Limudei Code-ish:
Using Computer Science Cognitive Skills To Deepen Higher Order Thinking in Jewish Education

Presented to the S. Daniel Abraham Honors Program in Partial Fulfillment of the Requirements for Completion of the Program

Stern College for Women
Yeshiva University

May 6, 2020

Sarah Adena Engel

Mentors: Professor Alan Broder and Dr. Deena Rabinovich

Education is an ever-evolving field, adapting to the new concerns of each generation to best prepare its youth for their future. In the past half-century, the explosion of computers and increased accessibility of technology has forced educators to think deeply about the ways technology can be, and should be, integrated into a classroom setting. As more and more studies emerge showing the effect of technology and computational thinking on brain development, even the most seasoned educators need to adjust for the new ways of thinking their students are developing through their exposure to computers. Traditionally, we see the integration with general studies which must adapt to account for both integration of technology, and an awareness of students' technologically-based critical thinking skills. The Judaic studies classroom, however, should also adapt given an awareness of the modern technology students are exposed to and find ways to incorporate the new critical thinking skills their students are developing as a result of computer accessibility in the 21st century. Currently, Judaic studies classrooms use computers as a resource. But the focus of this thesis is to demonstrate an ever deeper connection that can be made tapping into critical thinking. While the two domains of computer science and Judaic studies may seem very different, the steps involved in thinking are actually quite similar and developing the skills in one can help develop the skills in the other.

While many Judaic studies classrooms have began to adopt technology as a means for streamlining certain logistics in the classroom, such as using an online resource to provide students with free copies of a rare text, or using multimedia to play music or videos relating to curriculum content, this thesis will examine the use of technology-based critical thinking skills in a Judaic studies classroom. In order to analyze this, one must ask the following

question: how does the logic utilized in various Jewish exegetical and legal works mirror logical cognitive patterns and processes used in the development of new technologies? For if one recognizes that the Torah, Talmud and canon of Jewish exegetical and legal works utilize many of the same logical processes used in disciplines like engineering and computer science, one can use the thinking behind technology development to deepen the learning experience of a student in the Judaic studies classroom[1]. An integrated understanding of the parallels in the underlying logic of both disciplines will enable students to apply the logic of modern computer science to gain an appreciation for the ancient wisdom and existing logical structure of ancient Jewish texts.

The goal of this thesis is to first, establish that the logical parallels between the underlying structure of computer science and Jewish text do exist. Once this foundation has been established, this paper will explore educational theory behind a constructivist model of computer science education based on the research of Seymour Papert. This model will then be applied to enhance learning in the Judaic studies classroom given the already-established parallels between computer science and Jewish text. After establishing the parallels, and defining educational criteria, this thesis will explore current models of computer integration

---

[1] Seymour Papert, in his book, *Mindstorms* (which will be quoted extensively throughout the remainder of this thesis) argues about a similarity between the study of math and literature. However, he also gives his readers a critical caveat which must be given here as well when I argue about a logical parallel embedded in both Torah study and computer science.

Papert writes that "I do not wish to reduce mathematics to literature or literature to mathematics. But I do want to argue that their respective ways of thinking are not as separate as is usually supposed." (Papert, 39). Likewise, in this thesis, I am *not in any way* intending to reduce the ever-infinite and Divinely-given logic of the Torah to the human-developed and very limited world of computer science. Rather, I simply hope to point out the parallels in some of the logical structures between the two domains, and argue that using our understanding of humanly developed and limited computer science as a way to create new intellectual models to enhance student comprehension of Jewish texts, we can further deepen our students' understanding of the Divine word of G-d.

in a Judaic studies classroom, and determine whether or not they are up to par with the criteria established by Papert. Finally, this thesis will present an original model for integration I developed in my research with the Sefaria Project.

Logical Parallels shared by Computer Science and Jewish Legal Texts

Jewish legal texts in general tend to be highly structured, logical and sometimes difficult for even the most seasoned student to grasp. Many of these texts, on a cognitive level, have very similar underlying logical processes to those engineers use when thinking about problem solving. For example, the Talmudic passages giving highly specific cases to derive an overarching law can be conceptualized as an analysis of "big data" with boundary conditions, or a passage with a seemingly convoluted disagreement amongst sages can be easily broken down into an outline using conditional statements (a fundamental core of almost all computer programs which allows the program to make decisions). By drawing on these parallels in a futuristic world where perhaps the Judaic studies student enters the *Beit Midrash* well-versed in programming logic, but a novice to Talmudic study, these parallels can help create not only a bridge, but a familiar model of critical thinking to help aid the student as they begin a journey of deep engagement with Jewish texts.

While both a file of computer code and the Talmud are terse documents, using few words to describe complicated and highly technical realities, the parallels in logic far outweigh this superficial commonality of terseness. Both a computer scientist developing code and a student of Talmud trying to understand a difficult passage are starting at one level of understanding, and are both being asked to move to a different level of conceptualization.

A student decoding a passage of Talmud starts by identifying keywords, translating, to hopefully move from the micro level to the macro level, concluding by establishing a larger framework for the overall argument[2]. While computer scientists also have to shift between micro and macro levels, they work in an opposite direction. A computer scientist starts by facing a large problem, and then needs to break it down and disambiguate all of its nuanced parts into objective sub-problems understandable to a computer. When using or reviewing code written by others, a computer scientist still takes a "top down" approach, looking to first uncover the larger structure before delving into the nuanced details of the code. The goal of a computer scientist is to take a macro level challenging problem, and subdivide into a series of concise and descriptive micro level keywords. While the processes of code development and understanding a passage of Talmud move in opposite cognitive directions, with a Talmudist beginning at the micro level and moving to the macro level, and a computer scientist beginning at the macro level, and implementing the problem in the micro level, both share a need for critical thinking skills, and the ability to conceptualize problems on multiple levels almost simultaneously. If educators were to be able to harness the specific skills being used in both domains, perhaps that, in addition to the deeper parallel logic embedded in both fields, would allow for both fields to help each other, gaining from the shared structured conceptualization skills.

[2] It can be argued that a truly expert student of Talmud might actually work more like a computer scientist, moving in a "top down" manner to conceptualize the given problem. Once a student of Talmud is no longer deterred by the challenging vocabulary of the text, they often will approach a given passage with an overall understanding of the problem, and then work down from the macro to the micro-level, understanding specific nuances, mechanisms and applications of the given argument. Either way, both the Talmudist and the computer scientist must become exceptionally familiar with conceptualizing problems in different dimensions, and being able to fluidly move between the macro-level of understanding to the micro-level of understanding, and then back again.

One example of the conceptualization of a Talmudic passage providing numerous case studies as parallel to the logic behind data analysis, one must look at a specific example from the text. The first *Mishna* in the tenth chapter of *Masechet Pesachim* states as follows: "On the eve of *Pesach* close to *minchah* one may not eat until nightfall. Even the poorest person in Israel must not eat [on the night of *Pesach*] until he reclines. And they should give him not less than four cups [of wine], and even from the charity plate." (Sefaria).

The above *Mishna*, while containing some directives such as "one may not eat until nightfall" or "must not eat until he reclines" is also providing two cases, the case of the average Jew, and the case of the "poorest person in Israel". A simple data analysis program will intake a series of raw data points, and analyze certain trends or commonalities between the different values to try and determine the underlying trends which shape the decisions and habits of the items being studied. The Talmud, perhaps a bit like a data analysis program, will take the series of recorded data points from the *Mishna* and will look at them on the whole, and attempt to deduce the underlying logic or law which is forcing the cases to manifest themselves in that way.

The Talmudic passage which explores the cases presented in the aforementioned *Mishna* goes into a very lengthy discussion of the laws of *Shabbat* and *Pesach*, but the very first part of the corresponding Talmudic extrapolation is just a small example of the sages trying to deduce the underlying law which prompted the *Mishna* to record the given cases in that way. Page 99b of the Talmud in *Masechet Pesachim* states:

> "The Gemara expresses surprise at the mishna's statement that one may not eat on Passover eve from the time that is adjacent to *mincha*. Why discuss this *halakha* particularly with regard to the eves of Passover? Even on the eves of Shabbat and other Festivals it is also prohibited to eat in the late afternoon, as it was taught in a *baraita*: A person should not eat on the eves of Shabbat and Festivals from *minḥa*

> time onward, so that he will enter Shabbat when he has a desire to eat and he will enjoy the Shabbat meal; this is the statement of Rabbi Yehuda…" (Sefaria)**.**

In the above passage, the Talmud uses the formulation of the case studies presented in the *Mishna* of specifically referring to *Erev Pesach* to contrast with other data points which hold the same law to be true even for *Erev Shabbat*. Using this specific data point of not eating on *Erev Pesach* and contrasting it with an existing conflicting data point (which makes this one seem unnecessary and repetitive) the Talmud goes on to develop a complex approach to the differences between *Erev Pesach* and *Erev Shabbat* in terms of laws of eating in the afternoon.

A second, and perhaps slightly more obvious example of a Talmudic discussion with a data analysis pattern can be found in *Masechet Sotah* 4a and 4b. In this passage, the sages provide a series of seemingly unrelated examples to help define the time needed in seclusion with a warned man to render a woman prohibited to her husband[3]. There are multiple advantages to conceptualizing these examples as data points and boundary cases which flesh out the specific parameters of this halachic time constraint. Not only does it make the passage easier to comprehend, but it allows for the student to see seemingly arbitrary definitions of timing as highly functional, logical and relevant to determining the specific amount of time the woman must be in seclusion with her husband before becoming prohibited. Over the aforementioned two pages of Talmud, multiple sages suggest different time definitions, and

[3] The prohibition of a Sotah occurs when a woman, suspected of promiscuous activity with another man is warned by her husband (under very specific and limited criteria outlined by Jewish law) not to be in seclusion with this specific other man. If the woman, after the warning, is found in a specific form of seclusion with this other man for a certain amount of time (as is being defined by the Talmudic passage quoted here) she is brought to the Temple, and undergoes a ritual process to try and determine her innocence (if she still claims to be innocent of adultery with this other man).

then each of these definitions are challenged and essentially tested for boundary conditions if one conceptualizes the debate as data analysis of a sort. The passage reads:

> "The *baraita* clarifies: And what is the measure of seclusion...
> The *baraita* quotes several practical examples of this period of time… Rabbi Elazar ben Yirmeya says: This is equivalent to the time needed for a weaver to tie a string. Rav Ashi asks: Is this speaking of where the ends of the string to be tied are far apart from each other, or is it speaking of where they are near to each other?..." (Sefaria)

One can look at just this first part of the Talmudic passage, and see a computer science-like pattern emerging, of a proposed data point subsequently being tested for boundary cases by Rav Ashi. This continues throughout the rest of the section, with other "data points" being suggested by other sages, and Rav Ashi continuously testing for boundary cases. This passage continues:

> "...Ḥanin ben Pineḥas says: This is equivalent to the time that a woman may need to extend her hand into her mouth to remove a wood chip from between her teeth. Rav Ashi asks: Is this speaking of a case where the wood chip is stuck between her teeth, or is it speaking of a case where it is not stuck? …
>
> ...Peleimu says: This is equivalent to the time that a woman may need to extend her hand into a basket in order to take a loaf of bread. Rav Ashi asks: Is this speaking of an occasion where the loaf adheres to the basket, or is it speaking of a case where it does not adhere? Is this speaking of a case where the basket is new, whereby the tips of the shoots forming the basket might restrain the loaf, or this speaking of where the basket is old and smooth, enabling easy removal? Is this speaking of a case where the loaf is hot and therefore softer and may adhere to the basket, or is this speaking of a case where the loaf is cold and easily removed? Is this speaking of a case where the loaf is made of wheat, which is slippery and takes longer to remove, or is this speaking of a case where the loaf is made of barley, which is easily removed? Is this speaking of a case where the loaf is soft, so that it may catch upon the side of the basket, or a case where the loaf is hard, where this is not a concern?..." (Sefaria)

In the above excerpt from *Masechet Sotah*, three sages suggest time limits for the measure of seclusion needed to render a woman a prohibited *Sotah*, and Rav Ashi challenges their assertions by testing edge cases and looking for underlying principles in common. By

viewing all of the proposed cases as data points, and Rav Ashi as a function serving to test for their edge cases, the seemingly arbitrary and disjointed cases provided in the Talmud become highly relevant and necessary for the definition of the time at hand. Seeing the Talmud in this light allows for a student to see the relevance in a seemingly antiquated debate, and understand the overall structure of the conversation as highly necessary for the definition of the matter at hand.

Beyond a data analysis pattern, a second logical structure at the core of computer science which appears frequently throughout the pages of the Talmud is the conditional statement. A conditional statement evaluates a series of provided conditions, and if certain criteria are met, the computer will execute a corresponding specific component of the program. The structure of these statements is often in a series of "if", "else if" and "else" bracketed portions of code, which evaluate the given conditions and then execute their internal code only if the criteria are met. So for example, if someone wrote a program to distribute alcoholic beverages, the computer might first check if the age of the customer is greater than 21, and if so, distribute a beverage to the customer. Else, if the customer is (as implicitly presumed by the computer) of an age less than 21, the computer will execute a different component of the program which may refrain from serving an alcoholic beverage to a customer below the legal drinking age in the United States.

The Talmud will frequently explain the nuances of a given situation, and the different actions which one will take contingent on the circumstances of said situation meeting certain criteria in a series of "if" statements. These Talmudic statements almost precisely parallel the

way conditional statements are conceptualized in computer science. A clear example of this appears in the following passage, from *Masechet Avodah Zarah* page 7a:

> "In a situation where there were two Sages sitting together and one deems an item impure and the other one deems it pure, or if one deems it prohibited and the other one deems it permitted, the questioner should proceed as follows: If one of the Sages was superior to the other in wisdom and in number, one should follow his ruling, and if not, he should follow the one who rules stringently. Rabbi Yehoshua ben Korḥa says: If the uncertainty exists with regard to a Torah law, follow the one who rules stringently; if it exists with regard to a rabbinic law, follow the one who rules leniently. Rav Yosef said: The *halakha* is in accordance with the opinion of Rabbi Yehoshua ben Korḥa." (Sefaria)

While at first, the above Talmudic passage may seem dense and complicated, one can easily unpack the various ways a "questioner" should proceed by identifying the conditional statement structure embedded in the text. By centering an outline of this passage around the "if" and "else" statements in the text itself, one can more clearly view the nuances of the law in the case of a "questioner" receiving conflicting views. In fact, if the student is familiar enough with the basic tenets of conditional statements of computer science, they could even outline this Talmudic passage in pseudo-code, as seen in Figure 1 on the left. Once again, by identifying the parallel logical structures between computer science and a Talmudic passage, a student can hold more information in their head in a more succinct way. Drawing on these

```
# Original proposal by the Gemara
if sage1 > sage2:
    follow sage1
elif sage2 > sage1:
    follow sage2
else:
    follow stricter approach
# Rabbi Yehoshua ben Korha's approach
# (which Rav Yosef says we ultimately follow)
if sage1 > sage2:
    follow sage1
elif sage2 > sage1:
    follow sage2
else:
    if issue.status == torahLevel:
        follow stricter approach
    elif issue.status == rabbinicLevel:
        follow lenient approach
```

Figure 1

parallels also allows the student to condense vast amounts of material into easily chunk-able and logical pieces.[4]

A third comparison between the inherent logic of the Jewish legal system and computer science is the understanding of the importance of minutiae. Often, novice students of Jewish law complain about the minutiae embedded in every detail of *halacha*, for why should one care so much about the order in which one lights the *Chanukah* candles, or how exactly one should move when shaking a *Lulav*? By understanding the *halacha* as almost an algorithm for *avodat Hashem* (worship of G-d), a student with a deep understanding of the necessary precision in a given algorithm will better understand the logic and importance of every minute detail within Jewish law. By drawing this parallel between the critical nuances of *halacha* and the critical nuances of an algorithm, educators can instill within the 21st century student a deeper confidence in Jewish law, and a staunch assurance of the necessity of even the seemingly most arbitrary details.

A specific example of this can be found in an analysis of almost any algorithm, and comparing the necessary attention to details with almost any section of the *Shulchan Aruch* detailing the tremendous nuances of any Jewish law. For illustration, consider the Huffman Coding data compression algorithm, which takes a text and compresses it into a smaller file by building a tree data structure based on frequency of a given character's appearance (Encyclopedia Britannica)[5]. This underlying tree data structure is built by repeatedly finding the character which appears in the given text least frequently, and then the subsequent least

[4] While an easy comparison between the two fields would be looking at their large vocabulary of instructive key words, this thesis aims to move beyond that and focus on the similar logical processes needed to conceptualize both domains.

[5] Note: Huffman Coding is a tremendously complex and detailed algorithm, and is not given full technical treatment in this paper as it is not the main focus.

frequently-appearing character, and so forth. Depending on how the algorithm is implemented, different ways of handling this subsequent minimum character storage can vastly affect the efficiency of this algorithm, and how much time it takes to run on a large body of text[6]. These tiny nuances, relating not to the algorithm as a whole (but simply the storage of intermediary values in order to compute a data structure which is used to then encode the given characters for compression) have a vast impact on the time it takes for this algorithm to run. When students are exposed to thinking in this manner, and begin to understand the vast differences little (seemingly-arbitrary) details can have on the functioning of a real-life algorithm, they begin to adapt to the pattern of thinking where they no longer challenge small details as being unnecessary. Rather, students become accustomed to engaging *with* details when they appear, because clearly they are extremely significant both in algorithms, and in Jewish law.

Using Computers to Bolster Critical Thinking in a Constructivist Judaic Studies Classroom

While many logical parallels do exist within the structure of Jewish law and computational thinking, how can one can use this overlap to enhance Judaic studies in the 21st Century classroom? In order to begin to think deeply about integration, one must first take a step back and understand more generally the educational theory behind the integration of computers and computational thinking in a pedagogical setting.

---

[6] For those familiar with the technicalities of time complexities and big-O notation, consider the following which was discussed in Professor Teitelman's Fall 2019 Algorithms course at Stern College: if one uses an ArrayList to store the the characters and their frequencies, finding the next minimum each time takes O(n^2). Using a Min-Heap reduces this time complexity to O(n*logn), whereas using two parallel queues further reduces the time complexity of storing and retrieving the next minimum frequency character to O(1).

Seymour Papert wrote the seminal work on teaching computers, and computational thinking to students, and understanding the aforementioned within the context of emerging educational theory. His book, *Mindstorms*, published at the advent of accessible computing in the 1980s, uses Piagetian educational theory to explain how the introduction of computational thinking in a classroom could reshape the way the world teaches children how to think[7]. Piagetian constructivist education proposes that "knowledge is never the result of a passive activity of receiving information, but of an active engagement with the world through manipulation of artifacts and interactions with people and contexts" (Bers, Matas, Libman 167-168). Using this theory as his foundation, Papert argued that computers have become an incredible tool which can allow students to further engage in experiential learning. By simply introducing a computer into the classroom in a calculated way, one gives the student the power of interacting with different people, things, and environments. Constructivist education at its core is about giving students the opportunity to construct their own learning, by building, thinking and interacting. Constructivists argue that a computer is not just a tool for exposing students to more material to build with, but rather computers are "also expressive, epistemological tools that can bring insights into our own thinking" (Bers, Matas, Libman 167-168).

Papert specifically wrote *Mindstorms* in part as an explanation of a new educational tool called LOGO, which provided a simplified programming language for younger students

[7] Papert spent five years at Piaget's Center for Genetic Epistemology in Geneva, and reflects at the start of his seminal work, *Mindstorms*, that constructivism involves "looking as children as the active builders of their own intellectual structures. But to say that intellectual structures are built by the learner rather than taught by the teacher does not mean they are built from nothing…. Like other builders, children appropriate to their own use materials they find about them, most saliently the models and metaphors suggested by the surrounding culture." (Papert, 19)

associated with a mini robot called “turtle” which they could program to move in different ways, and behave in response to certain commands. While some of his work is specific to this currently-outdated technology, Papert also makes broader predictions about how computing could change the way children think, if computers were to become something which was economically feasible to have in every classroom. While in the 1980s this may have felt like a futuristic dream, the relevance of his work takes an even more significant role in the current climate, where computers appear ubiquitously throughout our world, and our classrooms.

The central argument in *Mindstorms* is that while many feel “computer-aided instruction” occurs when a student is taught by a computer (for example, using a computer to access an automated online learning module), ideally education incorporating computers should allow the students to engage more actively in their own learning experience (Papert, 5). Papert calls the former “online module” model when the “computer programs the child”, but argues that the best way to use computers in education is to have the “child program the computer” and through this process develop a deep mastery over “the art of intellectual model building” (Papert, 5). His premise, as explained above, is taken from the theory of Piagetian Constructivist education, where “children [are seen] as builders of their own intellectual structures", since “all builders need materials to build with”, effective introduction of the computer into the classroom can widen the scope of accessible “building materials” for a child deepening their education, and the way they think, in the most profound way (Papert, 7). In fact, Papert argues further that children “learning to use computers can change the way they learn everything else” (Papert, 8). This is central to the

argument proposed in this paper: if we identify the similar thinking skills present in Judaic studies and computer science, and harness the "changed" way students with exposure to programming[8] are being taught to think, Jewish educators can perhaps more effectively teach their students the analysis skills necessary to truly break down a text.

How exactly does studying computer science change the way someone thinks? In addition to the obvious exposure to very concrete analytical skills, breaking down problems into smaller component solvable pieces, and learning how to think in abstractions, Papert argues that a student of computer science turns into an epistemologist. Computer science in many ways is figuring out how to take human problems, and make them understandable to a computer. In order to do this, one must first gain a very deep understanding of both the problem, and ways of thinking about the problem, in order to then procedurally "teach" the computer how to go about solving the problem. Papert writes that, "in teaching the computer how to think, children embark on an exploration about how they themselves think... Thinking about thinking turns the child into an epistemologist, an experience not even shared by most adults" (Papert, 19). By developing these meta-cognitive skills, students have a deeper self-awareness and ability to understand how different problem solving techniques are suited for different problems. Papert explains that "by deliberately learning to imitate mechanical

[8] There is a common misconception that computer programming is an extremely advanced subject, only for the most gifted and talented of students. While it is true that computer science is a very rigorous subject, and definitely challenges the students to think differently than they may have before, it is ultimately a subject like any other. Just like mathematics and English are studied by both kindergartens, and PhD students at the highest level, computer programming is accessible to students of all ages. Papert addresses this directly in *Mindstorms*, explaining as follows:

> "For some readers this might be tantamount to the suspension of disbelief… for them, programming is a complex and marketable skill acquired by some mathematically gifted adults. But my experience is very different. I have seen hundreds of elementary school children learn very easily to program, and evidence is accumulating to indicate much younger children could do so as well. The children in these studies are not exceptional..." (Papert, 13)

thinking, the learner becomes able to articulate what mechanical thinking is, and what it is not" (Papert, 27). He argues that having this self-awareness gives the student more confidence as a learner, and a "new degree of intellectual sophistication" as "work[ing] with the computer can make it easier to understand that there is such a thing as 'style of thinking'" (Papert, 27). Harnessing this new awareness and these new cognitive abilities in a Judaic studies classroom would allow students to be more aware of different kinds of texts they may be encountering, self-regulate which thinking patterns may be best suited for different logical structures that may emerge from the text, and then solve those problems step-by-step in a way that deepened their understanding of the given passage. Students who may be confused why aggadic passages in the Talmud are structurally so unlike technical halachic passages may be more comfortable understanding the diversity, and embracing the Talmud as a widely diverse text. Just like different problems in computer science may require different styles of thinking, the Talmud has different passages requiring different styles of thinking. Overall, an adoption of computer science thinking in a Judaic studies classroom can enable students to emerge more confident, more comfortable navigating different cognitive skills needed to decode different texts, and more comfortable with the notion that the canon of Jewish texts is not illogical, rather all-encompassing of different styles of thought.

Another benefit of teaching students how to program is that they gain an ability to understand abstract concepts more concretely. Computer science provides an intellectual model with which one can speak to abstract concepts in a logical way. Papert explains that his interest in computers was prompted by the notion that students would gain "from the way in which computer models seemed able to give concrete form to areas of knowledge that had

previously appeared so intangible and abstract" (Papert, 23). He proposes, based on this claim that, "children who had learned how to program computers could use very concrete computer models to think about thinking, and learn about learning" (Papert, 23). As modern Jewish educators whose students have likely been exposed to programming, or will be exposed to programming in the near future, it is critical that one capitalizes on these transferable skills in the Judaic studies classroom. So much of theology is abstract, and so much of Jewish law seems beyond young students at first glance, however, implementing the skills for building concrete models out of abstract ideas the students themselves are gaining by programming, one can more effectively teach far-reaching Judaic concepts in a concrete and understandable way.

A specific way computer science allows students to concretize abstract concepts is through the paradigm of viewing computer science as the study of developing a descriptive language for problem solving. The reason why computer science is ultimately about the ability to develop a descriptive language for problem solving is because "computers are called upon to do many things, and getting a computer to do something requires that the underlying process be described, on some level, with enough precision to be carried out by the machine…" (Papert 99-100). Papert posits that, "an important part of becoming a good learner is learning how to push out the frontier of what we can express in words… the development of descriptive languages for talking about learning" (Papert, 96). The true mark of higher order thinking as developed by many educational theorists, is the ability not just to understand information, but to synthesize the information perhaps to a point where one is

able to accurately explain it, or bring multiple sources together making critical connections.[9] Papert explains how this development of a descriptive language affects learning in the following passage from *Mindstorms*:

> "People need more structured ways to talk and think about the learning of skills. Contemporary language is not sufficiently rich in this domain. In a computer-rich world, computer languages that simultaneously provide a means of control over the computer and offer new and powerful descriptive languages for thinking will undoubtedly be carried into general culture. They will have a particular effect on our language for describing ourselves and our learning. To some extent this has already occurred. It is not uncommon for people with no knowledge of computers to use such concepts as 'input', 'output' and 'feedback' to describe their own mental processes… programming concepts can be used as a conceptual framework for learning… giving us words and concepts to describe what had previously seemed too amorphous for systematic thought" (Papert, 98).

Through studying computer science, the student begins to develop a descriptive language for being able to describe their own thinking process, and develop further into a student who possesses the aforementioned ability to discern between different "styles of thinking".

By studying computer science, and integrating computer science patterns of thought into the Judaic studies classroom, teachers have richer opportunities to provide their students with an ability for meta-cognition, and understanding the differences between different kinds of texts, and an enhanced ability for them to go about their own problem-solving process. Furthermore, studying computer science allows students with these enhanced meta-cognition skills to more gracefully embrace complexity. Students of computer science gain the knowledge as to how to navigate complexity within a given problem, a skill which is also highly necessary for advancing the level of Jewish studies in the classroom. Papert argues that exposure to the "internal intelligibility of the computer world offers children the

[9] See pages 19-20 for further elaboration on this idea, specifically in reference to Bloom's Taxonomy.

opportunity to carry out projects of greater complexity… children are able to acquire a feel for complexity" (Papert, 118). By failing to expose their students to the complexities embedded within Judaic studies, educators are robbing their students of opportunities to use their full intellectual capacities within the classroom. First, educators are ignoring enhanced thinking skills students are developing in other classes. Secondly, educators are not drawing upon these skills for encountering complexity in the Judaic studies classroom, which can lead students to see Jewish studies as not-intellectual, and overly simplistic. If Jewish studies teachers were able to draw on the intellectual skills for handling nuance, students would be more intellectually engaged in the Jewish texts studied in class.

Beyond the thinking skills one gains by studying computer science, one also gains certain soft-skills also critical for the learning process. Many students are afraid to take intellectual risks, or fail at a learning endeavor[10]. These internal defense mechanisms end up holding the student back severely, as they become accustomed to never trying anything new unless they are guaranteed to succeed. Papert explains that this binary of students feeling they either "got it [right]" or "got it wrong" inhibits learning and mitigates the student's ability to reach their full academic and intellectual potential (Papert, 23). In contrast to most other disciplines, when one learns to program, "you almost never get it right the first time. Learning to be a master programmer is learning how to become highly skilled at isolating…

[10] Reshma Saujani, founder of Girls Who Code (a non-profit organization teaching young girls coding skills), posits in her 2016 TedTalk that society is conditioning men to be brave, and women to be perfect. Since the trial and error process of coding is the opposite of instant perfection, many girls are intensely deterred from computer science from their first exposure. Through her organization, she works to combat this by teaching young girls not only how to code, but also how to be brave by getting comfortable with failure. Based on her thesis, it is possible to assume that many girls who have exposure to Jewish text study may be especially deterred by the complexity, trained to focus on perfection as opposed to bravery in the face of challenge. An added bonus of using computer science in the Judaic studies classroom is that perhaps female students who especially struggle with perfectionism may become more accustomed to exhibiting perseverance in the face of challenging Jewish texts.

the parts that keep the program from working." (Papert, 23). Similarly, when studying a very complex Jewish text, often encrypted in difficult Hebrew or Aramaic, many students are tempted to give up immediately, not accustomed to the process of "debugging" inherent to computer science, consistently returning to a difficult problem until it is eventually solved. Papert explains that by studying computer science, students gain the skills to realize that it's not about whether the solution is "right or wrong, but... is [it] fixable." (Papert, 23). Computer science forces students to develop grit as learners, and to not give up even when the problem solving becomes difficult[11]. Transferring these skills explicitly to the Judaic studies classroom would allow students to gain more mastery over text, normalizing the fact that one is expected to dwell over difficult passages of text again and again, not getting it right the first time, but sitting with it until they have "solved" the problem. These soft-skills that come parceled to the study of computer science are just as necessary in Judaic studies classrooms. If taught to students, these skills could turn students into both masters over the text and give them the soft-skills necessary to fulfill the ultimate goal of most Jewish educational institutions: cultivating their students into life-long learners of Torah.

What about the educators who claim to not be "tech-savvy" or adept at using technology in the classroom, or in their personal lives? According to Papert, one cannot "underestimate the potential effect that a massive presence of computers and other interactive objects might have on children" (Papert, 25). He argues that especially for those accustomed to viewing learning as a process which can only occur when a teacher stands at the front of the room and lectures, utilizing computers in the classroom in an interactive way seems

[11] For a more comprehensive study of grit, see Dr. Angela Duckworth's work, including her book *Grit: The Power of Passion and Perseverance*

purposeless. However, one cannot ignore the computer revolution happening in the world, nor can one ignore the impact it is having on contemporary students. Papert argues that, "The educator as an anthropologist must work to understand which cultural materials are relevant to intellectual development… [therefore the educator] needs to understand which trends are taking place in the culture" (Papert, 32). If an educator chooses to ignore the computer revolution, they are not only removing themselves of opportunities to provide their students with relevant intellectual development, but they also make their curriculum more foreign to students embedded in modern culture.

Does this mean that in order to be a relevant educator, one must know the popular music and social media platforms? While that may help a teacher make connections with students and their interests, Papert is arguing for something much deeper than assigning students to write a song about a Jewish holiday to a popular rap tune. Papert is suggesting that being a culturally relevant educator in its truest sense means thinking about the way people are thinking in the modern world, and what materials they are using to achieve such thinking (which in this specific case, means adapting to the way the computer has shifted the way humans think). The role of the educator must "take the form of working with these trends" to accommodate for the deep intellectual shifts in the way humanity perceives problem solving, and applying their material in such a way that it becomes intellectually viable and relevant for students growing up in an age where, because of the computer, humans are internally processing information differently (Papert, 32). Applying this mentality to Jewish studies in particular would help combat an unfortunate reality of students feeling their general studies to be more intellectually stimulating than their Judaics (which

may be a function of the intellectual and cultural relevance with which its taught) and therefore walking away from a Yeshiva education believing Shakespeare to be wiser than the Talmud. If the child is taught Talmud in a manner that they have been culturally molded to intellectually perceive as rigorous, they will walk out of their Yeshiva education inspired to be a life-long learner. The student will walk out understanding the infinitely deep genius of the Talmud, appreciating the intellectually mind-bending logic, and seeing the intellectual and practical relevance of such legal debates in the modern world.

Beyond Papert's important work, there are other educational models which also support the importance of pushing students towards deeper thinking, something which can be facilitated through the study of computer science. In 1956, Benjamin S. Bloom from the University of Chicago, along with a team of other researchers developed a taxonomy for education and cognitive skills. They subdivided cognitive tasks into six categories, ranging from lowest level of "knowledge" to the highest level of "evaluation". Each category was then assigned verbs which describe tasks which use this skill. In 2002, David Krathwohl and a team of researchers redesigned this taxonomy to account for metacognitive skills, and the ability for students to think about their own thinking process (similar to the aforementioned "style of thinking" Papert argues a student of computer science develops). The revised taxonomy builds from verbs which designate tasks which are factual, then advances to delineate tasks for expressing conceptual knowledge, then advances further to procedural knowledge, and finally to meta-cogntitive knowledge. Each level is seen as a progression, for "mastery" over a certain category of a cognitive process (i.e. factually understanding the details of a World War) is a necessary "prerequisite" for the next level of analysis (i.e.

perhaps understanding what motivated each side, or arguing about the legitimacy of each side in the war) (Krathwohl, 212-213).

Currently, traditional education has over-focused on the acquisition of factual knowledge, especially in the Judaic studies classroom where there are so many facts which are necessary for students to recall in order to become knowledgeable and practicing Jewish adults. However, Krathwohl argues that it is "objectives that involve the understanding and use of knowledge… [that] are usually considered the most important goals of education" (Krathwohl, 213). Through the prism of Bloom's Taxonomy, one can further understand how the integration of computer science thinking into a Jewish studies classroom could be further advantageous, in terms of using the parallel logic to give the students a framework for more rapidly doing the lower-level (and yet simultaneously *extremely* necessary) cognitive tasks such as translating a text, or organizing "who said what" in an argument. With this use of organized thinking from computer science and logical structures, students can more rapidly complete the necessary lower-level understanding tasks in order to move towards higher-order thinking skills such as understanding implications of a given text, and generating their own ideas based on solid factual understanding of the flow of the text.

Pushing students at younger ages towards deeper thinking about Jewish texts will embolden them as life-long learners, who can appreciate the unbounded depth and complexity in the nuance of Jewish texts. Students will no longer be walking away from a Jewish education so overwhelmed by and overtired from exposure to only factual knowledge that they do not perceive any analytical rigor in Judaic studies. By exploiting the parallel logical structures to Jewish text found in computer science to serve this purpose, educators

can deepen their student's acquisition of the factual prerequisite knowledge, and then move more deeply into meta-cognitive tasks, especially since through the integration of computer science, students have already begun to develop their own awareness of different "styles of thinking".

Precedents for Computer Science Integration in the Judaic Studies Classroom

Papert outlines a tremendous vision for ideal constructivist computer education, with many applications for the parallel logic embedded in Judaic studies. However, is the current integration of technology into the Judaic studies classroom fulfilling Papert's vision of a truly constructivist classroom? The notion of integrating technology into the Judaic Studies classroom is not a new or novel idea, and has been around for almost as long as technology itself. Starting from audio recordings of *shiurim* prior to the internet, until the emerging mobile applications used for accessing content today, the Jewish community has consistently been attempting to better and better use technology to further accessibility to Torah. Many current Jewish technology projects are focusing on using technology to increase the prevalence of, accessibility to, and "fun"-factor of Jewish educational experiences. These are all extremely worthwhile projects that benefit the community tremendously, however it is very different from the model which is being proposed in this thesis. Many of these programs use fun videos, interactive animations, and the accessibility of the internet to make Torah content not only widely available, but "flashier" and more exciting. In this thesis, however, my argument is towards the integration of the "back-end" thinking skills used in the development of these technologically-generated special effects. If one harnesses the parallels

between the process of technological development and the process of learning a Jewish text, one can aid the student of Jewish students to more deeply understand Jewish texts, and move more rapidly from lower-level thinking skills towards higher-order analysis.

There has been limited work done on a more "back-end" focused integration of computer science into the Jewish studies classroom. There is one study worth analyzing both for its advances in merging the two, and to evaluate in contrast to my proposal in terms of areas for possibly deeper integration models. In 2013, Marina Umaschi Bers, a researcher on educational technology from Tufts University published the results of a study called "Mi Ani" focusing on the use of robotics in early childhood Jewish education. This study taught kindergarten students the basics of robotics, and had them program robots to "walk" along a timeline of their year in kindergarten, with the ultimate goal of using the robots as a forum for expressing their Jewish identity. According to Bers, the goal of the Mi Ani project was to facilitate an "educational experience that promotes exploration of issues of identity by inviting them to create robotic creatures as alter-egos and program their behaviors in response to both Jewish… and secular… events" (Bers, Matas, Libman, 164).
The students underwent a multi-month long process of building a timeline, planning a program, testing the program on their robots, and concluding with a showcase that invited the parents to watch the students' robots run alongside their timeline. Students programmed robots to pause along the timeline posted on the wall at crucial moments, spin to symbolize moments of happiness, or blink little lights to symbolize other emotions or events the students may have experienced.[12] According to the research team, the benefits of integrating

[12] A more detailed description of the project: "'As a first step children reflected on their experiences during the year, guided by their teachers during open circle times coming up with a timeline consisting of different events during the academic year that were meaningful to them... Each child chose three moments in the year as

programming robots into this reflective project, "children are invited to explore their relationship with Judaism through dynamic behaviors rather than static symbols or objects" (Bers, Matas, Libman, 165). While this study definitely integrated computer science in an age-appropriate way into a Jewish studies classroom, the technology was still being used as a medium for expressing content in a more exciting or "flashier" way, as opposed to fundamentally using the logical principles of computer science to deepen the structure and rigor of text study.

Admittedly, the "overarching goal of the Mi Ani project" was not to use computer science principles to deepen text study, the researchers explicitly state that the goal was to design a program that by "leveraging on 21st century skills (such as technological fluency), can promote ways for minorities and diasporas to sustain their communities over time." (Bers, Matas, Libman, 165). While this may be the most rigorous study performed to date on an integrated computer science and Jewish studies classroom, the explicit goal of the study does not match the objective of the proposal in this thesis, to use computer science principles and the embedded parallel logical structures in Jewish studies to deepen the rigor of Jewish text. The Mi Ani project, of using robotics to express identity, could be applied to any content matter, and was not explicitly designed to deepen Jewish text-based understanding[13].

"stations" at which his or her robot would stop and perform an action. For example, one child programmed his robot to stop along the timeline at November, spinning to represent eating turkey on Thanksgiving, whereas another programmed her robot to sing at December to represent singing Hanukkah songs, and a different one to shake to express excitement because their classroom was studying butterflies. Children decorated the robots to represent themselves, using art materials to depict their interests, and their characteristics. For example, one child decorated her robot with drawings of all her favorite sports, whereas another molded a clay image of herself that she attached to the top of her robot. Each child programmed his or her robot to travel alongside the timeline stopping at three points in time to perform different actions, demonstrating children's own understanding of significant moments of their experience throughout the year' (Bers, Matas, Libman, 170-171)."

[13] Part of this discrepancy may lie in the fact that the study was designed for early childhood students which have a different focus in their education, as opposed to later education. In her article, Bers quotes the following

In the case of Mi Ani, they chose to apply the robotics lesson to a Jewish topic, however in general, the proposed educational program was not designed for a Jewish text explicitly, and as aforementioned, was designed to be applied to all different kinds of minority groups working to retain their identities.

Bers, in her article, concurs with Papert's claim mentioned earlier in this paper, of the necessity for Jewish educators to remain culturally relevant with their education, both in matching the styles of thinking emulated in the classroom to the styles of thinking being developed in the technological world, and in terms of "taking advantage of the full potential of high-tech" (Bers, Matas, Libman, 166). Quoting research done in 1984 by Sherry Turkle, the Mi Ani Project was "inspired by the potential of the computer to become a 'second self' or 'psychological machine', not because it has a psychology of its own, but because it provokes us to think about our own sense of self and identity" (Bers, Matas, Libman, 167). Using the computer as this kind of alter-ego, children are motivated to start thinking about "their own identity as a process, as opposed to a fact" (Bers, Matas, Libman, 167). Furthermore, this project aimed to teach students how to use technology to make a "better world through their use of computational skills and new ways of thinking" (Bers, Matas, Libman, 167). Despite these many parallels to the advantages of computer science education which contain many echoes from Papert's work, integration of computer science and Jewish studies can be taken one step further, building on the inherent logical parallels between the

---

excerpt from *The Jewish Preschool Teacher's Handbook*: "As opposed to later Jewish education, which is often focused on cultivating Judaic skills or content knowledge, early educators lead their students "to feel, to understand, to live, and to love Judaism" (Wolf & Nowak, 1991, p. vii)

With regards to early childhood students and programming, Bers writes: "Previous research has shown that children as young as four years old can understand the basic concepts of computer programming and can build and program simple robotics projects" (Bers, Matas, Libman, 172)

structure of the text, and the structure of building a computer program. Drawing on these parallel processes will allow students to meet these same aims as the MiAni Project, and even go one step further. Like the MiAni Project, students will learn how to make the world a better place through computational thinking, and will engage in a study of their own psychology by analyzing their own thinking. However, a deep integration model will push students one step further, moving them more deeply into the Jewish text itself.

Proposed Model based on Research done in conjunction with the Software Engineering Team and Education Team at Sefaria

For the past two summers, I have had the immense privilege of working as a Software Engineering intern at the Sefaria Project[14]. Sefaria is a non-profit open-source project working to digitize the entirety of the Jewish canon, making both the text and the code to support the interlinking of the texts available to the public for free. Under the mentorship of Senior Product Engineer, Russel Neiss, as well as the Chief Learning Officer, Sara Wolkenfeld, I developed a preliminary curriculum over August 2019 working to use computer science principles and structured thinking to deepen student understanding of Jewish text study. The project was called "Torah Hacking" and aimed to develop three model lessons which integrated computer science and Judaic studies in a way that builds on both the parallel logic in both fields, and attempts to more comprehensively fulfill Papert's constructivist educational dream for computer science education. These lessons were built in

[14] This project would not have been possible without my two incredible summers of learning and exploration as a software engineering intern at Sefaria. I especially have tremendous gratitude for my engineering mentor, Russel Neiss, who not only gave me immense technical training, but further fueled my passion for Jewish education. I also have infinite gratitude to Sara Wolkenfeld, Sefaria's Chief Learning Officer for her educational insight, and editing of the Torah Hacking project.

an interactive Jupyter Notebook system (which combines a text editor with a live IDE for running Python code). The goal was to use these Jupyter Notebooks to build lesson plans for an integrated computer science and Jewish studies class for middle schoolers with limited coding backgrounds, yet who are motivated to learn and explore more about the field. The process included drafting interactive coding-based lesson plans based on existing non-programming-based lessons in the Sefaria "Exemplary Pedagogy" group of source sheets developed by experienced educators, and adapting them to include basic computer science principles and Python programming to deepen student understanding of the text.

A joint effort between the Sefaria engineering and education teams, Torah Hacking was a mini-series of three lessons that experimented with using the logical patterns which emerge from computer science to prompt higher order analysis of Jewish texts (see the appendix for sample lessons). The hope is that students will gain basic exposure to aspects of computer science, and will be familiarized with "under-the-hood" Sefaria-specific technology and use both to deepen analysis of Jewish text, and build mini-tools to share with their broader communities.

Based in Jupyter Notebook, a platform which allows for integrated code and text, these three standalone lessons were a sampling of the potential that exists at the intersection of computer science and Jewish studies. Our goal was to empower budding Torah scholars and technologists to actively take hold of the power extant in programming, and use it to think more deeply about text while simultaneously building tools to enhance their respective Jewish communities.

Future possibilities for this project include a more advanced series for students with a basic programming background which would delve deeper into the parallel logical structures between certain passages of Mishna and basic algorithms. Alternatively a more hands-on course focused less on deepening Jewish textual analysis, but rather on building more robust applications based on the Sefaria API would also be a possibility.

This model project, which aims to build an actual manifestation of many of the claims explained above in this paper, led myself and the team of mentors to ponder critical questions such as identifying which texts may lend themselves more easily to logical parallelisms with computer science in such a way that the use of computer science aids the student in understanding the text. Furthermore, though our vision to deeply intertwine the two fields in order to deepen one's understanding of Jewish text was clear to us, finding ways to actually implement this in a cohesive lesson was incredibly challenging. It was all too easy to fall into the trap of superficially connecting the two fields in a way that just taught computer science within the context of Jewish content (likened to teaching addition by having students add quantities of *lulavim* and *esrogim*, which does not actually teach anything about the laws of Sukkot, rather the Jewish content simply adorns the mathematical content). This task, and process of writing lessons, redrafting, analyzing and critiquing left us with only three model lessons at the end of a month of full time work. Opportunities for further development are many, including most significantly the question as to whether or not it would be possible to build an entire curriculum around this theory of deep integration.

Conclusion

Ultimately, the study of computer science is the study of descriptive language, of developing logical structures through which to concretize abstract notions and relay instructions to a machine. As children gain increasing exposure to computers, not only do their modes of accessing information change, but the very way students learn to think and process information changes as well. It is critical that the Judaic studies classroom adapts to these changes in student cognition, and harnesses the parallel logical structures inherent within both the study of computer science and Jewish texts to provide students with formalized mechanisms for processing text more quickly, and for progressing into higher-order thinking about those texts at a younger age.

Furthermore, as thinking changes to meet the new problem solving needs of a computer generation, it becomes even more imperative for Jewish studies educators to draw on the inherent parallels between the two fields in order to speak of Jewish subjects in a way that the students will perceive as just as (if not immensely more) intellectually rigorous than their secular subjects. The study of computer science, as elaborated upon by Papert, has shifted the way students think, and more importantly the way students think about thinking. Bringing these elements of constructivist education and metacognition into the Jewish studies classroom would enable students to use their most advanced thinking skills to more deliberately and effectively unpack a text.

Current models for computer science and Jewish studies integration do not go deep enough, relying on the technology as a vehicle through which to deliver content in a more exciting way. My proposed model, as demonstrated in the three sample lessons researched,

built and designed at Sefaria, hopes to create a curricular model that allows for deep integration of the logical principles guiding both fields in order to deepen the actual content acquisition and study of Jewish texts. As 21st Century Jewish educators, if we begin to use the computer not just as a vehicle for delivering content, but rather as a means for pushing our students to think more deeply about how they think about Torah, we can raise the level of engagement and rigor in the classroom, and meet our ultimate goal of developing our students into passionate life-long learners, deeply aware of the infinite intellectual (and then hopefully by extension, spiritual) depth inherent in our texts.

Appendix

Below are links to non-interactive versions of the three lessons I developed with Sefaria over the course of my summer 2019 internship. In addition to the PDFs (which are again, not fully representative of the student experience using an interactive Jupyter Notebook), there are links to video demonstrations of the interactive versions. The full link to the entire project can be found below. On that document, you will find links to download the .ipynb files which can be run in a Python 2 Jupyter Notebook, and instructions for getting the system running on your computer.

The Sukkah Sensor

The Sukkah Sensor is a lesson which explores using Python objects to virtually simulate multi-faceted descriptions within a Mishnah. This lesson reinforces text analysis, and uses virtual models to test boundary cases.

- PDF of the lesson (non-interactive): https://drive.google.com/file/d/1zV1AXIVnjT5qQge0jPsSUQWj0HVC0yaf/view
- Video Demo: https://www.youtube.com/watch?v=tpbTaqWUou0&feature=youtu.be

Sefaria 401

Building on an existing lesson (https://www.sefaria.org/sheets/125796) by Rabbi Tzvi Sinensky which introduces students to using Sefaria through the website interface, this fourth track adds an introduction to accessing Sefaria texts from the API. Students learn what an API is, and learn Sefaria-specific syntax for creating text references. This lesson culminates

in a mini-project using the Sefaria-specific technology to build a TorahBOT, as well as in a reflection about the benefits and challenges of digitized learning.

- PDF of the lesson (non-interactive): https://drive.google.com/file/d/1Ikhu1O7Zx2NqREvUAYMbjxeskvEjhlsP/view
- Video Demo: https://www.youtube.com/watch?v=uZoMJgx8Xgc&feature=youtu.be

Between Him and Her: A Ketubot Supplemental Activity

Built on an existing lesson (https://www.sefaria.org/sheets/143730?lang=bi) by Sara Wolkenfeld, Sefaria's Chief Learning Officer, this activity adds a virtualization of the Mishnah and introduces the student to the process of debugging code. Through this process, the student will be forced to analyze the text in a new light, and think more deeply about the multiple factors involved in the Mishnah.

- PDF of the lesson (non-interactive): https://drive.google.com/file/d/12K7ocHrr--vXjXrBHiYGppZpZ1zgStXq/view?usp=sharing
- Video Demo: https://www.youtube.com/watch?v=XC5QiokWgjs&feature=youtu.be

Full Link

Below is the full link to the entirety of the work completed during the Summer 2019 with the Sefaria Project.

https://docs.google.com/document/d/1jJ1Xgqmu5EDYXkOoCe7TgXVae74V9xxLPX32PDEmMPU/edit?usp=sharing

Works Cited

David R. Krathwohl (2002) A Revision of Bloom's Taxonomy: An Overview, Theory Into Practice, 41:4, 212-218, DOI: 10.1207/s15430421tip4104_2 (retrieved from: https://www.tandfonline.com/doi/abs/10.1207/s15430421tip4104_2)

DevTech Research Group. "Mi Ani: An Exploration of Jewish Identity with Robotics." *Youtube*, based on the work of Marina Umaschi Bers, Jared Matas & Nehama Libman, 23 July 2010, https://www.youtube.com/watch?v=9O4Mz1iLDWw

"Data compression." *Britannica Academic*, Encyclopædia Britannica, 7 Feb. 2020. yulib002.mc.yu.edu:3043/levels/collegiate/article/data-compression/2216. Accessed 25 Apr. 2020.

Marina Umaschi Bers, Jared Matas & Nehama Libman (2013) *Livnot U'Lehibanot*, To Build and To Be Built: Making Robots in Kindergarten to Explore Jewish Identity, Diaspora, Indigenous, and Minority Education, 7:3, 164-179, DOI: 10.1080/15595692.2013.787062

Mishnah. Sefaria, https://www.sefaria.org/texts/Mishnah

Papert, Seymour. *Mindstorms: Children, Computers and Powerful Ideas.* New York: Basic Books Inc, 1980.

Saujani, Reshma. "Teach girls bravery, not perfection." TED. Feb. 2016. Lecture. https://www.ted.com/talks/reshma_saujani_teach_girls_bravery_not_perfection?language=en

Steinsaltz, Rabbi Adin Even-Yisrael. *The William Davidson Talmud*. *Sefaria,* https://www.sefaria.org/william-davidson-talmud

Wolf, S. F., & Nowak, N. C. (1991). The Jewish preschool teacher's handbook (Rev. ed.). Denver, CO: A.R.E. Publishing, Inc.