestEd
WestEd.org

Background

As more students take computer science courses at different points in their education, they are likely to encounter more than one programming language. Recent research has investigated analogical transfer, or how students use shared relationships to transfer knowledge from one context to another, in this case, from one programming language to another [8].

- *Positive transfer:* when students make connections between surface similarities that reflect real similarities.
- *Negative transfer:* when surface similarities lead students to incorrect assumptions.

Much research has been done on how students learn a new programming language based on their previous programming language, along with various interventions meant to support positive transfer. The different strengths and weaknesses students bring to a new language based on their prior knowledge is outlined in table to the right.

Learning programming languages and natural languages both entail grappling with syntax and semantics. The following sections apply approaches from applied linguistics that are used to learn second and third natural languages to the issue of transfer in programming languages.

Transcodification

In transcodification activities, students are shown equivalent constructions in two languages, and their literal and figurative meanings are discussed to explore subtle semantic differences [4].

Natural Languages

Della Putta (2016) describes the following example: when learning the verb “ir” in Spanish (conjugated in first person present tense as “voy” below), teachers use miming to show the physical displacement of one construction versus the temporal displacement of the other.

> *Voy a estudiar en la biblioteca.*
> ‘I go/am going to study in the library.’ (p. 259)
>
> *Mañana voy a estudiar química.*
> ‘Tomorrow I am going to study chemistry.’ (p. 259)

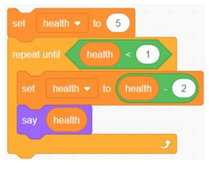
Programming Languages

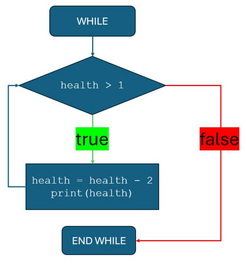
Teachers can use flowcharts to illustrate the difference in execution between while loops and repeat until loops.

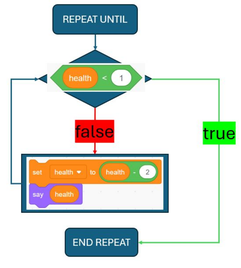
Python

```
health = 5
while health > 1:
    health = health - 2
    print(health)
```

Scratch







Students’ Strengths and Weaknesses in a Second Programming Language

Languages	Strengths	Weaknesses
Python → Java	• Semantic transfer: matching syntax, similar syntax, functions, data structures [10]	• Conceptual misunderstandings: types; objects (especially with different syntax) [10]
Blocks-based → Java	• Semantic transfer: initially acquired various concepts faster than text-based students [13] • Programming practices: less prone to common errors than text-based students [12]	• Programming practices: no difference compared to text-based students [13]
Scratch → Java	• Semantic transfer: for loops (initial confusion overcome more quickly); counters [1, 7]	• Conceptual misunderstanding: for loops (favor while/forever loops) [1, 7] • Syntax errors, missing brackets [1]
Alice → Java	• Semantic transfer: control structures (conditionals and iteration), expression evaluation [3]	• Conceptual misunderstanding: lower exam scores than pseudocode students [6]
Snap! → Java		• Conceptual misunderstanding: for loops, while loops, variables/expression evaluation, conditional statements [11]
Scratch → Python		• Syntax errors (string quotes, operators, variable assignment) [9] • Conceptual misunderstanding: type casting, type matching, conditional loops (using opposite condition in a while loop, or if statement to break a while loop) [9]

Translanguaging

Translanguaging is used in teaching by incorporating “intentional instructional strategies that integrate two or more languages” [2]. Teachers work on raising metalinguistic awareness, or awareness of a language system (e.g., syntactic structures, morphology, or sound patterns), to give students the tools to guess new words and constructions.

Natural Languages

When students develop metalinguistic awareness, they are able to make complex changes to words so that they fit the rules of the target language [5]. Fuster (2022) presents the following examples of spontaneous negative transfer and guided positive transfer (p. 14):

Negative Transfer

"**carpet**" in English (*Spanish*: "alfombra") ≠ "**carpet**-a" in Spanish (*English*: "folder")

Positive Transfer

“**stress**-ed” in English = “e-**stres**-ado” in Spanish

Programming Languages

A common instance of negative transfer: students with experience in Scratch use an if statement to break from a conditional while loop in Python, incorrectly using Scratch rules with Python syntax.

Scratch

Python (negative transfer)

```
while not (ask == "stop"):
    # do something with ask
    if ask == "stop":
        break
```

[9, p. 560]

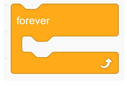
Python (correct)

```
while not (ask == "stop"):
    # do something with ask
    # loop with break automatically
    # when ask == "stop"
```

Solution: increase students’ metalinguistic awareness of “infinite loops,” explicitly discuss the vocabulary and concepts needed to analyze the semantics of this control structure.

Forever loop (Scratch)

- *Infinite* loop: will continue unless stop block is reached.



Infinite loop (Python)

- If the condition is always true, then like a forever loop, it will continue unless break is reached.

```
while True:
    # do something
```

Conditional loop (Python)

- *Conditionally* infinite loop: will continue as long as the condition following while is true. If the condition can be false, then the loop will break automatically.

Input Manipulation

Input manipulation gives students positive evidence in context for the correct semantic interpretation [4].

Natural Languages

Della Putta (2016) provides the following example used to teach students the meaning of the “ri-” prefix in Italian:

> Le strane abitudini del signor Rossi
>
> Franco Rossi è un ingegnere che ha delle strane abitudini: fa sempre tutto due volte. La mattina si alza, torna a letto e poi si ***ri***alza ancora. Poi prepara il caffè per tutta la famiglia ma, subito dopo, lo ***ri***prepara, un’altra volta! Poi va al lavoro, entra in ufficio, esce e ***ri***entra ancora. La sera, finalmente, torna a casa, saluta i figli, li ***ri***saluta e poi bacia e ***ri***bacia Anna, sua moglie. (p. 262)

Programming Languages

While a new control structure is being introduced, students could be shown code with key words and punctuation highlighted (i.e. syntax highlighting) to illustrate the crucial aspects of the syntax and provide positive examples of the correct, contextualized use of the structure. Below the syntax highlighting shows while loops in Java and their outputs, demonstrating the directionality of the conditional statements.

```
int health = 5;
while (health > 0){
    health = health - 2;
    System.out.println(health);
};
```

```
3
1
```

```
int myPoints = 0;
while (myPoints < 10){
    System.out.println("Low score");
    myPoints = myPoints + 2;
};
```

```
2
4
6
8
10
```

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2201209. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Selected References

[1] Michal Armoni, Orni Meerbaum-Salant, and Mordechai Ben-Ari. 2015. From scratch to “real” programming. ACM Transactions on Computing Education 14, 4 (2015), 1–15. https://doi.org/10.1145/2677087
[2] Jasone Cenoz and Durk Gorter. 2020. Teaching English through pedagogical translanguaging. World Englishes 39, 2 (2020), 300–311.
[3] Wanda Dann, Dennis Cosgrove, Don Slater, Dave Culyba, and Steve Cooper. Mediated transfer: Alice 3 to java. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (Raleigh, North Carolina) (SIGCSE 2012).
[4] Paolo Della Putta. 2016. Do we also need to unlearn constructions? the case of constructional negative transfer from Spanish to Italian and its pedagogical implications. Applied Construction Grammar (2016), 237–268. https://doi.org/10.1515/9783110458268-010
[5] Carles Fuster. 2022. Lexical transfer in pedagogical translanguaging: Exploring intentionality in multilingual learners of Spanish. Ph. D. Dissertation. Department of Education, Stockholm University.
[6] Ryan Garlick and Ebru Celikel Cankaya. 2010. Using Alice in CS1. Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (2010). https://doi.org/10.1145/1822090.1822138
[7] Shuchi Grover, Roy Pea, and Stephen Cooper. Designing for deeper learning in a blended computer science course for middle school students. Computer science education 25, 2 (2015), 199-237.
[8] Yvonne Kao, Bryan Matlen, and David Weintrop. 2022. From one language to the next: Applications of analogical transfer for programming education. ACM Transactions on Computing Education 22, 4 (2022), 1–21. https://doi.org/10.1145/3487051
[9] Majeed Kazemitabaar, Viktar Chyhir, David Weintrop, and Tovi Grossman. 2023. Scaffolding Progress: How Structured Editors Shape Novice Errors When Transitioning from Blocks to Text. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (Toronto ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 556–562. https://doi.org/10.1145/3545945.3569723
[10] Ethel Tshukudu and Quintin Cutts. 2020. Semantic Transfer in Programming Languages: Exploratory Study of Relative Novices. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE ’20), June 15–19, 2020, Trondheim, Norway. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387406
[11] David Weintrop, Connor Bain, and Uri Wilensky. 2017. Blocking Progress? Transitioning from Block-based to Text-based Programming. In Proceedings of the American Educational Research Association. 1–8.
[12] David Weintrop and Uri Wilensky. Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In Proceedings of the eleventh annual international conference on international computing education research (2015), 101-110.
[13] David Weintrop and Uri Wilensky. 2019. Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. Computers amp;amp; Education 142 (2019), 103646. https://doi.org/10.1016/j.compedu.2019.103646