The Search for Meaning: Inferential Strategic Reading Comprehension in Programming

Maria Kallia
University of Glasgow
Glasgow, UK
maria.kallia@glasgow.ac.uk

ABSTRACT

Background and Context. Cognition, in all its forms, is inferential; inferential reasoning underlies processes like decision-making, problem-solving, argumentation and text comprehension. In the Psycholinguistics domain, it is argued that the study of text comprehension - the way that a coherent representation or a mental model of the text is constructed - is a study of inference generation. In programming education, code comprehension has been considered a milestone for students' progression. An important goal of code comprehension tasks is for the readers to build coherent representations or mental models of the code.

Objective-Hypothesis. A common activity during which code comprehension is required is studying a solved programming problem (SPP). There is a threefold comprehension process that takes place during SPP tasks: comprehension at the problem level, comprehension at the program level, and their relationship. These three processes guide the comprehension and the construction of a coherent mental representation of the whole example; drawing from Psycholinguistics, we hypothesize that differences in the way that students construct the mental model of a given SPP may be associated with differences in inference generation during these three processes.

Method. Through a mixed research design that brings together a theoretical-conceptual study and an empirical one, an inferential model for SPP is first developed based on scholarly research in Psycholinguistics and Programming Education. The inferential model is then tested and further refined through a comparative case study during which we identify and compare the kind of inferences a proficient, an average, and a struggling first-year undergraduate student made during the comprehension of an SPP task after the end of the first semester.

Findings. The results of this study demonstrate that the more advanced students employed specific inferential strategies during the three SPP comprehension processes: *Global Backward Explanations and Forward Predictions*, *Local Backward Explanations and Forward Predictions*, *Associations* and *Paraphrasing*, all of which guided their SPP comprehension and successful transfer to an isomorphic task.

Implications. The paper provides the first evidence towards a) the importance of inference generation during the study of SPP tasks; b) the kind of inferences made during successful SPP comprehension; c) the development of new research directions and theoretical insights and concepts in the area of SPP comprehension; and d) the development of teaching practice that elicits inferential strategic reading comprehension in order to explore and guide students' understandings before actual problem-solving.

CCS CONCEPTS

• **Social and professional topics** → **CS1**; • **Theory of computation** → **Program reasoning**.

KEYWORDS

inferences, program comprehension, reasoning, programming

ACM Reference Format:
Maria Kallia. 2023. The Search for Meaning: Inferential Strategic Reading Comprehension in Programming. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1 (ICER '23 V1), August 7–11, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3568813.3600135

1 INTRODUCTION

In all aspects of our lives, inference generation is critical; inference generation refers to the process of filling in information that is not explicitly given [23]. Moshman [37] posits that thinking is the conscious application and coordination of inferences to achieve one's goals and it is evident in inferential activities like problem-solving, decision-making, planning, argumentation etc. In reading comprehension (comprehending texts) as well, people make inferences to generate meaning. In a general sense, successful reading comprehension depends on an inferential process which allows the recognition of meaningful relations between different parts of the text, and between the reader's background knowledge and those parts [23].

The critical role of inferencing in text comprehension (e.g., narrative texts) has led researchers to assert that drawing inferences is not just a side-effect of comprehension but a plausible cause [6]. However, research demonstrates that while expert readers instantly produce inferences, novices do not have the strategies needed to interpret texts [13]. The difficulties faced by poor comprehenders are usually assigned to the following three reasons: a. difficulties in integrating information explicitly mentioned in the text (text-connecting inferences), b. difficulties at incorporating information outside the text with information in the text to fill in gaps (gap-filling inferences) and c. lack of knowledge on when and how to make inferences [6].

ICER '23 V1, August 7–11, 2023, Chicago, IL, USA

ACM ISBN 978-1-4503-9976-0/23/08…$15.00
https://doi.org/10.1145/3568813.3600135

In programming, code comprehension has been considered an important milestone for students' progression [19]. In this study program comprehension refers to the process by which students construct a mental model of a given program which is critical for understanding the underlying mechanisms under which the program achieves its goals. Program comprehension underlies many activities such as when studying solved programming problems (SPP - a problem text accompanied by a code solution), a common activity used in programming at all levels of expertise. However, many learners focus on shallow processing (superficial details) of the corresponding solution which prohibits the construction of procedural schemata and thus, transfer to real problem-solving situations [33]. To this end, Margulieux et al. [33] suggested subgoal labelling as a technique to facilitate learners to recognize procedural structures in worked examples.

At some point, though, in their learning journey, students need to develop strategies to engage independently with SPP examples, whether these are found in lecture materials, textbooks, online sources, or in peers' solutions. From our experience, some students develop these strategies quicker than others while the latter struggle to engage with these tasks. And while research in programming education has suggested explanations for this issue (e.g., [31][58]), the current research aims to dig deeper into the underlying reasons causing this problem.

There is a threefold reading comprehension process that takes place during the study of SPP examples: comprehension at the problem level, comprehension at the program level, and their relation. This threefold process guides the comprehension and the construction of a coherent mental representation of the example. Guided by literature in Psycholinguistics and Programming Education, we hypothesize that differences in the way that students construct this mental model of a given SPP may be associated with differences in inference generation during these three processes. In other words, we suggest that students' comprehension of an SPP example depends on students' inferential skills which make explicit the conditions and reasons under which code-related actions occur, and in relation to the problem; thus, students who are better at making inferences understand the meaning that the SPP example is trying to convey and are better able to build up the mental representation of the example during the comprehension process.

Through a mixed research design that brings together a theoretical-conceptual study and an empirical one, an initial inferential model for studying SPP tasks is first developed based on scholarly research in Psycholinguistics and Programming Education. The inferential model is then tested and further refined through an empirical study during which we identify and compare the kind of inferences a proficient, average and struggling first-year undergraduate student of similar academic background made during the comprehension of an SPP task.

This is the first study to our knowledge that attempts to understand inference generation in programming education and in the comprehension of SPP examples particularly. Identifying and examining the kind of inferences we generate could reveal further insights about students of varying comprehension skills and the difficulties they experience with integrating information in the problem domain and program domain, or whether they do not even attempt to generate inferences [7] which has significant implications to teaching.

Therefore, the paper aims to answer the following research questions:

(1) What is the role of inference generation for the comprehension of SPP examples?
(2) What kind of inferences are generated during this process?

The rest of the paper is structured as follows: the next section discusses literature aiming first to guide and inform the reader about inferential research in the Psycholinguistic domain and reading comprehension in Programming Education and demonstrates how these two domains combined lead to the foundation of this study's inferential model which is empirically then tested and presented in the second part of the paper. Section 3 presents the methodology and Section 4 presents the results of the empirical study. Section 5 is a discussion that brings together all the findings, refines the model, and makes suggestions for further research.

2 THEORETICAL BACKGROUND & INITIAL INFERENTIAL MODEL

2.1 Text Comprehension

The narrative text is a type of text that describes a sequence of events and actions in the pursuit of goals ([15][2]). These texts usually centre around a complicated problem to be solved and the actions, events and plans that someone follows to solve it [4]. Narrative text comprehension refers to the process of understanding and interpreting the meaning of narrative texts. A reader's main aim when reading a narrative text for comprehension is to understand and interpret a. the goals, the motives or reasons that explain a specific action, and b. the events, actions and states that cause or enabled events [14]. Thus, the process of comprehending narrative texts requires the readers to establish and maintain causal coherence between the various events and ideas described in the text [52]. It includes not only the detection of syntactical structures and the extraction of meaning of individual sentences but also the identification of the relationships between various parts of the text, as well as between the text and the reader's previous knowledge. These relations integrate the events, ideas and actions described in the text in a coherent structure represented in memory and are the result of inferential processes [53].

Text comprehension (e.g., narrative texts) has been the focus of study in various fields; since the seventies, researchers' attention on the mechanisms involved in text comprehension centred around inferences with some arguing that comprehension is actually an inferential activity [43]. The central role of inferences in reading comprehension has urged researchers to answer questions relevant to the definition of inferences and their nature in reading comprehension, the different kinds of inferences generated during comprehension, whether they are generated online (during comprehension) or off-line (after comprehension), and how they are being generated [40].

2.1.1 ***Inferences.*** In the context of text comprehension, inferences are considered necessary for connecting ideas and filling in information not explicitly stated in the text. Research evidence shows that inference ability is one of the exclusive predictors of

reading comprehension [54][24]. Within the Psycholinguistic domain, Rickheit et al. [43][p. 8] describe an inference as the *"generation of new semantic information from old semantic information in a given context"*. Hammadou [18][p. 28] defines inferences as a cognitive process, a reasoning process employed to construct meaning, generalisation and explanation [40], while Elbro & Buch-Iversen [11] define inferences as information that is generated during reading to fill in information that is not in a text.

Inferences depend upon information for establishing local and/or global coherence. The former "is exemplified" when readers link current read information with one immediately preceding. The latter is maintained by linking current read information with an earlier one, not the one immediately preceded, and with prior understandings [7]. Both these activities are critical for constructing a mental representation that encodes and links information explicitly mentioned in the text with background knowledge (ibid).

Another categorisation splits inferences into *Explanations*, *Predictions*, and *Associations*. *Explanations* refer to the reasons why something occurs, *Associations* refer to features and functions of something or someone, and *Predictions* refer to upcoming consequences of a central event [38]. *Explanations* are backwards-oriented while *Predictions* are forward-oriented but both integrate sentences or information across the text [51]. *Explanations* link the focal sentence with text information or prior knowledge and serve as reasons, explaining why something occurs. *Predictions* refer to causal consequences of a focal event and may or may not be substantiated but if they are, they help to build coherence. *Associations* elaborate the text and add information which may not be used again or may be used as an explanation or prediction and thus, help to bridge sentences or thoughts. *Associations* provide information "who does what to whom with what, when and where", and information on features, properties, relations and functions of persons, objects or concepts[51].

2.1.2 ***Processing and Levels of Representation.*** To fully understand the meaning of a text, readers must go beyond a surface understanding; fully understanding a text indicates building up a mental representation of what the text is about or building the situation model [26]. A central component in this process is inference processing, referring to the process by which readers integrate text information with background knowledge to fill in information not mentioned explicitly in the text [25][36]. Integrating information in the text with the readers' background knowledge is, thus, critical for the construction of a coherent representation. This representation is dynamic as each sentence revises, extends, strengthens or challenges the text representation constructed so far. There are different representation levels of text understanding. Van Dijk and Kintch [55] distinguish three: the surface code, the textbase and the situation model.

The surface code represents features of the text e.g., word meanings and syntactic knowledge and it has not been studied much since it does not contribute much to comprehension [12]. The textbase reflects the meaningful relations between components in a sentence or across sentences and requires minimal inferential processing [12]. These relations are directly prompted by the text and thus, the textbase is very closely related to the text itself; information is explicitly stated in the text.

The situation model relies heavily on inferential processing and reflects the referential meaning of the text which depends on the integration of information in the text with background knowledge. This integration results in strong learning outcomes and possibly facilitates transfer to new situations ([35] cited in [12]). Radvansky et al. [41][p. 156] argued that the "creation of a situation model is essentially an inference-making process in which the given information and general world knowledge is used to construct an understanding of the described situation". Basic syntax and vocabulary are usually enough to build the textbase model but for the situation model, high-level inferencing and comprehension monitoring are critical [42].

When a particular text is vague or implicit, readers need to enhance the situation model through inferential processing. Generally, inference generation and the situation model are mutually dependent meaning that inferences are fundamental for the construction of the situation model and the latter facilitates inference generation [3]. The kind of situation model that readers construct depends on their goals while reading along with their prior knowledge; that is why readers construct different mental models.

Overall, readers who process the text in the first two levels (surface and textbase) engage in shallow processing; that is because these two levels do not engage the learner with the deeper meanings of the text. As Davoudi [10] argues, deep meanings are achieved when learners are constructing causes that explain why certain events or actions occur, by inferring the global message of the text and by connecting the situations described in the text with background knowledge. The number of inferences produced during comprehension, therefore, can be regarded as one index for the richness of text comprehension.

2.2 Program Comprehension

Program comprehension [1] refers to the process of constructing a mental model of a program; this mental model incorporates the components, the structure, the execution behaviour and the purpose of the program (or parts of it). Research in programming education has highlighted the importance of program comprehension tasks which are regarded as scaffolds for writing programs (e.g., [29][31][30][32][58]).

One of the first models of program comprehension was suggested by Brooks [5] and centres around knowledge domains (problem and programming domains). Brooks proposed that programming is about constructing mappings from the problem domain to the programming domain and thus, program comprehension involves the reconstruction of these mappings: *"Using this idea of the knowledge domain, the task of understanding a program for a programmer becomes one of constructing, or reconstructing, enough information about the modelling domains that the original programmer used to bridge between the problem and executing program"* [p. 545]. This process is hypotheses-driven and stems from the programmers' knowledge of the task domain; these hypotheses are verified or nullified by searching the program for "beacons" suggesting a particular function. Although not rejecting a bottom-up strategy of program comprehension, Brooks highlights a top-down approach

[1] we only present literature that influenced our model due to space restrictions - readers interested in this area can refer to [19][47][44] for a literature review

during which hypotheses about knowledge domains and their relationship to the executing program are refined.

Brooks's theory suggests that hypothesis generation starts as soon as the programmer acquires information about the task and contributes to the reduction of the space of the program explanation and interpretation. These primary hypotheses, stemming from the programmer's previous knowledge, focus on the global structure of the program and includes information about the inputs, outputs, major data structures, and processing sequences. Only if the problem domain is totally unfamiliar will the generation of the primary hypotheses be delayed until the program text is revealed. The verification of the hypotheses takes place when the programmer tries to find evidence in the program text by producing subsidiary hypotheses. Some of these hypotheses will be indeed verified and the programmer will bind code to these, others will be rejected, and others will be produced when the programmer finds code that cannot be bound to any of the already generated hypotheses. Overall, Brooks regards program comprehension as a top-down and hypothesis-driven process which requires knowledge of the structure of the domains and the mappings between them.

Another model designed to help educators with program comprehension activities is the Block Model [47] suggested by Carsten Schulte (Figure 1). The Block Model is organized into a 4x3 matrix, with *columns* representing different dimensions of a program (Structure: text and program execution, and Function), and *rows* reflecting programming structures of increased complexity (atom, block, relations, macro-structure).

According to Schulte [47][p. 69], comprehension first starts by "sensing the program text which is a sequential process", during which readers engage with a word-by-word reading process, construct new information and add this to their mental model. The process then moves from word to word, to blocks and relations between blocks until the program's overall structure is recognized. Reading comprehension depends on the code already read, the reader's knowledge and the reader's goal, and endeavours to construct or hone an abstract and general mental model [47].

The *Text surface* of the model refers to the representation of the program text (e.g., syntax) and requires some lexical and syntax knowledge. This corresponds to the surface code of understanding a text presented in subsection 2.1.2 which represents features of the text like word meanings and syntactic knowledge. The *Program Execution* refers to the semantics of a program and relates to the text-based representation we presented above. The final dimension, *Function*, refers to the purpose of the program or parts of a program in relation to an extrinsic context; the purpose or goals of a program stems from an external source. This aligns with the situation model presented in subsection 2.1.2 (supported also in [19]).

The model also differentiates between two types of knowledge: program knowledge and domain knowledge. Program knowledge refers to extracting and comprehending the necessary information from the text surface and inferring the corresponding operational semantics. Domain knowledge is used to understand the context and thus, to understand the program's goals. In his paper, Schulte [47] notes that "understanding of function and goals of a program relies on inferences, and on a knowledge type referred to as domain knowledge" [p. 152]. He goes on to suggest that information is extracted from the program text and through inference generation, the reader comprehends the program execution and the program's goals and function. Both program execution and goals/functions need to be extracted from the text surface with the use of inferences which include real-world or domain knowledge.

Both Brooks's theory as well as Schulte's Block Model were the leading frameworks for our inferential model presented below.

Macro structure	(Understanding the) overall structure of the program text	Understanding the algorithm of the program	Understanding the goal/the purpose of the program (in its context)
Relations	References between blocks, e.g.,: method calls, object creation, accessing data	Sequence of method calls, object sequence diagrams	Understanding how subgoals are related to goals, how function is achieved by subfunctions
Blocks	Regions of Interests (ROI) that syntactically or semantically build a unit	Operation of a block, a method, or a ROI (as sequence of statements)	Function of a block, maybe seen as sub-goal
Atoms	Language elements	Operation of a statement	Function of a statement. Goal only understandable in context
	Text surface	Program execution (data flow and control flow)	Functions (as means or as purpose), goals of the program
Duality	Structure		Function

Figure 1: Block Model from [47]

2.3 Initial Inferential Model for SPP tasks

Based on key literature presented in the previous subsections, the initial inferential model is presented here and brings together reading comprehension theorisations in the two fields of Psycholinguistics and Programming Education.

As it is evident in Figure 2, at the centre of this model sits the situation model, referring to the mental representation constructed by the comprehension of the SPP example - similar to the Function side of the Block Model as also mentioned in [19]. Two levels comprise the model: the problem level and the program level and both contribute to the construction of the SPP situation model. Comprehending an SPP example is an iterative process requiring the reader to move between the two levels repetitively.

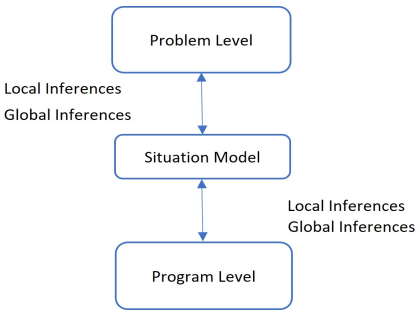

Figure 2: Initial Inferential Model

When readers are presented with an SPP example, the problem text is first presented to them. The reader, through inference-making at the problem level, starts constructing the mental representation of the problem. In this first version of the situation model, the reader infers goals, both superordinate goals referring to the main requirements of the problem as well as problem-based subordinate goals which implement superordinate goals or other subordinate goals. To do that, the reader draws inferences by extracting and connecting information presented in the text, the domain knowledge, as well as previous knowledge [47][5]. These inferences are produced locally, within the problem domain; thus, we name these as Local inferences in Figure 2 (after the empirical study, we exemplify these further). Since this is an SPP task, not all subordinate goals are expected to be "visible" to students or they might be to some advanced students; therefore, the situation model, in terms of goals, is not expected to be complete from the beginning. Information that will be generated at the program level is expected to be connected with the problem level informing the learner's understanding of how the program models specific parts of the problem; these inferences are called Global.

Having read and understood the problem description, the reader moves on to the program level and attempts to understand the code solution by using the situation model constructed thus far and updating it accordingly. As highlighted by Bos et al.[3], inference generation and the situation model are mutually dependent, meaning that inferences are fundamental for the construction of the situation model and the latter facilitates further inference generation. We expect two broad types of inferences to take place here as in the previous level: *Global* inferences and *Local* inferences. *Global* inferences act as bridges between students' program-level understandings and problem level; thus, they refer to the process of binding code to already generated hypotheses, as noted by Brooks [5]. *Local* inferences refer to inferences that bridge and connect information within the program level (e.g., some of the inferences hypothesized by the Block Model [47]), updating the situation model of the SPP example accordingly. This latter phase relates also to the code-binding process highlighted by Brooks, during which code cannot be bound to any of the produced hypotheses and new hypotheses are generated.

Overall, the initial model presented here binds together the key literature presented in subsections 2.1 and 2.2 and hypothesizes the broad kind of inferences expected to be found at each phase. The next section discusses the methodology with which the model was tested and further refined with explicit types of strategic inferences generated at each level.

3 RESEARCH METHODS

The paper follows a mixed methodological approach with which an initial inferential model is built through a conceptual study, and then an empirical study, based on a comparative case study design, serves as a validation and extension of the model.

The first part of this paper, presented above, aimed to develop the initial inferential model for SPP comprehension based on research available in scholarly literature in Psycholinguistics and Programming Education. Following McGregor's [34] guidelines for conceptual papers, the paper first presented key literature around inference generation in text comprehension in the subject area of Psycholinguistics; following this, subsection 2.2 presented key literature on program comprehension, finishing with section 2.3 which presented the inferential model by highlighting the connections with the literature presented in the previous subsections.

The second part of this paper aims to validate and refine the model empirically. The methodology employed for this part of the research follows a comparative case study design with three first-year undergraduate students of varying programming abilities: proficient, average and struggling.

Stake [50] describes case studies as "the study of the particularity and complexity of a single case, coming to understand its activity within important circumstances" (cited in [8][p. 375]), while Simons [48][p. 21] adds that it is about an in-depth investigation of an entity to capture its complexity and uniqueness. By employing a case study research design in this study, the aim is to provide unique examples of real contexts that enable the readers to understand the ideas presented earlier more deeply and how these fit in with the experiences and ideas brought up by the case. Because the objective of this study is to understand the potential role of inferences in SPP comprehension, a comparative case study design was employed.

Following a comparative case study necessitates specific methodological steps as defined by Kaarbo and Beasley [20] and are articulated below.

Step 1: Identify specific research questions for focused comparison. The specific research question addressed with the comparative case research design is "What is the role of inference generation for the comprehension of SPP examples and what kind of inferences are generated during this process?"

Step 2: Identify Variables From Existing Theory. The aim of this step is to identify the independent variable or explanatory variable that may explain the dependent variable. This was done by an extensive literature review, a summary of which was presented in subsections 2.1 and 2.2 demonstrating the critical role of inference generation, which constitutes this study's explanatory variable, while the students' comprehension levels of an SPP example (e.g., shallow or deep understanding) constitute the study's dependent variable.

Step 3: Case Selection. One of the constant debates in social science in the area of case studies is what constitutes a case [8]. Robson [45] argues that a case study can include just an individual and can extend to include social groups, organisations, institutions or even a nation. No matter how researchers choose to define a case, what is important is to clarify what constitutes a case for the particular research. Kaarbo and Beasley [20][p. 380] argue that there are three important tasks in selecting cases: "a. selecting comparable cases, b. selecting cases that differ on the dependent variable, and c. selecting cases across subgroups of the population". For the purposes of this research, a case is regarded as a single participant who represents a case of a subgroup of people demonstrating the same experience and skill in programming. In the participant's subsection, further details are provided on how the selection fits with the above criteria.

Step 4: Operationalize Variables and Construct a Case Codebook. Subsection 3.3 explains in detail how the codebook in relation to the explanatory variable (inference generation) was developed.

Step 5: Code-Write Cases. Narrative is the most employed method for presenting the results of a comparative case study. The aim of

the narrative in this sense is to allow the researcher to demonstrate the variables under investigation and how they manifest in the specific contexts. Since a whole detailed narrative is not always feasible due to several constraints (e.g., space limit), the researcher may focus only on short descriptions of the coded results [8]. As will be evident in the results section, the codes referring to inferential strategies and how these were manifested in the accounts of the students are first presented. A narrative explanation is also provided to add additional insights into the students' differences.

Step 6: Comparison and Implications for Theory. The aim of this step is to determine whether the explanatory variable differentiates across the results of the dependent variable. Both in the results section as well as the discussion, the differences between the cases in relation to both the independent and dependent variables are highlighted and how these inform the inferential model correspondingly.

3.1 Participants

Three cases were selected for the investigation of the phenomenon under study and are described below:

The three first-year computer science undergraduate students who took part in the study (18-19 years old) were all from the same university in the UK and the same cohort (academic year 2022). The students had no previous experience in programming when they joined the Python programming course taught for about 4 months (late September 2022 - December 2022). The course is specifically designed for those students who have never programmed before they join the university. Each week students attended lectures (4 hours in total) as well as a lab session where they worked with their tutors on several exercises (including code comprehension and problem-solving tasks).

The researcher purposefully invited students with whom she had worked closely during the semester. Specifically, the researcher acted both as a lecturer in a cohort of 150 students and as a tutor to 15 students. She sent invitations to 6 of her students whom she tutored personally and evaluated as belonging to one of the three categories (proficient, average and struggling), an evaluation that was also corroborated by the students' final grades in the course (the study took place after the students' final examination). Purposeful sampling was necessary as it was important for the aims of this study to select participants who demonstrated similar engagement patterns with the course, meaning that they attended the lectures and all the labs, had the same lecturer and tutor, as well as submitting all weekly exercises demonstrating an effort to learn.

Therefore, the three students had similar backgrounds in terms of their programming experiences before they joined the course, had similar experiences within the course, and all three demonstrated significant effort and engagement with the course. The proficient student represents a case of a group of first-year undergraduate students with no programming experience before joining the course, who significantly engaged with the course, and with a high final grade; the average student represents a case of a group of first-year undergraduate students with no programming experience before joining the course, who significantly engaged with the course, and an average final grade, and finally, the struggling student represents a case of a group of first-year undergraduate students with no programming experience before joining the course, who significantly engaged with the course but with a low final grade.

Ethical approval was granted by the researcher's university. Prior to the study, all participants were informed about the aims of the study and signed a consent form. The study was conducted in January 2023.

3.2 Empirical Study - Process

A very challenging question in inference research concerns the way an inference can be detected. In the literature, there are different methods that have been employed to this end depending on the time of inference detection. One method that has been extensively used is protocol analysis during which the participants are asked to think out loud as they read the text. Readers read a sentence at a time and then articulate aloud their understanding [51].

Similar to research methodology in Psycholinguistics, students in this study were instructed to read the problem text and then the solution (Figures 3 and 4). There were two parts contained in the problem: a familiar part, which required students to engage with patterns they had used before in the class (e.g., calculating the average of numbers in a list), and an unknown, unfamiliar part (highlighted with bold and underlined in Figure 3), which the students had not addressed before during the class. The students were not informed about this in order to avoid turning their attention on identifying the unfamiliar part of the problem as this was in fact part of the investigation. The researcher was particularly interested in examining how the students' inferential strategies and task engagement may differ in relation to previous knowledge or the absence of it.

A file "students.txt" contains information about students' grades in 6 assignments. Each Line includes the student's name and surname followed by the symbol ":" and then the 6 grades separated by space. Write a program that a. prints the name of students in descending order based on their average grades and b. prints all the names of the students with average score 20. **The search should finish as soon as possible.** If no student was found, it prints the message "student was not found".

Figure 3: Problem Task

```
myfile = open (" students.txt")
line = myfile.readline()
namelist = []
markslist = []
while line !="":
        list = line[:-1].split(":")
        listmarks = list[1].split()
        sum = 0
        for mark in listmarks:
            sum+=int(mark)
        average = sum/6
        namelist.append(list[0])
        markslist.append(average)
        line = myfile.readline()
sort (namelist, markslist)
print (namelist, markslist)
found = 0
i = 0
while i < len(namelist):
        if markslist[i] == 20:
            print (namelist[i])
            found = 1
        elif markslist[i]<20:
            break
        i = i + 1
if found == 0:
    print ("student was not found")
```

Figure 4: Code Solution

The students were instructed to read the problem text sentence by sentence and to tell the researcher about their thoughts. It is important to highlight that the students were not told to explain their

understandings to the researcher as explaining is in fact an inferential strategy and the researcher did not want to dictate a strategy to them. They were only told to communicate their thoughts to the researcher having as an overall aim of engagement to solve a second, isomorphic task. The students were first presented with the first sentence of the problem and communicated any thoughts to the researcher. When they were ready to proceed - meaning that they were not contributing any further thoughts - they moved on to the next sentence which was hidden until that moment. The previous sentence was also present to avoid cognitive overload. The students were again asked to tell the researcher about their understanding and thoughts. The process was repeated until all the sentences of the problem text were visible. After that, the students were presented with the code solution, one code statement at a time (with the previous statements visible), and asked again to communicate their thoughts to the researcher. If no thoughts were produced, the students were free to proceed to reveal the next statements until they could communicate any thoughts to the researcher; thus, they could proceed with revealing the whole program should they wish to.

As a final step, the students were asked to demonstrate their understanding in an isomorphic task exercise. In this exercise, the students had to fill in the blanks in the code solution. The reason for engaging students with this task was to give them a purpose/goal for engaging with the previous SPP task (as presented in the literature, the situation model or mental representations depends on the goal of reading) and to use their performance as a measure of the dependent variable corroborating also our personal evaluation of students' comprehension during the SPP task. The whole process was audio-recorded and transcribed later.

3.3 Data Analysis

As in Trabasso and Magliano [51], the analysis of the protocols began by first parsing the utterances the participants made while reading a focal sentence into clauses - these were single ideas demonstrated in a sentence. Therefore, an uttered sentence may contain more than one clauses. Each clause then was examined for its relationship to the focal sentence which produced that thought, for determining the type of inference. To this end, the researcher examined whether the clause aimed to paraphrase or metacomment or to explain (explanations), associate (associations), and predict (predictions) as in [51].

Therefore, following the guidelines described in [51] regarding the inferences, the researcher adopted a mixed way of coding starting with a deductive way focusing on the four main types of strategic inferences mentioned above. After that, the researcher applied inductive coding to further clarify the inferential categories and identify subcategories which are presented in the results section.

For the deductive coding, the following inference definitions were employed (these will be refined in the results section):

- *Paraphrasing* refers to a reading strategy repeating text input in a simpler way.
- *Explanations* are inferences referring to answers to why questions. Thus, to determine if a clause is an explanation, one can think of whether it is an acceptable answer to a why question in relation to an event or action in the problem or program level of the SPP example.
- *Predictions* infer actions or events that follow from a focal statement - they function as expectations of events and may or may not be substantiated.
- *Associations* are inferences referring to specifications of actions, and they provide answers to how something is implemented (e.g., using a list as a collection or a running total variable). They provide further details on how goals are implemented.

For identifying goals, the researcher coded as *superordinate* goals, goals that are the main problem requirements. *Subordinate* goals implement *superordinate* goals and can be problem-level based or program-level based depending on the level of reference, following suggestions by Schulte [47][p. 157] who highlights: *"It might therefore be very useful to develop two different versions of Goal-descriptions of blocks: One version is focusing on the local understanding, whereas the other version focuses on the global understanding of a block."*

The author, with the help of another researcher, coded together the proficient student protocols and designed a codebook with all the categories and subcategories providing examples and descriptions for all categories. They then separately coded the protocols of the other students and interrater reliability between the two researchers was calculated and was high (k=.83). The two researchers discussed the areas they disagreed with and informed the codebook accordingly.

3.4 Quality Criteria

The study follows the quality criteria in qualitative research suggested by [16]. These refer to "trustworthiness," which contains four aspects: credibility, transferability, dependability, and confirmability. In this study, credibility is ensured with investigator triangulation (discussed above), transferability with thick descriptions of behaviours and contexts, and dependability and confirmability with audit trail, reporting the research steps throughout the study [28].

4 RESULTS

This section is split into two sub-sections referring to the two levels hypothesised in the inferential model: the problem and the program level. For each of the levels, the inferences found as well as the differences between the three students are reported.

4.1 Problem Level

All three students started reading the problem text, sentence by sentence. The proficient student generated a total of 18 instances of inferences: 6 were *Paraphrases*, 7 were *Predictions*, 1 was an *Association*, and 4 were *Explanations*. The average student generated 9 inferences of which 4 were *Paraphrases*, 3 *Predictions*, 1 *Association* and 1 *Explanation* while the struggling student generated 3 inferences of which 1 was a *Paraphrase*, 1 was a *Prediction*, and 1 was an *Explanation*. Interestingly, the struggling student, in comparison with the other two, seemed as if reading the problem sentences without an attempt to understand the underlying meaning of the problem text.

Below, the kind of inferences identified are explained.

Table 1: Inferential Strategies at the problem level

Inferential Strategy	Students Proficient	Students Average	Students Struggling
Predictions	7	3	1
Explanations	4	1	1
Associations	1	1	0
Paraphrase	6	4	1
Total	18	9	3

4.1.1 Explanations. *Explanations*, as backward inferences, were used on two occasions:

- Text-related explanations: to justify the reasons why certain information was given in the problem text
- Goal-related explanations: to justify the reason why a certain action (e.g., a problem-based subordinate goal) needs to be taken as means to satisfy a superordinate goal

In the first case, *Text-related Explanations* were activated from the students' background knowledge. For instance, on reading the sentence "Each line includes the student's name and surname followed by the symbol : and then the 6 grades separated by space", the proficient student said: *"ok, we need that to understand how the information is structured in the file and access it accordingly"* giving in this way the reason why this information was important for them to know.

In the second case, *Explanations* as *Goal-related* were again activated from the students' background knowledge. For instance, on reading the sentence "prints the name of students in descending order based on their average grade", the proficient student explained: *" <subordinate goal> in order to print the names in descending order or else we will not be able to sort them if we don't keep the data somewhere"* highlighting both the superordinate goal and the subordinate goal needed to implement the former.

In both cases, *Explanations* served as bridges between the text and students' background knowledge.

4.1.2 Predictions. *Predictions*, as forward inferences stemming from previous knowledge, were categorised as follows:

- Expectations: these are predictions about upcoming actions (e.g., subordinate goals)
- Critical-Anticipation: predictions of the importance of a focal sentence

The first one - *Predictions* as *expectations* - were manifested in students' verbal protocols as causal consequences of a focal sentence. Usually, these consequences identified problem-based subordinate goals – goals necessary to achieve superordinate goals or other problem-based subordinate goals. For instance, when the proficient student read the sentence "print the names of the students in descending order based on their average grade" they said: *"we will probably have to calculate first their average scores and put them in a list and then sort it somehow"* identifying the subordinate goals "calculate the average" and then "sort" that are needed to implement the superordinate goal.

In the second case, *Predictions* as *critical-anticipations*, occurred once in the proficient student protocol and once in the average student. These were manifested when a superordinate goal was identified as important for the solution but the students did not proceed to further explanations which suggested that they were uncertain of how this goal could be implemented. For the proficient student, this superordinate goal referred to the second part of the problem that the student had not seen before. Specifically, when the student saw the phrase "the search should finish as soon as possible", they said: *"hm... that is interesting. The search should finish as soon as possible, so there must be a way to do that somehow during the search..."*. Although the student did not predict a way to achieve this goal, they did identify its significance and created an anticipation for how this would be approached at the program level.

4.1.3 Associations & Paraprases. *Associations*, as knowledge-based inferences that elaborate procedures by explaining how something is to be achieved, were evident only once for the proficient and average students. Both students explicitly mentioned the use of a list or a dictionary for putting in the information from the file, elaborating in that way how information from the file is to be processed. *Paraphrases* were also used mostly by the proficient and the average student and much less by the struggling student. These were actually the first strategy used by the advanced students after reading each sentence and helped them emphasise and turn their focus on the superordinate goals of the problem.

4.1.4 Students' Strategic Inferential Differences at the Problem Level. Most of the inferential strategies identified above were evident in the proficient and average students' verbal protocols. Both these students used more frequent *Predictions* and *Paraphrases* and fewer *Explanations* and *Associations*. *Paraphrases* helped the students focus on the main ideas in the text emphasising superordinate goals, and *Predictions* assisted the students with identifying problem-based subordinate goals implementing superordinate goals.

Similarities between the proficient and the average student's inferential strategies occurred on the part of the problem that was already familiar to them. The main difference was evident in the way these two students handled the unfamiliar part of the problem. The proficient student identified the unfamiliar part and generated a *critical-anticipation Prediction* demonstrating acute comprehension awareness. The average student failed to generate this inference which led to further issues at the program-level comprehension explained in the next subsection.

By using these inferential strategies, the proficient, and to a good extent the average student, created a rich initial situation model or mental representation of the SPP example at the problem level before they move on to study the code. As for the struggling student, they generated the least amount of inferences at the problem level and appeared to focus on just reading the text without attempting to generate meaning, and thus, they entered the program level with a very poor mental representation of the SPP example. In Brooks's terms [5], the student did not generate hypotheses to guide their understanding later at the program level.

4.2 Program Level

All three students started reading the program line by line and explained their thoughts to the researcher. At this level, in comparison with what was evident at the problem level, the vast majority

of inferences generated by the students were *Explanations* (backward inferences) and *Associations*, followed by *Predictions* (forward inferences). The proficient student produced 19 *Explanations* from which 10 were *Global* and 9 *Local* (explained below), 10 *Predictions* from which 4 were *Global* and 6 *Local* (explained below), and 34 *Associations*; the average student produced 17 *Explanations* from which 9 were *Global* and 8 were *Local*, 5 *Predictions* with 3 being *Global* and 2 *Local*, 49 *Associations*; and the struggling student produced the least amount of inferences: 8 *Explanations* with 4 being *Global* and 4 *Local*, 2 *Predictions* with 1 being *Local* and 1 *Global*, and 41 *Associations*. In total, the proficient student generated 40 Function understandings according to the Block Model, the average student 27 and the struggling student 10.

Table 2: Inferential Strategies at the Program Level

	Students		
Inferential Strategy	Proficient	Average	Struggling
Predictions	10(4G/6L)	5(3G/2L)	2(1/1)
Explanations	19 (10G/9L)	17 (9G/8L)	8 (4G/4L)
Associations	34	49	41
Function Understandings	40	27	10

4.2.1 Explanations. Explanations are inferences that give reasons for why an event/action has taken place. These are related to goals. Explanations were categorised into two main sub-categories:

- Global Backwards Inferences (GBI): act as bridges between the program level and the problem level. These are inferences that connect information at the program level with superordinate and problem-based subordinate goals.
- Local Backwards Inferences (LBI): act as bridges within the program level. These inferences connect information at the program level to a program-based subordinate goal.

A *Local Backward Explanation* inference explains the reasons why an action (e.g., a code statement or a block of code statements) is implemented in relation to a program-based subordinate goal already known (it was generated previously, e.g., through predictions mentioned below). For instance, having generated a program-based subordinate goal of the block of statements calculating the sum of a student's grades in a list, the average student explained the role of the statement "sum = 0", by saying: *"we want to add the grades from that list so we need to initialise a running total to zero first"*. In this case, the student explained the role of this statement in implementing the program-level subordinate goal that calculates the sum of the grades. *Global Backward Explanations* refer to reasons why the function of a code statement or block of statements at the program level is needed to satisfy problem-based subordinate goals or superordinate goals at the problem level. As an example, when the proficient student saw the code statements adding the names and average grades of students to two lists, he said: *"we do that because we want to sort the grades in descending order"* referring to the corresponding problem-based subordinate goal.

4.2.2 Predictions. Predictions, as forward inferences, occurred less frequently than explanations at this level and were categorised into the following categories:

- Global Forward Inferences (GFI): predict a future action that follows a focal statement or block of statements and is related to a superordinate or problem-based subordinate goal.
- Local Forward Inferences (LFI): predict a future action that follows a focal statement or block of statements and it is about a program-based subordinate goal; or it is just about a causal descendant (e.g., next code statement) of the focal statement or block.

For instance, for a *Global Forward Prediction*, when the proficient student saw the statement "sum = 0", he said, *"ok, it is going for the average of students' marks – let's see if that is the case"*. The student predicted that what follows from that code statement was the implementation of the problem-based subordinate goal that calculates the average. For a *Local Forward Prediction*, when the student encountered the sentence "sum += mark", they said, *"now the next statement will divide the sum by 6"* which is a causal descendant of the previous statement in order to calculate the average score.

4.2.3 Associations. We group into the theme of *Association*, inferences explaining what a code statement or a block of code statements does at the *Program Execution* or *Function* side of the Block Model. These inferences describe what and how a piece of code does something (e.g., "this statement adds the name in the list"). These inferences were not categorised as *Explanation* inferences as they do not refer to reasons for implementing a piece of code in relation to a known goal. Trabasso and Magliano [51] note that *Associations* may be used as an *Explanation* or *Prediction* later on and thus, help to bridge sentences or thoughts. For instance, having produced a *Function* understanding of a block through *Associations*, this understanding can then be connected with other information at the program level through a *Local Backward Explanation*. The main difference between the three students in relation to these inferences is that the proficient student provided 34 such inferences focusing mostly on the *Function* side of the Block Model the average student produced 49; most of them were related to the *Program Execution*. The struggling student focused almost explicitly on the *Program Execution*. For instance, when the student described the for-loop they said: *"the for loop here will iterate through the listmarks"*.

4.2.4 Students' Strategic Inferential Differences at the Program Level. As Table 2 depicts, the proficient and average students generated three times and two times more *Predictions* and *Explanations* respectively than the struggling student. Both these students engaged in *Global* and *Local Backwards Explanations* and *Predictions* although the latter were most evident in the proficient student. Both students generated *Global Forward Predictions* as expectations of upcoming problem-based superordinate or subordinate goals and *Local Forward Predictions* as expectations of upcoming program-based goals or actions. Similarly, both students generated *Global Backward Explanations* which referred to superordinate or problem-based subordinate goals while the *Local Backward Explanations* referred to program-based subordinate goals. Overall, by engaging with these two broad inferential strategies, both students monitor well

their comprehension by bringing together their understandings produced at the problem level with that at the program level and vice versa as well as within the program level. However, the main difference between these two students refers to the strategies involved in understanding the unfamiliar part of the problem.

When the proficient student encountered the block of statements for handling the unfamiliar part of the problem, they first generated a program-based subordinate goal and then bridged that to the problem by the use of a *Global Backwards* inference; the student said: *"...it exits the loop when the grade is less than 20, but why is it doing that? Ah, the search should end as soon as possible..."* and then they continued by trying to understand exactly how and why the particular set of statements implement this superordinate goal by producing *Local Backward Explanation* inferences and *Associations* that together explained that the list was sorted so once a grade less than 20 is found all the subsequent grades are going to be less than 20 and there is no need to continue the search. So, a local block functionality was followed by a *Global Backward* inference that connected the program and the problem level and that connection assisted the student in understanding the underlying mechanism of this connection by generating further local inferences (at the program level, whether *Explanations* or *Associations*). In the isomorphic problem task, the student was able to fill in the blanks correctly demonstrating a transfer of understanding from one problem to the other.

Interestingly, the average student identified the local function of the block but they did not attempt to connect it with a problem-based subordinate goal or to a superordinate goal through a *Global Backward Explanation*, failing to understand the underlying mechanism of that block. Specifically, when they reached the unfamiliar block, the student explained the local functionality and move on to the next set of statements. To the researcher, it was clear that a comprehension failure occurred as the student's behaviour and strategies were not the same as the ones employed in the familiar part of the program. Until that point, the student kept their program-level understanding in accordance with the problem level by providing explanations (giving reasons) of actions and events taking place. So, why did the student fail to see that they were missing something in that particular case? Going back to the student's inferential strategies at the problem level, it was evident that when reading the problem sentence related to that particular block ("the search should finish as soon as possible"), the student, in comparison with the proficient, did not generate a predictive inference (critical-anticipation in the case of a proficient student) nor tried to paraphrase the sentence. It seems like the student-produced inferences based mostly on their background knowledge but when they could not access prior knowledge they did not use other strategies to help them in this process. This interpretation aligns with Vaugh and Bos [56] who argued that some students rely too much on their background knowledge and miss insights from the text. Subsequently, when the student moved on to the isomorphic problem, they filled in all the blanks correctly apart from the blank that was related to this particular unfamiliar piece of code. The student tried to remember what the SPP example had in this gap without logically thinking about it. When the student was informed that they answered incorrectly this part, they went back to the SPP example and said: *"yes, I am not sure I understand why is this piece here"*. Then the researcher reminded the student to read the exercise again and try to understand which part of the problem this block tries to solve. When the student did that, they noticed: *"oh that part here ... the search should finish as soon as possible, that is what that part of code is about, right?"* They then turned to the program and started investigating how the program actually achieves that, mirroring the steps of the proficient student.

As for the struggling student, they produced the least amount of *Explanation* and *Prediction* inferences in relation to the other two students which caused many problems in the understanding of the whole example. As was mentioned in the above subsection, the student focused mostly on generating *Associations* and these mostly at the *Program Execution* level of the Block Model. When the student finished trying to understand the SPP example, they moved to the isomorphic task where they were trying to fill in the gaps by recalling the SPP example but failed to do so. It is clear that the student could not abstract away from the specifics of the text which prohibits them to generate *Function* understandings - program-based subordinate goals, let alone connecting these to the problem level (*Global Explanation* or *Prediction* inferences).

5 DISCUSSION

5.1 What is the role of inference generation during the study of SPP and what kind of inferences are generated?

The first research question of our study aimed to shed light on the role of inference generation during the study of SPP examples and to identify the kind of inferences that are generated during this process. The analysis of students' protocols, presented in the results section, demonstrates that the two more advanced students generated a greater number of inferences both at the program level and the problem level than the struggling student which was then translated correspondingly to success or failure in the isomorphic task. This observation was also evident between the proficient and average student regarding the unfamiliar part, during which the average student failed to generate the inferences the proficient student did; this again was translated to errors in the corresponding part of the isomorphic task. Therefore, variations in our explanatory variable (inference generation) seem to explain variations in comprehension of the SPP example, as was evident both during the students' protocols as well as in their performance in the isomorphic task.

The inferences identified are depicted in the following figure which demonstrates the inferential model presented in subsection 2.3 further refined with the specific inferences identified by this study.

The findings of this study highlight that four kinds of inferential strategies were employed during the comprehension of the SPP example: at the problem level, *Paraphrasing* and *Predictions* were the most frequently used strategies, followed by *Explanations* and *Associations* as defined for the purposes of this study. At the program level, *Explanations* and *Associations* were most frequently used, followed by *Predictions*. Therefore, it seems that comprehension at the problem level is mostly prediction-based - a process during which possible solutions and steps of the problem are produced based on initial information extracted from the problem text, the domain and prior knowledge – whereas, at the program

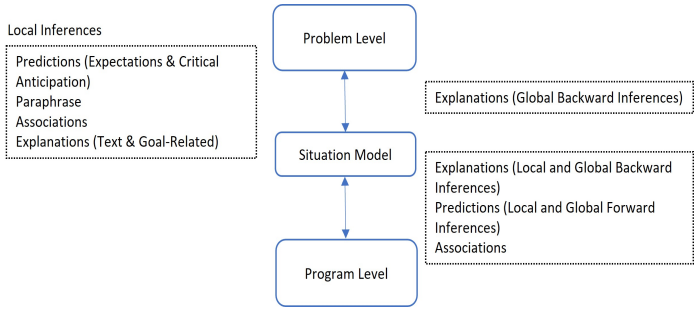

Figure 5: Inferential Model

level, comprehension is mostly explanatory-driven during which reasons for performing actions (code statements) are given. Both these findings align well with Brooks's [5] hypotheses generation, which on one hand, reduces the space of the solution exploration, and on the other, it is used as a binding mechanism between code and generated hypotheses. The findings of this study align also well with theories of schemas and plan acquisition experts use to guide their problem-solving process and mirror Soloway's point of view [49][p. 851]: *"learning to program amounts to learning how to construct mechanisms and how to construct explanations."*

Differences between the three students were evident at both levels which influence correspondingly the construction of a coherent mental representation of the SPP example. At the problem level, the struggling student barely expressed any thoughts after reading a sentence and kept revealing the next sentences of the problem jumping straight forward to the program level. This strategy seemed to have obstructed the student from bringing in prior knowledge and connecting it to the corresponding problem, something that is highlighted as a main problem poor readers experience [59][6]. On the contrary, the proficient and the average student generated thoughts in almost every sentence of the problem; *Paraphrasing* facilitated students to focus on main ideas and identify superordinate goals; *Predictions* prompted students to generate problem-based subordinate goals and connect these to superordinate goals through the use of *Explanations*, while *Associations*, although used rarely, offered more concrete details on how subordinate goals are to be implemented. Although the proficient and the average student employed similar strategies for the familiar part of the problem, they differ on the unfamiliar part with the average student failing to generate a predictive inference relevant to the unknown superordinate goal. It seems that the average student relied too much on their previous knowledge, something that was evident at the program level too. Vaughn and Bos [56] highlighted this as a potential problem struggling readers face by blocking inputs from the text and relying too much on their prior knowledge.

At the program level, the differences in inferential strategies between the three students continued to be evident. When reading the code statements the struggling student focused more on the inferences categorized as *"Associations"*, that is, inferences explaining what a code statement or block of code statements does at the *Program Execution* and *Function* side of the Block Model but they do not explain reasons for performing these actions in relation to a goal. Furthermore, the struggling student focused more on the *Program Execution* level of the Block Model with few occasions of attempting a Functional understanding. This finding aligns with program comprehension literature which suggests that novices do not "see the forest for the trees" [31], emphasising novices' difficulties to reason at the relational level of the SOLO taxonomy [58].

The two more advanced students, in comparison with the struggling student, employed at least two to three times as many inferential strategies as the struggling student related to *Explanations* and *Predictions*. *Explanations* are backward inferences and explain - give reasons for - why actions are happening. Thus, they facilitated students to monitor and guide their comprehension both at a local level, but also to connect information generated at the program level with the problem level by means of *Global Backward Explanations* (e.g., [17][22][21]). Further to this, *Predictions* which are forward inferences, facilitated students to recognize "beacons" signalling the use of a specific plan which is about to be implemented. This finding corroborates research suggesting that more advanced comprehenders are able in recognizing "beacons" and plans that facilitate the comprehension process (e.g., [5]).

Overall, the inferences produced by the proficient and average students led to the creation of a richer and more cohesive situation model than the struggling student which was translated to success and failure in the isomorphic task.

5.2 What are the implications for research and practice?

While understanding and comprehension are commonly used terms, it is often difficult to provide a precise definition as it strongly depends on the context it is applied in. Kintsch [27][p. 178] uses the phrase *"comprehension as a paradigm for cognition"* to refer to reading comprehension as a cognitive process that involves both the processes of perception (e.g., making sense of sensory inputs) as well as thinking or problem-solving that includes processes such as planning, search spaces and means-end strategies. According to Broek[52], understanding a narrative text is not a matter of understanding isolated actions but rather constructing a series of connected events that make a coherent story. This coherence is the outcome of a tedious inferential process during which the learner infers relations among the events and actions that are happening in the text. Therefore, Schmalhofer et al. [46][p. 106] advocate that understanding comprehension is understanding the *"dynamics by which inferences are drawn during reading, their maintenance over time, and their interplay with the representation being constructed."*. The role of this inferential process in reading comprehension is highlighted by Graesser et al., [15], who argued that comprehension is achieved when harmony takes place between three components of a communication system: the author's intended meaning, the text itself, and the reader's constructed meaning trying to uncover the author's goals.

This research study focused on understanding comprehension in the context of programming education and, more specifically, during the study of SPP examples. The findings align with the observations above and offer deeper insights into a phenomenon that has been one of the central focuses in programming education. The current research highlights that indeed SPP comprehension is a

cognitive process including processes of perceptions and problem-solving [27], and indeed, is a matter of causally connecting information to make a coherent whole [52], a result that is foregrounded by the dynamics of inference generation ability. More precisely, the results support the initial hypothesis posed in the introduction of this paper and further extend it by providing the first evidence showing that the SPP three-fold process of comprehension is guided by students' inferential strategies.

Difficulties in the students' inference generation were linked to failure to activate and apply relevant-existing background knowledge (e.g., [6][39]), difficulties in integrating information both within and between the two levels, as well as problems with relying too much on background knowledge or focusing too much on the text (e.g., [56]). Therefore, future teaching strategies would benefit from inferential strategic comprehension (similar to the one employed by the more advanced students) which targets both the two levels separately - helping students to activate relevant knowledge and to integrate information locally - as well as their relationship, helping them integrate information globally.

It seems that from both a teaching and learning perspective, comprehending SPP examples requires two "modes" of reasoning: reasoning at a local level and reasoning at a global level. The former requires the learner to reason either within the problem domain or the program domain, generating information by connecting the information extracted from the text with domain knowledge or previously generated information. The latter requires the students to traverse between the two domains, a process that requires them to understand how the program domain models aspects of the problem domain and the other way around (e.g., [9]). From a teaching perspective, therefore, the study here corroborates Schulte's [47] point that the Function side of a block of code would be better split into a local and global functionality, to allow the learners to reason locally - what the block achieves at the program level (program-level subordinate goals) - as well as globally - how the block's functionality relates to the problem (problem-level subordinate or superordinate goals), and to allow the educator to bring about these to students' attention by means of strategic inferential comprehension (e.g., *Global & Local Backward Explanations and Forward Prediction, and Associations*).

Overall, the findings highlight the potential importance of inferential strategies for developing independent comprehenders. As Davoudi[10] points out, subject-matter thinking requires students to engage with justifiable inferences with the subject's content. Although this is a small-scale study with inherent generalisation to population issues from the research design, it opens a new direction in programming education which positions one of the most important human skills, inferential reasoning, at the heart of computer science teaching and learning. Future research should focus on understanding better the role and kind of inferences students need to generate to engage with different programming tasks. Additionally, identifying inference sequential patterns (e.g., sub-section 4.2.4), and how and when these are manifested has the potential to model comprehension strategies and scaffold further students' learning and teachers' practice. Understanding these will allow us to design interventions that facilitate inference generation during various tasks and evaluate their effectiveness. The author's future research goals aim to contribute to all of these aims.

6 LIMITATIONS

The empirical part of this study shares the limitations of case studies particularly related to the generalisation of the findings which is often criticised in qualitative research. However, as Yin [1] argues, case studies do not aim to generalise the findings to populations but rather to theorisations and analytical propositions. Similarly to this, Walsham [57][p. 78] highlights four types of generalisations in relation to case studies: development of concepts, generation of theory, drawing of specific implications, and the contribution of rich insight, all of which we attempted to cover in this study.

Additionally, the results presented here are based on a specific programming language and on a small-scale program like the ones we expect our students to be able to understand. It is likely that in a larger program, with several functions or methods, an adaptation or extension of the definitions provided here will be needed.

Overall, despite the exploratory nature of this study, the three case studies presented here are cases that are usually found in programming courses; thus, they can identify and describe comprehension phenomena in the field of programming education related to the role of inferential reasoning, and how these contribute to students' learning and skill acquisition in programming.

7 CONCLUSION

The aim of this study was to investigate the role of inference generation during comprehension of SPP examples and the kind of inferences that are generated during this process. It was hypothesized that differences in the way that students construct the mental model of a given SPP may be associated with differences in inference generation during three comprehension phases: comprehension at the problem level, comprehension at the program level, and their relation. To investigate this hypothesis, an initial inferential model is first constructed stemming from literature in Psycholinguistics and Programming Education. An empirical study followed the theoretical exploration in which a comparative case study design was employed to identify and compare first-year undergraduate students' inferential strategies and their potential effect on comprehension. The findings presented above corroborated the initial hypothesis and extended the model with specific types of inferences identified during the analysis of the students' protocols. The findings of this research offer new research and teaching insights and directions in the area of program comprehension which has been a central focus in programming education.

8 ACKNOWLEDGEMENTS

I would like to thank my colleagues in the Centre of Computing Science Education at the University of Glasgow who provided constructive feedback before the submission of this paper, and Dr Marinos Kintis for the time he dedicated to act as the second coder for the qualitative analysis.

9 REFERENCES

REFERENCES

[1] Aberdeen, T. (2013). Yin, rk (2009). case study research: Design and methods. thousand oaks, ca: Sage. *The Canadian Journal of Action Research*, 14(1):69–71.

[2] Black, J. B. and Bower, G. H. (1980). Story understanding as problem-solving. *Poetics*, 9(1-3):223–250.

[3] Bos, L. T., De Koning, B. B., Wassenburg, S. I., and van der Schoot, M. (2016). Training Inference Making Skills Using a Situation Model Approach Improves Reading Comprehension. *Frontiers in Psychology*, 7(February):1–13.

[4] Bower, G. H. and Morrow, D. G. (1990). Mental models in narrative comprehension. *Science*, 247(4938):44–48.

[5] Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, 18(6):543–554.

[6] Cain, K. E. and Oakhill, J. V. (1999). Inference making ability and its relation to comprehension failure in young children. *Reading and Writing*, 11(5/6):489–503.

[7] Carlson, S. E., van den Broek, P., McMaster, K., Rapp, D. N., Bohn-Gettler, C. M., Kendeou, P., and White, M. J. (2014). Effects of Comprehension Skill on Inference Generation during Reading. *International Journal of Disability, Development and Education*, 61(3):258–274.

[8] Cohen, L., Manion, L., and Morrison, K. (2017). *Research methods in education*. routledge.

[9] Cutts, Q. and Kallia, M. (2023). Introducing modelling and code comprehension from the first days of an introductory programming class. In *Computing Education Practice*, pages 21–24.

[10] Davoudi, M. and Moghadam, H. R. H. (2015). Critical Review of the Models of Reading Comprehension with a Focus on Situation Models. *International Journal of Linguistics*, 7(5):172.

[11] Elbro, C. and Buch-Iversen, I. (2013). Activation of Background Knowledge for Inference Making: Effects on Reading Comprehension. *Scientific Studies of Reading*, 17(6):435–452.

[12] Elleman, A. M. (2017). Examining the impact of inference instruction on the literal and inferential comprehension of skilled and less skilled readers: A meta-analytic review. *Journal of Educational Psychology*, 109(6):761–781.

[13] Goldman, S. R., McCarthy, K. S., and Burkett, C. (2015). 17 interpretive inferences in literature. *Inferences during reading*, page 386.

[14] Graesser, A. C. and Kreuz, R. J. (1993). A theory of inference generation during text comprehension.

[15] Graesser, A. C., Singer, M., and Trabasso, T. (1994). Constructing inferences during narrative text comprehension. *Psychological review*, 101(3):371.

[16] Guba, E. G. (1985). Naturalistic inquiry. *Beverly Hills: Sage Publications*.

[17] Haberman, B., Averbuch, H., and Ginat, D. (2005). Is it really an algorithm: The need for explicit discourse. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 74–78.

[18] Hammadou, J. (1991). Interrelationships among prior knowledge, inference, and language proficiency in foreign language reading. *The Modern Language Journal*, 75(1):27–38.

[19] Izu, C., Schulte, C., Aggarwal, A., Cutts, Q., Duran, R., Gutica, M., Heinemann, B., Kraemer, E., Lonati, V., Mirolo, C., et al. (2019). Fostering program comprehension in novice programmers-learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, pages 27–52.

[20] Kaarbo, J. and Beasley, R. K. (1999). A practical guide to the comparative case study method in political psychology. *Political psychology*, 20(2):369–391.

[21] Kallia, M., Cutts, Q., and Looker, N. (2022). When rhetorical logic meets programming: Collective argumentative reasoning in problem-solving in programming. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*, pages 120–134.

[22] Kather, P. and Vahrenhold, J. (2021). Exploring algorithm comprehension: Linking proof and program code. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*, pages 1–10.

[23] Kendeou, P., Bohn-Gettler, C., White, M. J., and Van Den Broek, P. (2008). Children's inference generation across different media. *Journal of Research in Reading*, 31(3):259–272.

[24] Kendeou, P., McMaster, K. L., and Christ, T. J. (2016). Reading Comprehension: Core Components and Processes. *Policy Insights from the Behavioral and Brain Sciences*, 3(1):62–69.

[25] Kendeou, P., Papadopoulos, T. C., and Spanoudis, G. (2015). Reading Comprehension and PASS Theory. *Cognition, Intelligence, and Achievement: A Tribute to J. P. Das*, (January 2015):117–136.

[26] Kintsch, W. (1998). *Comprehension: A paradigm for cognition*. Cambridge university press.

[27] Kintsch, W. (2018). Revisiting the construction—integration model of text comprehension and its implications for instruction. In *Theoretical models and processes of literacy*, pages 178–203. Routledge.

[28] Korstjens, I. and Moser, A. (2018). Series: Practical guidance to qualitative research. part 4: Trustworthiness and publishing. *European Journal of General Practice*, 24(1):120–124.

[29] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., et al. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4):119–150.

[30] Lister, R., Fidge, C., and Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *Acm sigcse bulletin*, 41(3):161–165.

[31] Lister, R., Simon, B., Thompson, E., Whalley, J. L., and Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the solo taxonomy. *ACM SIGCSE Bulletin*, 38(3):118–122.

[32] Lopez, M., Whalley, J., Robbins, P., and Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*, pages 101–112.

[33] Margulieux, L. E., Morrison, B. B., and Decker, A. (2020). Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education*, 7:1–16.

[34] McGregor, S. (2018). Conceptual and theoretical papers. *Understanding and evaluating research: A critical guide*, pages 497–528.

[35] McNamara, D. S., Kintsch, E., Songer, N. B., and Kintsch, W. (1996). Are good texts always better? interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cognition and instruction*, 14(1):1–43.

[36] McNamara, D. S. and Magliano, J. (2009). Chapter 9 Toward a Comprehensive Model of Comprehension. *Psychology of Learning and Motivation - Advances in Research and Theory*, 51(December):297–384.

[37] Moshman, D. (2004). : From inference to reasoning: The construction of rationality. *Thinking & Reasoning*, 10(2):221–239.

[38] Narvaez, D., Van Den Broek, P., and Ruiz, A. B. (1999). The influence of reading purpose on inference generation and comprehension in reading. *Journal of Educational Psychology*, 91(3):488–496.

[39] Perkins, D. N. and Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. In *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers*, pages 213–229.

[40] Pretorius, E. J. (2000). *Inference generation in the reading of expository texts by university students*. PhD thesis, University of South Africa Pretoria.

[41] Radvansky, G. A., Zwaan, R. A., Curiel, J. M., and Copeland, D. E. (2001). Situation models and aging. *Psychology and aging*, 16(1):145.

[42] Raudszus, H., Segers, E., and Verhoeven, L. (2019). Situation model building ability uniquely predicts first and second language reading comprehension. *Journal of Neurolinguistics*, 50(January):106–119.

[43] Rickheit, G., Schnotz, W., and Strohner, H. (1985). The concept of inference in discourse comprehension. In *Advances in Psychology*, volume 29, pages 3–49. Elsevier.

[44] Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, 13(2):137–172.

[45] Robson, C. (2002). *Real world research: A resource for social scientists and practitioner-researchers*. Wiley-Blackwell.

[46] Schmalhofer, F., McDaniel, M. A., and Keefe, D. (2002). A unified model for predictive and bridging inferences. *Discourse Processes*, 33(2):105–132.

[47] Schulte, C. (2008). Block model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In *Proceedings of the fourth international workshop on computing education research*, pages 149–160.

[48] Simons, H. (2009). *Case study research in practice*. SAGE publications.

[49] Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9):850–858.

[50] Stake, R. E. (1995). *The art of case study research*. sage.

[51] Trabasso, T. and Magliano, J. P. (1996). Conscious understanding during comprehension. *Discourse Processes*, 21(3):255–287.

[52] Van den Broek, P. (1990). Causal inferences and the comprehension of narrative texts. In *Psychology of learning and motivation*, volume 25, pages 175–196. Elsevier.

[53] Van den Broek, P. (1994). Comprehension and memory of narrative texts: Inferences and coherence.

[54] Van den Broek, P., Lorch, R. F., Linderholm, T., and Gustafson, M. (2001). The effects of readers' goals on inference generation and memory for texts. *Memory & cognition*, 29(8):1081–1087.

[55] Van Dijk, T. A., Kintsch, W., et al. (1983). Strategies of discourse comprehension.

[56] Vaughn, S. and Bos, C. S. (2012). *Strategies for teaching students with learning and behavior problems*. Pearson Upper Saddle River, NJ.

[57] Walsham, G. (1995). Interpretive case studies in is research: nature and method. *European Journal of information systems*, 4(2):74–81.

[58] Whalley, J., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. A., and Prasad, C. (2006). An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *8th Australasian Computing Education Conference (ACE2006), Australian Computer Science Communications*, volume 28, pages 243–252. Australian Computer Society.

[59] Woolley, G. and Woolley, G. (2011). *Reading comprehension*. Springer.