Languages, literacies, and literate programming: Can we use the latest theories on how bilingual people learn to help us teach computational literacies?

Sara Vogel
CUNY Graduate Center

Christopher Hoadley
New York University

Ana Rebeca Castillo
New York City Department of Education

Laura Ascenzi-Moreno
CUNY Brooklyn College

Languages, literacies, and literate programming: Can we use the latest theories on how bilingual people learn to help us teach computational literacies?

Sara Vogel[a]*, Christopher Hoadley[b], Ana Rebeca Castillo[c], and Laura Ascenzi-Moreno[d]

[a]Urban Education, The Graduate Center, The City University of New York, New York City, United States of America; [b] Educational Communication and Technology, New York University, New York City, United States of America; [c]New York City Department of Education, New York City, United States of America; [d]Brooklyn College, City University of New York, Brooklyn, United States of America.

*svogel@gradcenter.cuny.edu

Languages, literacies, and literate programming: Can we use the latest theories on how bilingual people learn to help us teach computational literacies?

Background and Context: In this theory paper, we explore the concept of translanguaging from bilingual education, and its implications for teaching and learning programming and computing in especially computer science (CS) for all initiatives.

Objective: We use translanguaging to examine how programming is and isn't like using human languages. We frame CS as computational literacies. We describe a pedagogical approach for teaching computational literacies.

Method: We review theory from applied linguistics, literacy, and computational literacy. We provide a design narrative of our pedagogical approach by describing activities from bilingual middle school classrooms integrating Scratch into academic subjects.

Findings: Translanguaging pedagogy can leverage learners' (bilingual and otherwise) full linguistic repertoires as they engage with computational literacies.

Implications: Our data helps demonstrate how translanguaging can be mobilized to do CS, which has implications for increasing equitable participation in computer science.

Keywords: K-12 computer science education; bilingual education; literate programming, translanguaging, computational literacies

Introduction

As universal Computer Science education initiatives have scaled up over the last several years, some states in the United States, including Texas, Virginia, and Georgia have passed laws enabling students to fulfill world language graduation credits by taking courses in computer science. In response, a host of detractors have decried the equivalence, including the CS education advocacy organization Code.org (Hirotaka, 2014) and the American Council for the Teaching of Foreign Languages (2017). The debate centred around the question: "Is learning CS just like learning a world language?” This question comes from a place of competition for limited curricular time, but often reflects a simplistic notion of language, computer science, and particularly of programming languages. Talking about curricular priorities hides a larger question, namely, how computer science is taught and learned with and through language.

Despite some commonalities, learning a programming language isn’t the same as learning a language like Spanish, Chinese, or French. But there are aspects of language learning and in particular, learning literacies across and through language, that may provide clues to how we can build on learners’ experiences to support their participation in computational literacies. The modern understanding of literacy relies less on producing or decoding written text, and more on skills and practices that allow participation in a discourse. Discourses, as defined by researcher of sociolinguistics, James Gee, are “ways of behaving, interacting, valuing, thinking, believing, speaking and, often, reading and writing that are accepted as instantiations of particular identities (or ‘kinds of people’, see Hacking 1986, 1994) by specific groups” (2015, p. 4). Participation in discourse is an important and empowering frame for learning. Failure to have access to, full participation in, and the ability to transform discourse are mechanisms by which injustice perpetuates itself in education and beyond. As computer science emerges as a relatively novel topic in the academic core, we need to clarify first what discourses are present in CS practice and education, and secondly we need to better understand the processes of promoting full participation in those discourses. In addition, since current participation in CS is not equitable, efforts to support “CS for all” must attend to how discourse marginalizes various groups including women and girls, non-dominant racial and ethnic groups, students with disabilities, and low-income students who may use language differently from dominant groups in CS fields. For this purpose, considering how teaching and learning CS is like, or unlike, language and literacy is just as important as considering how teaching and learning CS is like, or unlike, math and science education.

In this article, we consider the relationship between computer science, language, and literacy through a wider lens. We go beyond comparing the learning of programming languages and foreign languages by drawing on concepts from sociolinguistics and bilingual education. Following other scholars in this area (diSessa, 2000; Burke, 2012; Kafai & Burke, 2014; Vee, 2017), we argue that programming and computational thinking are aspects of literacies in the broadest sense, i.e., sets of practices that allow people to think alone and together while using meaningful representations that facilitate thinking and communicating. We posit a new approach to the teaching and learning of computer science rooted in the philosophy of literate programming (Knuth, 1984) and a theoretical lens from bilingual education called *translanguaging* (García & Li Wei, 2014). Finally, we describe how we have attempted to use translanguaging pedagogical strategies as a way to teach programming with the Scratch environment in bilingual middle school contexts. These strategies leverage the full repertoire of learners' semiotic[1] and linguistic skills, thus helping students participate in computational literacies.

Programming, Language, and Literacy

To reframe the debate around whether or not learning to program is like learning a language, we draw on conceptions from recent research and scholarship in sociolinguistics. What computer scientists call "natural language" ("natural-language understanding," 2016) -- as in "natural language processing," when a computer interprets a document written in human language -- is a complex construct. Depending

[1] A term used by linguists to describe making meaning with signs, symbols, and activity.

on one’s perspective, “language” might refer to two distinct, but interrelated ideas: “a named language” (a noun) or “languaging” (a verb which stands in for meaning-making, communicating, and expressing) (Otheguy, García, Reid, 2015).

When referred to as a noun, many use the word “language” to denote a named language, like Chinese, French, or Spanish -- a rule-bound system of communication, often politically and socially defined from an external perspective, meaning, outside of the language user (for example, the *Académie Française* makes decisions about the ‘official’ French language rather than all speakers of French). Programming languages (and their kin, such as protocols or standard data formats) used in computer science (e.g., JSON, C++, and Python) have some commonalities to named languages. Learning both programming languages and named languages involves understanding semantics, syntactics, and pragmatics (Pea & Kurland, 1984). The rules and vocabulary of programming languages are defined by standards bodies or implementers of compilers and interpreters, rather than emerging bottom-up from programmers who ‘speak’ those programming languages. But, as noted by many of the detractors to policies that count CS as a foreign language, unlike named languages, programming languages are highly constrained -- not nearly as versatile as a named human language like Spanish or French (American Council for the Teaching of Foreign Languages, 2017). Programming languages are not designed to convey the breadth of meaning that human languages do, nor can they.

These arguments are perhaps enough to prove that CS should not replace foreign language requirements for graduation because they are not equivalent, but there *is* some relationship that programming languages have to language, which the second definition of “language” can help us clarify. The second definition of “language” is better expressed through the verb “languaging” (García, 2009, p. 31). This conception centers how people communicate, make meaning, and express themselves. While not everyone might be viewed by society as a speaker of a particular named language, languaging is a universal capacity of human beings. People “do language” by deploying features from their repertoires (lexical, morphological, and grammatical linguistic features), as well as a range of other signs and symbols (images, gestures, written text) in concert and in context to achieve a desired effect. In this sense ‘languaging’ is more than merely executing a communications protocol; it is, instead, participation in the evolution of meaning by tying together the wider culture, the context of interaction, and particular semiotic signals, together with a deep and ever-changing process of interpretation and meaning-making. The dynamic, user-focused quality of languaging is perhaps best realized when considering the emergent ways people communicate that don’t reflect normative language standards: in text messages, the private language of teenagers, the mix of words from different regions used by a neighborhood grocer in a multiethnic neighborhood, the assemblages of text and images in memes. If we view languaging this way, we can consider the features of programming languages, such as code and syntax, as signs and symbols which become part of a person’s repertoire, and which can be used to make meaning both with and for computers and other humans. The formal specification of how computers interpret code may be relatively rigid, but code is only one part of how computer scientists communicate with each other. Computer scientists might communicate through non-code *formal* symbolic systems like flowcharts or UML diagrams, but they are also ‘languaging’ when they use less formal means: when they write comments in their code, talk about their work while pair programming, present at conferences, debug collaboratively on message boards, or share mockups and prototypes. Languaging in computer science involves more than the mechanistic decoding or creation of written texts in a named language, and it is certainly more than typing in computer code to be executed by a machine.

Another concept from linguistics helps us make sense of the languaging that takes place in computer science, the concept of *literacies.* By literacies we mean learning how to "interpret texts of a certain type in certain ways...through having access to, and ample experience in, social settings where texts of that type are read in those ways" (Gee, 2015, p. 53). The traditional definition of literacy honed in on the decoding and producing of print text in standard named languages, which marginalized people whose literacies and language practices were deemed "non-standard." New definitions of literacy, e.g. Gee's, come out of research and scholarship in New Literacy Studies from the last several decades, and posit that reading and writing are social, political, cultural and ideological practices (Street, 1984; Street, 1993; Knobel & Lankshear, 2007). Put another way, learning how to do languaging in specific contexts and for specific purposes is what we mean by literacy. Literacy from this perspective is about meaning-making, study and exposure to what Lave and Wenger call communities of practice (1991), or participating in a community of discourse, i.e., being able to co-construct meaning with a larger community of interlocutors (Hoadley & Favaro, 2015). This definition also enables educators and researchers to attend to the ways that language and power are intertwined (Bourdieu, 1991). In 1996, the New London Group's multiliteracies framework argued that the diverse and rich multilingual practices of people in the 21st century, as well as our increasing use of digital technologies to communicate and make meaning, imply not one static, standard "Literacy," (i.e., encoding and decoding written texts in a named language like the Queen's English) but domain and context-specific "multiliteracies," echoing the term 'multimedia' (New London Group, 1996). There are literacies associated with, for instance, communities on Instagram, (eg: food instagrammers, teenage selfie instagrammers, etc.) and different, evolving expectations, norms, and etiquette around how to combine written words, photos, emojis, geolocation, hashtags, and particular registers in those communities. In addition to those physical representations, status in the community, power, and culture all play a role in how people interpret and make-meaning through Instagram. Academic disciplines also have literacies: the ways in which scientists communicate with each other (both in journal articles and in other venues), the ways high school debate clubs communicate both during and outside of debates, and the ways in which mathematical concepts are discussed and understood in math classes.

Thus, when we think of programming 'literacy,' we need to think beyond formalistic reading and writing of computer code, just as linguists consider the literacy tied to languaging as being more than the mechanistic decoding or creation of written texts in a named language. Returning to our prior point about computer scientists, we can therefore say 'languaging' with programming languages requires multiple, intertwining literacies depending on the purpose and context for applying programming languages, computing concepts and practices. Those literacies might include those associated with 'reading," interpreting, and critically examining the computational artifacts we come across in daily life, echoing educationist Paulo Freire's conception that reading the word is reading the world (Freire, 1985). There are also computing literacies associated with different domains and communities, for instance, analysis and visualization of scientific data sets in different fields, programming digital stories or artwork for self-expression, and building tools and applications that have social justice aims. Just as there are as many Instagram literacies as there are subcommunities on the platform, there are as many computational literacies as there are communities that use computational artifacts as part of languaging. Those literacies may incorporate use of ‘named’ programming languages (e.g., people who ‘speak’ Python, Java, R, etc.) but also include knowledge of particular norms (e.g., how to participate in a particular open source project, or form of agile software development, etc.). Kafai and Burke argue for the social nature of computing literacy: “we need to move beyond seeing programming as an individualistic act and begin to understand it as a communal practice that reflects how students today can participate in their communities” (2014, p. 128).

A Literate Programming Philosophy for Participating in Computational Literacies

Given this broader notion of literacies, it is easy to see how computer code can be seen as one of the ways people communicate and make meaning together. We can move beyond narrowly looking at coding proficiency as just learning to produce and consume (syntactically correct) programs. Conceptualizing programming in that way would be analogous to treating reading and writing as memorizing grammar rules and dictionary definitions, disconnected from being able to read any useful literature or write anything as part of participating in a discourse community. Instead, we see different kinds of communities in which computer code is taken up as part of a larger, authentic meaning-making activity, for instance in the Maker movement (Vossoughi, Hooper, and Escudé, 2016), in various culturally relevant computing projects (e.g. Scott, Clark, & White, 2013), and even among professional computing communities. There are thousands of threads on professional sites like Stack Overflow (http://www.stackoverflow.com/) that help demonstrate how people communicate not only about, but with and through, computer code. Usually this communication is about concepts relevant in computer science, but not always.

So then what is computational literacy? Historically, ‘computer literacy’ or ‘computing literacy’ has implied having some basic skills in using computers, or being able to use common terminology about computers (such as ‘hard drive’, ‘RAM’, etc.). This notion of being able to talk *about* a domain and how it relates to social issues, in contrast to having deep knowledge in the domain, is similar to the way, for instance, *scientific literacy* has been used to refer both to knowledge about science that aids in everyday application of scientific ideas, in contrast to the way *science literacy* has been used to refer to more focused disciplinary knowledge that might prepare someone to be a scientist (Roberts & Bybee, 2014). Certainly, as computers have gone from rare objects to commonplace, our society has developed a literacy about computers, in that we all now are more and more comfortable with terminology like “log on”, “download”, “bandwidth”, etc. But these are not typically the terms or concepts associated with programming or computer science. We can ask if, just as there is a distinction between scientific and science literacy, there should be a distinction made between computational literacy as a general computing literacy, versus computational literacy as computer science literacy that is more focused on disciplinary practices found among professional computer scientists. Hiding among these distinctions is the question of whether how professionals use code should define expert disciplinary practice; just as amateur and citizen science can help redefine what we mean by doing science (Irwin, 1995), we can and should question whether expert computer science literacy need be solely defined by what professionals do.

If we take the notion of literacy in the New London sense seriously, we have to treat computational literacy as multiple literacies, and as acts of participation in a community in which computational representations are one of the ways in which people make meaning. Participating in computational literacies in this sense includes creating, reading, modifying, and, yes, executing code and computing as part of a community of discourse. Put another way, when looking for computational literacies, we should be asking the question “What conversation is this code a part of?”

This stance towards code as a representation for participating in a literacy is not novel. Even before the comments by authors like Kafai and Burke, noted computer scientist Donald Knuth made impassioned pleas for what he called ‘literate programming’ (1984). Knuth advocated that computer scientists should be able to access and become familiar with great works of coding, just as writers might become familiar with great works of literature. His stance can be boiled down to a belief that code is for people, and not just computers, to read. This idea that code is for sharing has been greatly facilitated in professional programming circles by the emergence of tools for distributed software engineering (e.g., version control systems), and open source code projects (including the legal, cultural, and technological infrastructure they rely on, from copyleft licenses to github).

Seeing coding as participating in the literacies of a community has implications for how people learn to code. Some researchers frame the object of learning in computing education as a form of ‘computational thinking’ (Wing, 2006; Grover & Pea, 2013), which tends to emphasize the ways that computer scientists decompose and solve problems using algorithms and computation (if not actual computers). Learning these skills and practices is an important part of being a computer scientist. However, it neglects other important aspects of how computer scientists work; just as learning grammar and reading is insufficient to do creative writing, learning the ins and outs of the particular syntax of various programming languages is insufficient to be a computer scientist. Even learning practices like abstraction, debugging, or applying commonly used algorithms does not make someone into a proficient computer scientist. Rather, we find the notion of literacy to be more resonant with the ideas of Papert and others of computation as an expressive and powerful mind tool. Framing what students learn as more like our modern notion of literacy--the idea of "engaging in ever more meaningful conversations in a community or discipline where computing practices and concepts are employed to accomplish authentic work and communication" (Vogel, 2020) allows us to consider not only the real work computing knowledge does in the lives of these learners, but also to consider the social context of that work and the infrastructures of communication that underpin it. As Annette Vee argues, "Literacy is a theoretically rich way to understand the relationship between communication and technology, in part because those who study literacy have long grappled with what it means for humans to work with socially situated, technological systems of signs." (Vee, 2017, p.4). Or, as Kafai and Burke state, "Computational thinking and participation do not need to make every child a programmer. But by giving children the capacity to make and share, they give children the opportunity to understand the digital public. This is why programming is not only the new literacy of the millennium but may be the defining literacy of future generations" (Kafai & Burke, 2014, p. 122, cf. "critical" and "situated" framings of computational thinking vs. cognitive in Kafai, Proctor, & Lui 2019).

The theory of communities of practice suggests that learning can take place when people have a way to do legitimate peripheral participation (Lave & Wenger, 1996); and programming environments can be designed in such a way to support learners from participating in literacies of code without necessarily fully mastering the syntax of a programming language or developing a full conceptual model of the computer. The Scratch project at MIT (following on Seymour Papert's legacy of Logo, Papert 1980) designed their programming language to leverage aspects of meaning-making that don't rely on text-based code, for instance emphasizing graphical user experiences or embodied robotic actions, and providing spaces to document code projects as metadata (e.g. project titles and free-form textboxes labelled “Instructions” and “Notes and Credits” to describe the project outside of the code itself). The Scratch environment began as a downloadable programming environment, but has evolved into an entirely cloud-based programming environment in which any program can be shared, all shared programs are accessible to the world, distributing a program always implies being able to see the code that makes the program run (with no ability to distribute an ‘executable’ without the ‘source code’) and all shared programs can be remixed. In this way, Scratch attempts to encourage its users to make meaning with the Scratch code within larger conversations.

A Translanguaging Pedagogy for Computational Literacies

As described in the last section, we view participating in computational literacies as having conversations about, with, and through code, in different discourse communities and contexts. Learning computer science through this frame demands a pedagogy that centres human meaning-making, communication, and expression (Vogel et al., 2019). We draw on a pedagogy rooted in a theory of language and literacy learning and teaching with bilingual people: *translanguaging*.

What is translanguaging?

Translanguaging pedagogy is rooted first in the theory of translanguaging, a concept from applied linguistics and bilingual education (García & Li Wei, 2014). Walk by any playground where multilingual students are at recess, and the dynamism of their language practices becomes apparent. Students use words from English, their home languages, and slang, sometimes all in the same utterance. They sing, gesture to each other, cry out, and fall over laughing as they play face-to-face game, and scroll through their phones to make memes of text, video, images and emojis. While many schools institute policies that regulate students' language -- requiring they use English only during some parts of the day, for instance -- like those students on the playground, people's language practices cannot be so easily categorized into named language buckets. Translanguaging refers to the fluid practices of people as they use and leverage the full span of their linguistic, semiotic, and social resources to make meaning (García & Li Wei, 2014; García & Kleyn, 2016). The translanguaging of bilingual and multilingual people is especially marked in society, but all people exhibit it. These dynamic languaging practices defy categorization into traditional language categories like "English," "Spanish," or "French," as actors use all of their communicative resources fluidly and flexibly (Otheguy, García & Reid, 2015).

Translanguaging as an analytical lens has transformed how language learning is viewed. Traditional cognitive models of bilingualism posed that language learning was a linear additive process involving the acquisition of formal grammatical rules (Lambert, 1974). As globalization and mass migration have made obvious the superdiverse linguistic environments in which speakers operate (Arnaut, Blommaert, Rampton, and Spotti, 2015; Blommaert, 2010; Jørgensen, 2008, Vogel & García, 2017), sociolinguistic studies have started to rethink that premise to account for the non-linear, dynamic ways that bilinguals actually use and develop language (Faltis, 2014; García, 2009; García and Li Wei, 2014). People do not learn languages as bounded wholes, but rather incorporate the "language features" (particular lexicons and syntaxes) they need to communicate depending on context and their own purposes.

Translanguaging as applied to education has shifted how language learners are positioned: as active and agentive in the creation of language. Traditionally, education research and practice on especially emergent bilingual students (students who may use a language at home and are learning a new one at school, see García & Kleifgen, 2018) has been guided by deficit-based theories (Valencia, 2010) which characterize their languaging as fractured, deviant, or deficient, especially when students' identities as bilinguals intersect with non-dominant racial, ethnic, and class identities (García, 2009; Flores & Rosa, 2015). Translanguaging theory grew out of a critique of those deficit-based approaches. It emphasizes what students have and can do, rather than what they lack, or what schools perceive to be the object of their learning (Standard English, for instance). It highlights the diverse and dynamic repertoires students draw upon as emergent bilinguals or multilinguals (García & Kleifgen, 2018). Many associate translanguaging with what bilinguals do when they use words in a more familiar language as they grasp for words or phrases in another one, but in actuality, the trans- prefix denotes that multilingual people's language practices "go beyond" use of state-endorsed named languages (García & Li Wei, 2014, p. 42; Li Wei, 2011; Vogel & García, 2017). Such a view of bilingualism challenges previous models of bi- and multilingualism, encouraging researchers and practitioners of applied language fields to value those individuals and peoples whose language practices have been traditionally minoritized and labelled as being "non-standard." Translanguaging is a theory crafted in solidarity with minoritized language users (García, Flores & Woodley, 2012). The concept of translanguaging is useful in understanding how even monolingual people have dynamic language repertoires that go beyond the official named language they speak. There has been much research examining multi/bilingual students' translanguaging in various school subjects, and evidence shows that for emergent bilinguals, a translanguaging pedagogical approach can support STEM learning and participation in STEM discourse (Mazak & Donoso, 2015; Poza, 2018; Rawal et al., 2019; Suárez, 2017).

Until recently, scholars have emphasized the *linguistic* aspects of translanguaging. There is now interest in extending the boundaries of translanguaging to fully encompass how emergent bilinguals call upon a "complex network of multiple ***semiotic*** signs" (García and Li Wei, 2014, p. 25), and indeed, how they "orchestrate their diverse and multiple meaning- and sense-making resources in their everyday social life" (Li Wei, 2018, p. 27). This shift in translanguaging as an analytical lens recognizes that people draw on resources "that they embody (e.g., their gestures, their posture), as well as those outside of themselves which through use become part of their bodily memory (e.g., computer technology)" (García & Kleifgen, 2018, p. 93). Extending translanguaging to encompass interaction with both social and digital tools (Blackledge & Creese, 2017; Kusters, Spotti, Swanwick & Tapio, 2017; Vogel, Ascenzi-Moreno & García, 2018) may explain how learners draw upon all their resources in CS activities.

What is Translanguaging Pedagogy?

The application of a translanguaging framework can have real implications for valuing the language practices of people who have been marginalized in CS fields. Translanguaging theory has been developed into a pedagogy, which, when applied to learning environments is a radical departure from how emergent bilinguals and their resources are viewed and built upon in most school contexts. Rather than envisioning student learning occurring solely through the lens of a socially constructed language (such as English), teachers who employ translanguaging pedagogy in their classrooms are aware that students bring a variety of language and social practices to school that must be tapped in order to properly build on their capacities. For example, teachers can employ translanguaging practices to assist students to understand vocabulary, develop and strengthen concepts across content areas, gain metalinguistic awareness and connect with others (García & Leiva, 2014; García & Li Wei, 2014; Garrity, Aquino-Sterling, &

Day, 2015). Teachers can enact translanguaging pedagogies through intentionally building in opportunities for students to call upon all their language and social resources within lessons and across longer units of study (Celic & Seltzer, 2012; García, Ibarra Johnson & Seltzer, 2017). Vogel & García (2017), outline the core components of teachers' translanguaging pedagogy as identified in García, Ibarra Johnson & Seltzer (2017):

- **Stance**: A belief that students' diverse linguistic practices are valuable resources to be built upon and leveraged in their education.
- **Design**: A strategic plan that integrates students' in-school and out-of-school/community language practices. The design of instructional units, lesson plans, and assessment are informed and driven by students' bilingual practices and ways of knowing, and also ensure that students have enough exposure to, and practice with, the language required for different academic tasks.
- **Shifts**: Moment-by-moment changes to an instructional plan based on student feedback.

Students' translanguaging practices, whether officially encouraged in school or not, have been demonstrated to facilitate engagement, basic and complex comprehension, critical thinking, metalinguistic awareness, and language production (García & Kleyn, 2016; Palmer, Martínez, Mateus and Henderson, 2014; Pontier and Gort, 2016). There is an emerging body of research which documents how translanguaging supports and enhances student meaning making. Daniel & Pacheco (2016) trace how multilingual adolescents craft understanding by actively using their home language regardless if it is a school-sponsored process. Kibler (2010) remarks that students' bilingual language proficiencies impact the writing process, documenting how bilingual students, regardless of their level of English language proficiency, use their home language to engage in a variety of meaning making practices such as understanding content, interacting with peers, composing, and learning words from a new language. She advocates that teachers recognize these processes and harness them to improve emergent bilingual students' writing. Hopewell’s work in reading (2010) attests to emergent bilingual students’ use of home language in order to negotiate reading comprehension to understand texts. She concludes to truly serve emergent bilingual students, teachers will need to “recognize and embrace students’ full learning potential” through the integration of home language practices into reading instruction (Hopewell, 2010, p. 617). Alvarez (2014) documents how emergent bilinguals also use what he calls “hybrid sense-making practices” (p. 327) to forge strong relationships across school and home contexts. Seltzer (2019) highlights transgressive aspects of translanguaging pedagogy. In her case study of English teacher, she describes how a teacher developed opportunities for students to role play their experiences and understanding of language. In doing this, students were able to critically analyze how their racial and linguistic identities are positioned within society through language. This study points to the potential contribution of translanguaging pedagogy to create new critical practices in classrooms. Taken together, these studies document how translanguaging is crucial in facilitating students’ academic engagement across content areas and in school in general.

Translanguaging pedagogy has the potential to help solve challenges in CS education related to broadening participation. Translanguaging pedagogy is aligned with movements in education research such as culturally sustaining pedagogy, which values “linguistic, literate, and cultural pluralism” as desired outcomes for education (Paris & Alim, 2014; García & Li Wei, 2014). Applying translanguaging pedagogy to CS education builds on previously developed culturally relevant and responsive computing approaches (Eglash, Gilbert, & Foster, 2013; Scott, Sheridan, & Clark, 2015; Goode, Chapman, & Margolis, 2012) by recognizing the critical role that language as sociocultural practice plays in CS learning for emergent bilingual populations and perhaps others. Given that translanguaging pedagogy requires educators to attune themselves to the dynamic profiles of emergent bilingual learners, it opens up space for them and their students to negotiate what “counts” as valid practices and concepts in the discipline of CS itself, and to use all of their language resources in pursuit of learning computational literacies.

Translanguaging pedagogy for computational literacies

In this section, we describe our approach to using literate programming and translanguaging pedagogy to foster computational literacies through a design narrative (Hoadley, 2004). We begin with an overview of our context, project, and methods, and then discuss our aims and overall pedagogical strategies. We then describe an example of a classroom activity that illustrates the approach, and finally we discuss some implications of the approach.

Our project, Participating in Literacies and Computer Science (PiLaCS), is a research-practice partnership between university researchers, the New York City Department of Education, and teachers and administrators in three middle schools with bilingual programs in the Washington Heights neighbourhood of New York City, a historically Latinx (Dominican) area, which has experienced gentrification and its displacement of some Latinx families. The neighbourhood’s school district serves 20,651 students from kindergarten through high school, and 85.4% of students are Latinx (New York State Education Department, 2019), many of them first or second generation immigrants. Researchers in the partnership include experts in bilingual education, applied linguistics, computer science and technology education, and urban education. The practitioners include teachers in Language Arts, Social Studies, Science and Math who have bilingual or Teaching of English to Speakers of Other Languages (TESOL) certification, as well as their principals and district-wide administrators engaged in New York's Computer Science for All initiative. New York City has committed to exposing all students to one meaningful computer science learning experience per grade band by 2025, and is actively experimenting both with CS curricula per se (i.e., computer science classes or curricular units designed to teach only CS) and with infusing computer science into other subjects. Our project takes the latter approach, attempting to infuse computer science curricula into other subjects at the middle school level.

Of particular interest is ensuring that computer science is accessible to the 40% of New York City students who speak languages other than English at home. Emergent bi/multilingual people new to English all over the world face additional challenges in learning to program (Guo, 2018; Reestman & Dorn, 2019), and some scholarly research has investigated how technological design choices (e.g. localization, [Dasgupta & Hill, 2017]) and pedagogical choices related to language of instruction and materials (Pal & Iyer, 2015; Soosai Raj, Ketsuriyonk, Patel, & Halverson, 2018) can support learners. Little research has been done to document the language practices of younger, K-12 emergent bi/multilinguals in the context of computing education.

While our project focuses on middle schools with bilingual and English as a New Language programs, we are not solely working with students who are learning English (whom the system designates "Multilingual Language Learners / English Language Learners"). When we meet them, students are dynamic bilinguals (García, 2009) at different points along bilingual / multilingual continua (Hornberger, 2003) in terms of their receptive and productive abilities in oral and written forms of different languages. We view all of the students in our partner classrooms through translanguaging's asset-based lens that recognizes and attempts to leverage the variety of ways they use language at home, in school, orally, in writing, and with and through technology, gesture, drawing, and other modalities. Even for a monolingual student who only speaks English, we still believe our approach of leveraging their linguistic practices (i.e., facility with varieties of English, colloquialisms, drawing, emoji, etc) can help them learn new linguistic and semiotic representations, including the formal representations of computer code.

So far, our project has used the block-based Scratch programming environment (Resnick et al., 2009), due to ease of use and approachability for novice programmers, its built-in collaborative and sharing tools (all Scratch programs can be shared with a click to the entire worldwide Scratch community, and all shared programs can be 'remixed' or copied and edited by anyone), and the fact that its interface seamlessly supports switching the interface between a number of global languages. That is, by clicking on one menu, a student can switch the interface of Scratch from English to one of fifty languages and back again, and all programming keywords used to label the programming blocks will switch from one language to another. For example, a "repeat until" block becomes a "repetir hasta que" block, and a menu item like "Save as a copy" becomes "Guardar una copia" when one toggles the interface to Spanish. Scratch supports approximately 75 languages. Our project uses collaborative teams of researchers and educators to invent ways to incorporate translanguaging pedagogy and a literate programming approach (with the comments and other non-code literate elements to programming in multiple named languages). Teachers involved in the project design and test activities that bring computer science concepts together with disciplinary content they would already be teaching, and they attempt to elicit and leverage the full language repertoires of their particular students to support their learning in the process.

In addition to designing, implementing and iterating on our pedagogical approach, our RPP also engaged in qualitative research in partner classrooms. We sought to determine how, when, and why students translanguaged (draw on a range of linguistic, social, and semiotic resources) as they engaged in computational literacies, and how could teachers, through curricular design and adaptation, support and leverage students' translanguaging while teaching computational literacies? The example which follows zooms in on just a few moments out of a larger corpus of data that was collected over two school years. This corpus included field notes, audio recordings, transcripts, photographs, digital and analog student work samples from hundreds of hours of observation spread across 6 teacher's classrooms, and transcripts and artifacts from 9 one-on-one interviews and 7 focus groups with students (loosely based on Brennan & Resnick's [2012] artifact-based interviews). The corpus also includes field notes, recordings and artefacts from co-design and reflection meetings involving researchers and teachers. To analyze student data, we drew from a methodology in applied linguistics called "moment analysis" (Li Wei, 2011), which focuses attention on the creativity and criticality of multilinguals' language-in-use at important "translanguaging moments" as well as research participants' sense-making and metacommentary about those practices. We used a triangulation method of collaborative descriptive inquiry (Himley & Carini, 2000) to explore those moments with researchers and teachers. Our characterization of the adults and their design process relies on descriptive qualitative analysis and personal reflections from the co-designers.

The examples we present to illustrate our pedagogical approach come from implementations of a curricular design our team has called the “Telenovela Unit” in bilingual Language Arts sixth and seventh grade bilingual classrooms at the school where co-author Ana Castillo and her colleague, Ashley Guílamo taught. To comply with New York State and US federal laws regarding the education of students whose level of English proficiency made them eligible for language learning services (students categorized as “Multilingual Learners / English Language Learners” [MLL/ELL] in New York), the school offered parents of the 51% of MLL/ELL students a bilingual program option.[2] There is great variation in the bilingual programs offered in NYCDOE, with some aiming for students to “transition” between a home language and English by using less of a home language in instruction over time, while others expect students to maintain and develop a home language. The program at this particular school used a “dual language” bilingual education model where the language of instruction alternated daily to support students’ bilingualism and biliteracy. The language-of-the-day policy was implemented flexibly by Castillo and Guílamo, who encouraged students to translanguage to support their learning of language and content. Bilingual students in the program had a mix of academic experiences. Some of them had experienced interrupted or inconsistent formal education, and might not have learned to read before arriving in middle school in the US, while others might have had consistent and rigorous schooling in the US or in another country. A significant number of the students were newcomers, meaning they had arrived in the US that school year, or in the previous three years. The conditions in the countries where students immigrated from might have been comparable to their lives in the States, some might

[2] Schools in New York may also serve students in “English as a New Language” classes, as long as there are bilingual options available to families nearby.

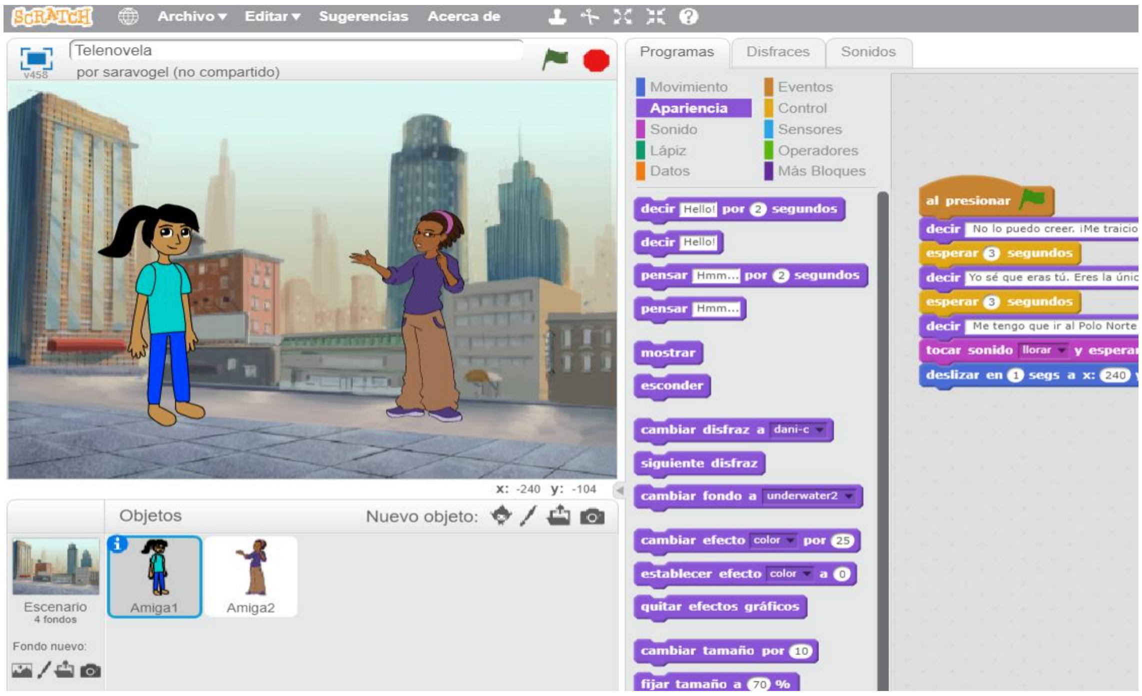
have had to cross the border with their families or had to escape dangerous situations. At the start of the unit, the students' familiarity with technology was also wide ranging – some students were experienced with and knowledgeable about computers, tablets, and mobile devices, and some never having used a laptop computer before. As Castillo, one of the classroom teachers who implemented the units, put it "The reality of these students and the inequalities in educational systems and their identities as children of color and/or immigrants, limited their access to technology and the opportunity to learn about coding and the CS world. It wouldn't be unfair to state that without a program like the one described in this paper, they wouldn't have been able to access this technology, experts in the field, and to the curriculum created by their teachers."

Below, we describe a curricular example from bilingual classrooms, excerpting data from the larger corpus to illustrate how translanguaging pedagogy can support students' participation in computational literacies.

One Translanguaging Design: Comparing Scratch and a Telenovela

In the "Telenovela" unit co-designed by PiLaCS teachers and researchers, students compared the scripts of a live action version and a Scratch animated version of a dialogue between two people. Both dialogues depicted a telenovela or Spanish Soap Opera. Students used this comparison to prepare them to eventually "remix" a Scratch project to produce their own digital telenovela dialogue. The lead designers of this unit were co-authors Castillo and Vogel, although there were many contributions from another teacher in Guílamo, and from co-authors Hoadley and Ascenzi-Moreno.The telenovela unit evidences many of the principles of translanguaging pedagogy. As noted above, the first principle of translanguaging pedagogy according to García, Ibarra Johnson and Seltzer (2017) is that teachers practice a **stance** that centres students dynamic translanguaging and ways of knowing. Castillo exhibited this stance when the design team first came together with fellow teacher Guílamo and project researchers to plan activities that would introduce students to the Scratch programming environment and interface. During a design team meeting, researchers shared with teachers that a common way to introduce Scratch is to use an analogy to a play: both employ actors (sprites), a stage, costumes, sounds, and scripts. With her Spanish-English bilingual language arts students in mind, Castillo remarked, "let's make it a telenovela instead—that's what our students watch with their families at home every night." Even those students who rolled their eyes at this genre would be familiar with its conventions.

In translanguaging pedagogy, educators strategically design instructional units, lesson plans, and assessment informed and driven by students' bilingual practices and ways of knowing, while also ensuring that students have enough exposure to, and practice with, the language required for different academic tasks García, Ibarra Johnson and Seltzer (2017). Taking Castillo's suggestion, the team designed an activity that would prompt students to compare and contrast two forms of expression: Scratch projects and telenovelas (Figure 1).



```
al presionar
decir No lo puedo creer. ¡Me traicio
esperar 3 segundos
decir Yo sé que eras tú. Eres la únic
esperar 3 segundos
decir Me tengo que ir al Polo Norte
tocar sonido llorar y esperar
deslizar en 1 segs a x: 240
```

Figure 1: Telenovela unit example Scratch project provided by the teacher

Not only would this activity support students in learning the parts of the Scratch interface and the function of programs within Scratch, it would also support language arts objectives around comparing and contrasting, storytelling, dialogue, and new computing-related vocabulary words in English and Spanish. Our design team carefully planned opportunities for students to individually and collectively engage their full translanguaging repertoires and literacies in activities. In the first lesson in teachers' sequence, designers aimed to engage students' embodied repertoires (Hua, Wei, & Jankowicz-Pytel, 2019) by asking them to act out a dramatic telenovela scene before comparing and contrasting the scene to an animated story coded in the Scratch environment. Those students selected to be "actors" did more than just read the Spanish dialogue written on their scripts -- they demonstrated their embodiment of telenovelas as a genre, mobilizing acting conventions including intonations and gesture.

Next, teachers showed students a picture of a telenovela set, and elicited the names of its different components (actors, backdrops, costumes, scripts) in both English and Spanish, aiming to expand their repertoires in both languages. Students noted down the names of the different components on the handout below (Figure 2). Teachers then showed students a Scratch version of the telenovela they had just performed, clicking the “see inside” button in Scratch to see the back-end editor that the project’s creator gets access to. Afterwards, teachers labeled the different parts of the Scratch interface on the Smartboard, as students followed along on their own handouts. Teachers asked students about the similarities and differences they saw so far between Scratch and telenovelas, noting them down on a chart or whiteboard as students dictated.

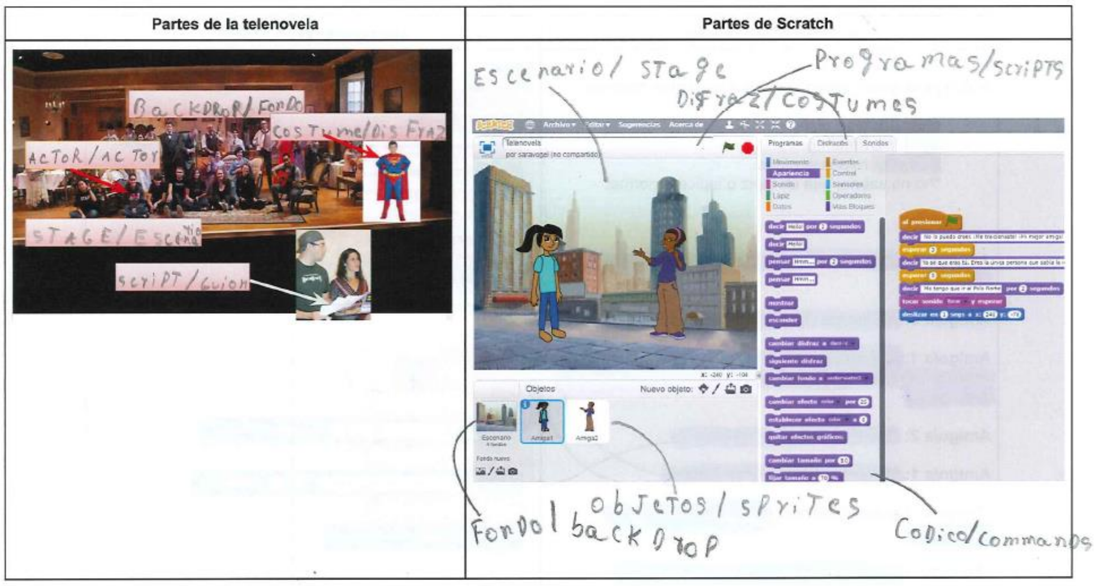
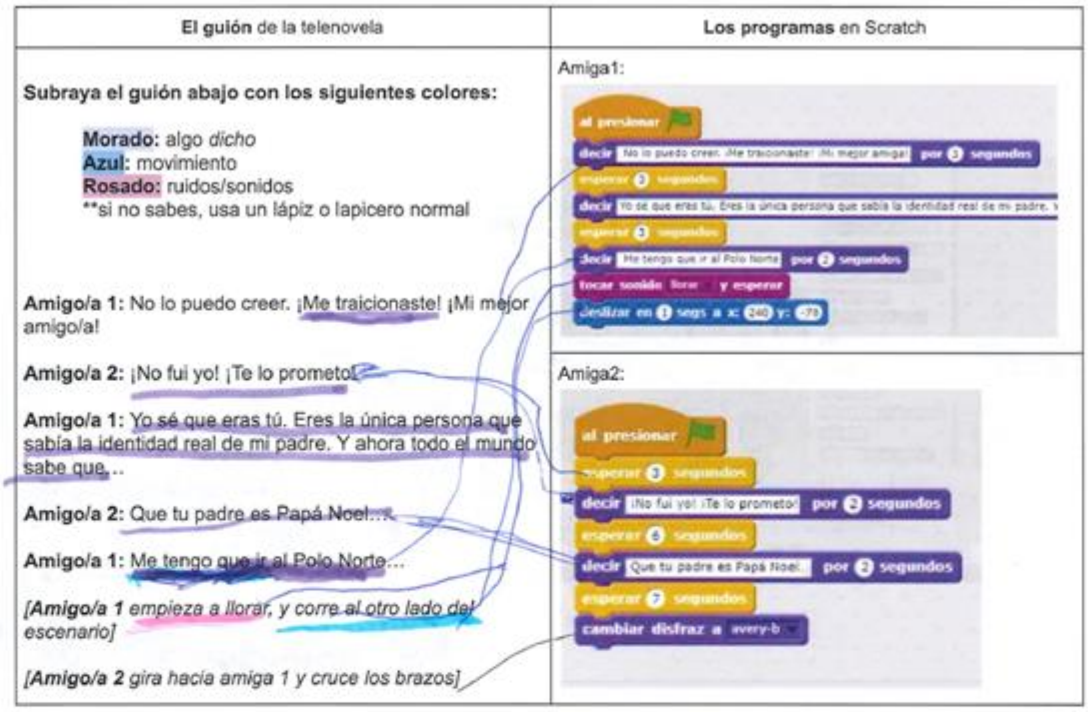


Figure 2: 7th grader Julio’s[3] worksheet.

Next, teachers asked students to turn over the handouts from the previous activity, and to fold them in half so they would only be able to see the left side -- the telenovela script (Figure 3). Teachers gave students a few minutes to work independently or in pairs to colour code the different components of the telenovela script (dialogue in purple, movement in blue, sounds or noises in pink), attending to

[3] Student names are self-selected pseudonyms.

whether elements of the script were dialogue, movement, or sounds and noises. This aspect of the activity helped students begin to ascribe meaning to the colours of the Scratch blocks, leveraging colour as a resource in students' translanguaging repertoires. Students were then asked to unfold their sheets and to draw lines between elements that seemed similar in the script and the Scratch codes used to control the two sprites in the dialogue, as a way to start making connections between the two genres. This literate programming (Knuth, 1984) activity encouraged students to use human language representations and literacies (Spanish dialogue, colour, lines, telenovelas) to help them make meaning of new semiotic resources and representations (Scratch code blocks and their sequencing).



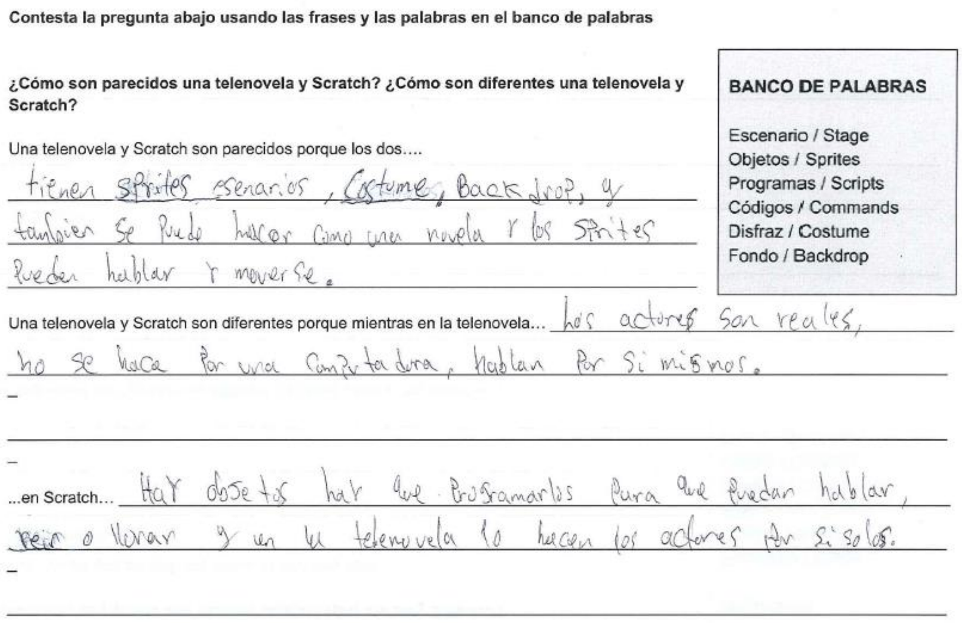
El guión de la telenovela	**Los programas** en Scratch
Subraya el guión abajo con los siguientes colores: **Morado:** algo *dicho* **Azul:** movimiento **Rosado:** ruidos/sonidos **si no sabes, usa un lápiz o lapicero normal **Amigo/a 1:** No lo puedo creer. ¡Me traicionaste! ¡Mi mejor amigo/a! **Amigo/a 2:** ¡No fui yo! ¡Te lo prometo! **Amigo/a 1:** Yo sé que eras tú. Eres la única persona que sabía la identidad real de mi padre. Y ahora todo el mundo sabe que... **Amigo/a 2:** Que tu padre es Papá Noel... **Amigo/a 1:** Me tengo que ir al Polo Norte... *[**Amigo/a 1** empieza a llorar, y corre al otro lado del escenario]* *[**Amigo/a 2** gira hacia amiga 1 y cruce los brazos]*	Amiga1: al presionar decir No lo puedo creer. ¡Me traicionaste! ¡Mi mejor amiga! por 3 segundos esperar 3 segundos decir Yo sé que eras tú. Eres la única persona que sabía la identidad real de mi padre. Y esperar 3 segundos decir Me tengo que ir al Polo Norte por 2 segundos tocar sonido llorar y esperar deslizar en 1 segs a x: 240 y: -79 Amiga2: al presionar esperar 3 segundos decir ¡No fui yo! ¡Te lo prometo! por 2 segundos esperar 6 segundos decir Que tu padre es Papá Noel... por 2 segundos esperar 7 segundos cambiar disfraz a avery-b

Figure 3. 6th grader Álvaro highlighted various parts of the telenovela dialogue in different colours and then made connections to the Scratch program code.

During a class discussion that followed, students justified their colour-coding choices and the connections they identified between the telenovela script and the code, translanguaging a great deal in the process. Sixth grader Álvaro read some of the telenovela script aloud to his partners to help him determine whether it had the expressive quality of dialogue. When puzzling over the meaning of the only motion code block in the dialogue, “deslizar en 1 segs a x: 240 y: -79” (“glide for 1 second to x: 240 y:-79”), Álvaro got up from his seat and slid across the floor, as if he were a baseball player sliding towards home plate, demonstrating his understanding of the concept “deslizar” with his whole body and making a connection to a sport he enjoyed. When Guílamo asked what the X and Y represented in that code block, sixth grader Mariposa used Spanish, English and gesture, and her knowledge of math, saying “eso es como un de estos” (this is like one of these) and crossing her arms to denote a coordinate plane, saying “they use that in math.” Guílamo stepped in to build on Luisa’s gesture with the math vocabulary, “ordered pair.”

Guílamo and Castillo also called students’ attention to a number of salient features, such as the elements of the Scratch code that do not seem to have an analogous component in the telenovela dialogue. During the whole class discussion, a few students noticed the “esperar 2 segundos” blocks (“wait 2 seconds”). Guílamo asked why such blocks might exist in the Scratch code, but not in the telenovela script. Sixth grader Nikki remarked that telenovela actors know to wait between lines, but if the wait blocks aren’t present in the code, characters will not have enough time to say their parts before the next one speaks. Guílamo intentionally and playfully interrupted the end of Nikki’s comments, incorporating roleplay into her translanguaging to demonstrate how “nada se va a entender” (“nothing will be understood”) unless the programmer places these wait blocks into the code.

Students then synthesized the ideas from the previous activities in a writing exercise which asked them to compare telenovelas and Scratch project (Figure 4, translation in Figure 5). Castillo encouraged students to begin their writing project with a turn and talk activity so students could connect their ideas in oral form to their ideas in writing. She also modelled compare-contrast discourse and sentence structure, as well as coherence and cohesion in the process, providing students with sentence-starters in English and Spanish and a bilingual word bank with the relevant Scratch interface vocabulary to encourage students to use new words and language structures. In this activity, students used the new words introduced to them during this lesson, but also came up with idiosyncratic terms to refer to components of the interface, such as “muñequito” (little doll) for the sprites and “vanderita” for the little green flag. Teachers welcomed such opportunities for students to use their full translanguaging repertoires to make sense of code and the interface in these ways, exhibiting the third principle of translanguaging pedagogy, **shifts**, or the moment-by-moment changes educators make to an instructional plan based on student feedback (García, Ibarra Johnson and Seltzer, 2017).



Contesta la pregunta abajo usando las frases y las palabras en el banco de palabras

¿Cómo son parecidos una telenovela y Scratch? ¿Cómo son diferentes una telenovela y Scratch?

Una telenovela y Scratch son parecidos porque los dos.... tienen sprites esenarios, Costume, Back drop, y tanbien se Puede hacer como una novela y los Sprites Pueden hablar y moverse.

Una telenovela y Scratch son diferentes porque mientras en la telenovela... los actores son reales, no se haca Por una Computadora, hablan Por si mismos.

...en Scratch... Hay objetos hay que Programarlos Para que Puedan hablar, reir o llorar y en la telenovela lo hacen los actores Por si solos.

BANCO DE PALABRAS

Escenario	Stage
Objetos	Sprites
Programas	Scripts
Códigos	Commands
Disfraz	Costume
Fondo	Backdrop

Figure 4. Seventh grader Diani’s writing activity.

Figure 5. Translation of above

Answer the question below using the phrases and words in the word bank.

How are a telenovela and Scratch similar? How are a telenovela and Scratch different?

A telenovela and Scratch are similar because they both… [start student response]

> *Have Sprites, stages, costumes, backdrops and also you can make a novela and the sprites can talk and move.*

A telenovela and Scratch are different because while in a telenovela [start student's response]

> *the actors are real, you don't make it with a computer, they talk for themselves.*

In Scratch…[start student's response]

> *There are objects, you have to program them to talk, laugh or cry, and in the telenovela the actors do that for themselves.*

Finally, students were invited to remix the telenovela project in Scratch to depict their own dramatic scenes, and to present these to the class. In merging their telenovela and language arts literacies with the Scratch interface and code through translanguaging and literate programming activities, designers of this activity aimed for students to experiment with and learn the names and functions of the different parts of the interface, and to recognize the potential of programming as a tool for bilingual self-expression -- one of the key goals of our project's integration into a bilingual language arts class.

Summary and conclusions

Activities like Castillo and Guílamo's telenovela unit help us demonstrate several of the ideas we have discussed about the relationship between computer science, language, and literacies. In this unit, code is not something to be learned in a vacuum—and it certainly does not replace traditional language learning. This activity frames code as one more meaning-making resource students can use to participate in broader computational literacies—to, in this case, have conversations about bilingual self-expression in a language arts context. The design of the activity encourages students to orchestrate resources from oral and written Spanish and English, gesture, movement, drawing, roleplay, and the conventions of telenovelas, along with code, to better understand basic ideas about Scratch programming. Translanguaging theory predicts that learners leverage existing linguistic and semiotic repertoires as they expand to adopt new ones, and in our case this is what we see, from the use of knowledge about telenovelas to better understand the relationship between sprites, scripts, etc., to the leveraging of knowledge of baseball terms in Spanish to learn the glide command in Scratch. Translanguaging theory also predicts that giving students encouragement to use their full repertoire will aid them as they attempt to participate in new literacies. By framing the activity as a language arts activity and one in which sharing and explanation are central (as opposed to solely creating and running programs), we invoke the ideas of literate programming and thereby leave the door open for students to bring their full linguistic repertoire to bear. Literate programming is not just a technique that supports experts moving from pseudocode to working software, but a philosophy that helps us recenter the languaging aspects of computational literacy, and this has powerful implications for how we teach CS at all levels, including for novices. Languaging and translanguaging are important ways to think about acquiring computational literacies and especially fluency in programming languages.

Although translanguaging pedagogy is proposed as a way to support multilingual learners, it may equally prove useful as learners engage in multiliteracies of programming and computer science concepts. Our project was originally conceived as a way to ensure full participation of emergent bilingual learners in school-based computer science instruction in the United States, but seeing coding as part of a conversation should be applicable to all learners. For monolingual English speakers, it can be a powerful lens for seeing how students coordinate among multiple semiotic resources, from code to pseudocode to gesture, images, or symbols. For novices in programming around the world, it may help frame the ways in which speakers of languages other than English confront programming languages designed around English keywords, validating the language and literacy practices they bring. While some have studied use of programming languages based on keywords or syntax of other (human) languages, such as an Arabic version of LISP (Elazhary, 2012), or localized versions of Scratch (Dasgupta & Hill, 2017), and many others have attempted to create textbooks or other curricular materials in languages other than English to support those for whom English is a challenge (e.g., Soosai Raj, Ketsuriyonk, Patel, & Halverson, 2018), the overall utility of having meaningful keywords in a programming language may be limited (Stefik, 2013). It also suggests a variety of pedagogical strategies, including inviting students to use a variety of linguistic resources to think about computer science problems, and considering, as we suggested before, what conversations are the code a part of? Seeing code as a subset of the expressions that allow students to express ideas (computational or otherwise) within a community of discourse opens possibilities for translanguaging, and it also creates opportunities for computer science education to be linked to and embedded in other subject areas (such as language arts in this example).

As much as computing literacies are shaped by digital technologies, they are also socially constructed, shaped by discourse communities that evolve over time. As more people, especially those who are members of racial, economic, and gender groups who have been excluded from and marginalized in computer science, participate in computing literacies, not only do they learn the discourses of computing communities, but they reshape and transform those communities and discourses. An approach rooted in translanguaging also opens us up to considering how CS as a community of practice, a discourse, and a literacy evolves -- as people transform it and use it for their own purposes. As we broaden participation in CS, we should expect this to happen, and indeed encourage it.

Thus, considering computer science education as a question of participating in literacies can also help us think about equity and justice in the field. Firstly, it helps us understand the social, linguistic, and shared construction of knowledge in computer science as it takes place in classrooms, communities and professional contexts, and second, it helps us notice the ways that discourse and language are used to keep people out of the field. It is important to note that tech-speak, jargon etc. can be as exclusionary as requiring command of standard English in CS fields. Our data helps demonstrate how translanguaging and the use of a broad repertoire of language resources can be mobilized to do CS, which has implications for monolingual learners as much as emergent bilinguals. Much as translanguaging pedagogy has supported a political agenda of inclusion by explicitly valuing varied and diverse language repertoires for participation in school literacies and beyond, we believe that translanguaging pedagogy and a literacy stance towards computing education can likewise support learners mobilizing their varied and diverse language backgrounds to participate in CS for all. When we ask: What conversation is this code a part of, we must also ask: Who participates in conversations code is a part of?

Acknowledgements

Sponsored by the National Science Foundation under grants CNS-1738645 and DRL-1837446. Views expressed are not those of NSF. We gratefully acknowledge the work of Dr. Kate Menken in conceptualizing the PiLaCS project, the participation of the educators and students in our partnership, and our colleagues who participated in the project and helped frame this work. We are also grateful to the reviewers whose

insightful comments pushed our thinking.

References

Alvarez, S. (2014). Translanguaging tareas: Emergent bilingual youth as language brokers for homework in immigrant families. *Language Arts*, *91*(5), 326.

American Council on the Teaching of Foreign Languages. (2017, January 26). Supporting the Study of World Languages and Computer Science. Retrieved August 31, 2018, from American Council on the Teaching of Foreign Languages website: https://www.actfl.org/news/position-statements/supporting-the-study-world-languages-and-computer-science

Arnaut, K., Blommaert, J., Rampton, B., & Spotti, M. (2015). *Language and Superdiversity*. New York & London: Routledge.

Blackledge, A. & Creese, A. (2017). Translanguaging and the body. International Journal of Multilingualism, 14(3), 250-268.

Blommaert, J. (2010). *The Sociolinguistics of Globalization*. Cambridge: Cambridge University Press.

Bourdieu, P. (1991). Language and symbolic power. Harvard University Press.

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Burke, Q. (2012). The markings of a new pencil: Introducing programming-as-writing in the middle school classroom. *Journal of Media Literacy Education*, *4*(2), 121–135.

Celic, C., & Seltzer, K. (2013). *Translanguaging: A CUNY-NYSIEB Guide for Educators*. Retrieved from CUNY- New York State Initiative on Emergent Bilinguals website: http://www.nysieb.ws.gc.cuny.edu/files/2013/03/Translanguaging-Guide-March-2013.pdf

Daniel, S. M., & Pacheco, M. B. (2015). Translanguaging Practices and Perspectives of Four Multilingual Teens. *Journal of Adolescent & Adult Literacy*. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/jaal.500/pdf

Dasgupta, S., & Hill, B. M. (2017). Learning to Code in Localized Programming Languages. Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, 33–39. https://doi.org/10.1145/3051457.3051464

DiSessa, A. A. (2001). *Changing Minds: Computers, Learning, and Literacy*. MIT Press.

Eglash, R., Gilbert, J. E., & Foster, E. (2013). Toward culturally responsive computing education. *Communications of the ACM*, *56*(7), 33–36.

Elazhary, H. (2012). Facile programming. *The Internatioanl Arab Journal of Information Technology*, *9*(3), p. 256-261.

Faltis, C. (2014). Language, language development and teaching English to emergent bilingual users: Challenging the common knowledge theory in teacher education & k-12 school settings. *Association of Mexican American Educators Journal*, *7*(2). Retrieved from http://amaejournal.utsa.edu/index.php/amae/article/view/123

Flores, N., & Rosa, J. (2015). Undoing Appropriateness: Raciolinguistic Ideologies and Language Diversity in Education. *Harvard Educational Review*, *85*(2), 149–171. https://doi.org/10.17763/0017-8055.85.2.149

Freire, P. (1985). Reading the World and Reading the Word: An Interview with Paulo Freire. *Language Arts*, *62*(1), 15–21. Retrieved from JSTOR.

García, O. (2011). *Bilingual Education in the 21st Century: A Global Perspective*. John Wiley & Sons.

García, O., Flores, N., & Woodley, H. (2012). Transgressing Monolingualism and Bilingual Dualities: Translanguaging Pedagogies. In Y. Androula (Ed.), *Rethinking Education, Volume 5: Harnessing Linguistic Variation to Improve Education* (Vol. 5). Bern: Peter Lang AG.

García, O., Ibarra Johnson, S., & Seltzer, K. (2017). *The Translanguaging Classroom: Leveraging Student Bilingualism for Learning*. Philadelphia: Caslon Publishing.

García, O., & Kleifgen, J. A. (2018). *Educating Emergent Bilinguals: Policies, Programs, and Practices for English Language Learners* (2nd Edition). Teachers College Press.

García, O., & Kleyn, T. (Eds.). (2016). *Translanguaging with Multilingual Students: Learning from Classroom Moments*. New York: Routledge.

García, O., & Leiva, C. (2014). Theorizing and enacting translanguaging for social justice. In *Heteroglossia as practice and pedagogy* (pp. 199–216). Retrieved from http://link.springer.com/chapter/10.1007/978-94-007-7856-6_11

García, O., & Li Wei. (2014). *Translanguaging: Language, bilingualism and education*. London, United Kingdom: Palgrave Macmillan Pivot.

Gee, J. (2015). Social Linguistics and Literacies: Ideology in Discourses (Fifth Edition). New York, NY: Routledge.

Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: The exploring computer science program. *ACM Inroads*, *3*(2), 47–53.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43. https://doi.org/10.3102/0013189X12463051

Guo, P. J. (2018). Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 1–14. https://doi.org/10.1145/3173574.3173970

Hacking, I. (1986). Making up people. In T. C. Heller, M. Sosna and D. E. Wellbery (with A. I. Davidson, A. Swidler and I. Watt) (eds), Reconstructing Individualism: Autonomy, Individuality, and the Self in Western Thought. Stanford, CA: Stanford University Press, pp. 222–36.

Hacking, I. (1994). The looping effects of human kinds. In D. Sperber, D. Premack and A. J. Premack (eds), Causal Cognition: A Multidisciplinary Approach. Oxford: Clarendon Press.

Hirotaka, A. (2014, January 30). ANYBODY CAN LEARN — Computer Science is Not a Foreign Language. Retrieved December 10, 2018, from Code.org website: https://blog.code.org/post/75129943201/language

Hoadley, C. (2004). Fostering collaboration offline and online: Learning from each other. In M. C. Linn, E. A. Davis & P. L. Bell (Eds.), Internet Environments for Science Education. Mahwah, NJ: Lawrence Erlbaum Associates.

Hoadley, C., & Favaro, S. (2015). Digital literacy in higher education. In J. M. Spector (Ed.), *The SAGE Encyclopedia of Educational Technology* (1 edition, pp. 221–223). Los Angeles: SAGE Publications, Inc.

Hopewell, S. (2011). Leveraging bilingualism to accelerate English reading comprehension. *International Journal of Bilingual Education and Bilingualism, 14*(5), 603–620.

Hornberger, N. (Ed.). (2003). *Continua of Biliteracy: An Ecological Framework for Educational Policy, Research, and Practice in Multilingual Settings*.

Himley, M., & Carini, P. F. (2000). From Another Angle: Children's Strengths and School Standards: the Prospect Center's Descriptive Review of the Child. New York, NY: Teachers College Press.

Hua, Z., Wei, L., & Jankowicz-Pytel, D. (2019). Translanguaging and embodied teaching and learning: Lessons from a multilingual karate club in London. *International Journal of Bilingual Education and Bilingualism*, *0*(0), 1–16. https://doi.org/10.1080/13670050.2019.1599811

Irwin, A. (2002). *Citizen Science: A Study of People, Expertise and Sustainable Development*. https://doi.org/10.4324/9780203202395

Jørgensen, J. N. (2008). Polylingual Languaging Around and Among Children and Adolescents. *International Journal of Multilingualism*, *5*(3), 161–176. https://doi.org/10.1080/14790710802387562

Kafai, Y. B., & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming* (M. Resnick, Ed.). Retrieved from http://www.jstor.org/stable/j.ctt9qf8rk

Kibler, A. (2010). Writing through two languages: First language expertise in a language minority classroom. *Journal of Second Language Writing*, *19*(3), 121–142. https://doi.org/10.1016/j.jslw.2010.04.001

Knobel, M., & Lankshear, C. (2007). *A New Literacies Sampler*. New York: Peter Lang International Academic Publishers.

Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, *27*(2), 97–111. https://doi.org/10.1093/comjnl/27.2.97

Kusters, A., Spotti, M., Swanwick, R., Tapio, E. (2017). Beyond languages, beyond modalities: Transforming the study of semiotic repertoires.

Lambert, W. E. (1974). Culture and Language as Factors in Learning and Education. In F. E. Aboud & R. D. Meade (Eds.), *Cultural factors in learning and education*. Bellingham, Washington: 5th Western Washington Symposium on Learning.

Lave, J., & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press.

Lave, J., & Wenger, E. (2001). Legitimate peripheral participation in communities of practice. In J. Clarke, A. Hanson, & R. Harrison (Eds.), *Supporting Lifelong Learning*: *Vol. Volume 1: Perspectives on Learning*. https://doi.org/10.4324/9780203996287-11

Li Wei. (2011). Moment Analysis and translanguaging space: Discursive construction of identities by multilingual Chinese youth in Britain. Journal of Pragmatics, 43(5), 1222–1235. https://doi.org/10.1016/j.pragma.2010.07.035

Li Wei. (2018). Translanguaging as a Practical Theory of Language. *Applied Linguistics*, *39*(1), 9–30. https://doi.org/10.1093/applin/amx039

Mazak, C. M., & Donoso, C. H. (2015). Living the bilingual university: One student’s translanguaging practices in a bilingual science classroom. In Transcultural Interaction and Linguistic Diversity in Higher Education (pp. 255–277). Springer.

Natural-language understanding. (2016). In A. Butterfield & G. Ekembe Ngondi (Eds.), A Dictionary of Computer Science (7th Ed.). Oxford, UK: Oxford University Press.

New London Group. (1996). A pedagogy of multiliteracies: Designing social futures. *Harvard Educational Review*, *66*(1).

Otheguy, R., García, O., & Reid, W. (2015). Clarifying translanguaging and deconstructing named languages: A perspective from linguistics. *Applied Linguistics Review*, *6*(3), 281–307. https://doi.org/10.1515/applirev-2015-0014

Pal, Y., & Iyer, S. (2015). Classroom Versus Screencast for Native Language Learners: Effect of Medium of Instruction on Knowledge of Programming. Proceedings of the 2015 ACM

Conference on Innovation and Technology in Computer Science Education, 290–295. https://doi.org/10.1145/2729094.2742618

Palmer, D. K., Martínez, R. A., Mateus, S. G., & Henderson, K. (2014). Reframing the Debate on Language Separation: Toward a Vision for Translanguaging Pedagogies in the Dual Language Classroom. *The Modern Language Journal*, *98*(3), 757–772. https://doi.org/10.1111/modl.12121

Papert, S. (1980). *Mindstorms.* New York: Basic Books.

Paris, D., & Alim, H. S. (2014). What are we seeking to sustain through culturally sustaining pedagogy? A loving critique forward. *Harvard Educational Review*, *84*(1), 85–100.

Pontier, R., & Gort, M. (2016). Coordinated Translanguaging Pedagogy as Distributed Cognition: A Case Study of Two Dual Language Bilingual Education Preschool Coteachers' Languaging Practices During Shared Book Readings. *International Multilingual Research Journal*, *10*(2), 89–106.

Poza, L. E. (2018). The language of ciencia: Translanguaging and learning in a bilingual science classroom. International Journal of Bilingual Education and Bilingualism, 21(1), 1–19. https://doi.org/10.1080/13670050.2015.1125849

Rawal, H., De Costa, P., & Tian, Z. (2019, March 10). Multimodality in Action: Navigating the Complexities of Science in a Multilingual Classroom. 2019 Conference of the American Association for Applied Linguistics (AAAL).

Reestman, K., & Dorn, B. (2019). Native Language's Effect on Java Compiler Errors. Proceedings of the 2019 ACM Conference on International Computing Education Research, 249–257. https://doi.org/10.1145/3291279.3339423

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for All. *Commun. ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Roberts, D. A., & Bybee, R. W. (2014). Scientific Literacy, Science Literacy, and Science Education. In N. G. Lederman & S. K. Abell (Eds.), Handbook of Research on Science Education Volume II (pp. 545-558): Taylor and Francis.

Scott, K. A., Clark, K., & White, M. A. (2013). COMPUGIRLS’ standpoint: Culturally responsive computing and its effect on girls of color. *Urban Education*, *48*(5), 657–681.

Scott, K. A., Sheridan, K. M., & Clark, K. (2015). Culturally responsive computing: A theory revisited. *Learning, Media and Technology*, *40*(4), 412–436.

Seltzer, K. (2019). Performing ideologies: Fostering raciolinguistic literacies through role-play in a high school English classroom. Journal of Adolescent & Adult Literacy, XX.

Soosai Raj, A. G., Ketsuriyonk, K., Patel, J. M., & Halverson, R. (2018). Does Native Language Play a Role in Learning a Programming Language? Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 417–422. https://doi.org/10.1145/3159450.3159531

Stefik, A. & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education, 13*(4), 19:1-19:40.

Street, B. V. (1984). *Literacy in theory and practice* (Vol. 9). New York: Cambridge University Press.

Street, B. V. (1993). *Cross-cultural approaches to literacy*. New York: Cambridge University Press.

Suárez, E. A. (2017). Designing Science Learning Environments That Support Emerging Bilingual Students to Problematize Electrical Phenomena [University of Colorado at Boulder]. https://scholar.colorado.edu/concern/graduate_thesis_or_dissertations/fb494855q

Valencia, R. R. (2010). *Dismantling contemporary deficit thinking: Educational thought and practice*.

Vee, A. (2017). *Coding Literacy: How Computer Programming Is Changing Writing.* Cambridge, MA: MIT Press.

Vogel, S., Ascenzi-Moreno, L., & García, O. (2018). An Expanded View of Translanguaging: Leveraging the Dynamic Interactions Between a Young Multilingual Writer and Machine Translation Software. In J. Choi & S. Ollerhead (Eds.), Plurilingualism in Teaching and Learning: Complexities Across Contexts (pp. 89–106). Taylor & Francis Ltd. https://academicworks.cuny.edu/cgi/viewcontent.cgi?article=1494&context=gc_pubs

Vogel, S., Ascenzi-Moreno, L., Hoadley, C., & Menken, K. (2018, May 3). Leveraging multilingualism to support computer science education through translanguaging pedagogies. Translanguaging: Opportunities and Challenges in a Globalized World. Colloque du CCERBAL (Canadian Centre for Studies and Research on Bilingualism and Language Planning) 2018 Conference, University of Ottawa. https://ccerbal2018.sciencesconf.org/

Vogel, S., & García, O. (2017). Translanguaging. In G. Noblit & L. Moll (Eds.), Oxford Research Encyclopedia of Education. Oxford University Press. http://education.oxfordre.com/view/10.1093/acrefore/9780190264093.001.0001/acrefore-9780190264093-e-181

Vossoughi, S., Hooper, P. K., & Escudé, M. (2016). Making Through the Lens of Culture and Power: Toward Transformative Visions for Educational Equity. *Harvard Educational Review*, *86*(2), 206–232. https://doi.org/10.17763/0017-8055.86.2.206

Wing, J. M. (2006). Computational Thinking. Commun. ACM, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215