Teaching Computational Reasoning Through Construals

Errol Thompson

Aston University, Birmingham, UK
E-mail: errol@wisdom.nz

ORCID ID: http://orcid.org/0000-0002-6270-2791
DOI: 10.26907/esd13.3.05

Abstract
Can construals be used to teach computational reasoning? This paper outlines some of the issues of teaching computational reasoning and then endeavours to show how it might be possible, through using the principles of variation theory to design teaching sequences and consequently construals that open the learner up to the computational reasoning ideas being considered.
Keywords: construal, computational thinking, invariant, Nim.

Обучение вычислительному мышлению используя интерпретации (Construals)

Эррол Томпсон

Астонский университет, Бирмингем, Великобритания
E-mail: errol@wisdom.nz

ORCID ID: http://orcid.org/0000-0002-6270-2791
DOI: 10.26907/esd13.3.05

Аннотация
Могут ли интерпретации быть использованы для обучения вычислительному мышлению? Эта статья рассматривает некоторые вопросы формирования вычислительного мышления и демонстрирует, как использование принципов теории вариаций, а следовательно и интерпретаций, для разработки последовательности обучения может раскрыть ученику идеи вычислительного мышления.
Ключевые слова: интерпретация, вычислительное мышление, инвариант, Ним.

Introduction

Computational thinking is a fundamental skill in computer science aimed at developing models that use computation to solve problems. In this paper, we explore how variation theory can be used to design teaching sequences that can then be used to create construals, objects that help us "explore and record our emerging understanding" (Beynon et al., 2015, p 9) and to use these to aid in the teaching of the computational concept of invariants. The emphasis is on the process by which an invariant might be discovered rather than focusing on the outcome of the discovery process.

In the first section, we explore what is meant by computational thinking and why we are using computational reasoning as an alternative description. This is followed by a discussion of variation theory and its use to make visible the object of learning. A discussion of constructionism and its relationship to programming and learning lays the focus for looking at the computational concept of invariants. This is the concept that the paper shows how through applying variation theory, a solution to two variants of the game of Nim can be solved and the learner can be aided in developing a process for discovering invariants in relevant problem spaces. It is envisaged that the process could be extended to other games or real world problems that display similar characteristics. We argue that the focus should be on making visible the process of discovery and not simply the conclusion of the process (i.e. the invariant). This is in line with the process as content approach to curriculum development (Costa & Liebmann, 1996).

From exploring these games, we seek to draw some conclusions about what makes a good construal and the features that are appropriate. We also seek to explore what should be observable and how this impacts the variations that we should use to explore a phenomenon such as computational reasoning.

Computational Reasoning

The common terminology when talking about the fundamental skill of computer science is computational thinking. However, there is no clear agreement about what this term actually means.

Aho (2011, p 7) argues that "*Computation is a process that is defined in terms of an underlying model of computation and computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms*". Wing (2006, p 33) had previously argued that "*Computational thinking involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.*" Both of these definitions are based on developing computational solutions although Wing accepts the possibility that there might be generic skills used to solve a wider range of problems.

An alternative perspective is provided by Kowalski (2011) who uses computational logic to review human instructions and to examine backward and forward reasoning. His argument is that applying computational logic can help us as humans to understand the reasoning process and to improve our approach to reasoning. Does the same apply to computational thinking or reasoning or are we talking about understanding more generic thinking skills and utilising a wider range of thinking skills?

It is Kowalski's perspective that has influenced our use of the terminology computational reasoning as opposed to computational thinking. We are arguing that computational thinking that produces a computational model is inadequate and that we need to foster an ability to reason about problems in such a way that we can verify the logic that helped us arrive at our solution. Many of the problems (i.e. tower of Hanoi, Nim, tic-tac-toe, mazes, …) that we use in computer science have solutions that are readily accessible to learners. The issue is not whether they can find these solutions but whether they can apply the process that discovered these solutions. We use our problems not to teach solutions but to teach processes or the reasoning that helps create these solutions. The skill being taught should be the problem solving process (computational reasoning). The techniques (computational thinking) form part of the tool kit to aid the computational reasoning process. The objective of our construals should be to endeavour to help the learner understand why these games have the outcomes that they do and not simply to be able to find or implement solutions. That is the learner should be able to reason about their solution and the problem space.

Variation Theory

According to variation theory, for the learner to perceive the required object of learning, the learner needs to be able to discern it from the background of other objects and to be able to discern its internal characteristics. This means that the critical aspects of the object must become visible to the learner. It is what is made visible that is possible to be learnt and not what is believed to be taught. A possible sequence for achieving this is:

Instantiation – Contrast – Generalisation – Fusion (Marton, 2015, pp. 53-54, 220)

Instantiation is "*the learner's initial encounter with the object of learning, in the form of a problem that captures holistic qualities of it*" (p. 221). Contrast involves a pattern of variance and invariance where the aspect to be discerned is varied against the invariance of other aspects (p. 49-50). Generalisation applies the opposite principle of keeping the aspect to be discerned constant while varying other aspects (p. 50-51). Fusion involves putting the parts together to reform the whole. The learner needs to be able to discern the aspect while all aspects vary (p. 51-52).

An example might be the learning of what the colour green means. Instantiation would mean introducing the learner to a green object.

Figure 1: Instantiated Circle

Contrast would be using similar shapes but different colours (see Figure 2).

Figure 2: Contrasting colours

Generalisation would involve retaining the same colour but altering the shape (see Figure 3).

Figure 3: Generalisation

Fusion would mean varying colour and shape while getting the learner to identify the colour (see Figure 4).

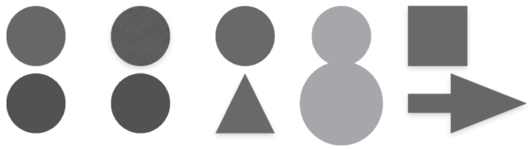


Figure 4: Fusion

If we consider a computational thinking technique then we want the learner initially to be exposed to the technique in a way that makes the technique visible and potentially enables the learner to ask questions about what the learner has seen (instantiation). Keeping the nature of the problem constant, the learner then needs to be exposed to variations in each critical aspect of the technique so that they become aware of that technique and can identify it. This can be verified by keeping that aspect constant while varying other aspects related to the technique. Finally, they need to see all aspects of the technique varied together.

Radical Empiricism and Variation Theory

Empirical modelling that is the foundational theory for the construction of construals is based on the radical empiricism proposed by James (1912). James framework is based on the argument that our conclusions or understandings should be based on fact but that they should be open to change based on new experience (James, 1987, pp vii-viii). The facts are our observations of the world and in particular how we draw together our understanding of the parts to form and understanding of the whole (James, 1909, p 7-8).

This philosophic perspective aligns well with the approach in variation theory. In variation theory and phenomenography, the variations that are discerned relate to critical aspects that help understand the whole, the phenomenon. These critical aspects are characteristics that aid the learner in understanding the object of learning. They could relate to the internal parts of the object of learning or to characteristics that distinguish the object of learning from its environment (Marton & Booth, 1997, pp 86-87). What the learner discerns impacts their understanding of the object of learning. The learner's understanding may differ from that of the teacher because of the way that the learner has experienced the object of learning.

Construals and Empirical Modelling

The objective in creating a construal is to enable the learner to observe the construal's internal workings and to form an understanding through experimentation. Beynon (2009, p. 75) describes a construal in terms of observables, dependencies, and agents. An observable reflects an internal value of the construal that can be seen as meaningful in the external context. A dependency defines a relationship between observables. This might be a formula for a calculation (i.e. in modelling movement, modifying the rate of acceleration should cause a corresponding change in the speed) that is triggered by the change of a value in an observable. Agents may be human or non-human. They are able to recognise changes to observables and make appropriate responses. A non-human agent may initiate a response automatically but differs from a dependency in that it is initiated based on the detection of a specific state occurring rather than simply responding to the change in value of an observable.

The learner constructs understanding through interaction with a model, a construal, that aids them in developing their understanding the phenomena being studied. This learning is achieved through observation of changes caused by dependencies between observables and agent initiated actions (Beynon, 2009, p 75). The model developed for the learner enables then to modify the values of observables and defined dependencies and agents would cause related observables to change and be reflected to the user. The intent is not to provide the learner with a blank sheet in which to construct a model but to provide a model with which the learner can observe and experiment (Beynon et al., 2015).

In the case of the construal to be discussed later, the modeller has explicitly used a visual representation of the state of the game. The learner can influence the state by simply playing the game or through setting values in key observables. If the learner wishes, they can observe some of the underlying values in the observables that are not immediately visible and create their own expression to decide how many stones to take. The environment also enables the learner to explore the underlying scripts what is displayed and that define dependencies between observables.

The aim of this paper is not to describe the construal but to examine how this environment can be used to implement a variation theory approach to learning.

Construals for computational reasoning

Applying the principles of variation theory to computational reasoning means trying to expose the critical aspects of the computational techniques. Combining variation theory with those of using observables, dependencies, and agency in construals, this paper explores how we might construct a construal that enables the learner to develop a computational understanding of the game and as a consequence a game playing strategies.

To illustrate these issues, we will use variants of the Nim game that have predictable outcomes.

Invariants

Zingaro (2008) defines invariants as "*properties of program segments that remain true throughout the scope to which the invariant applies*" (p 2). If we want learners to use invariants to aid them in writing their code then we need to help them identify invariants in the problems that they are trying to solve in the code segments that they are trying to write.

Seven Stone Nim

Seven stone Nim is a game involving two players. Starting with a pile of seven stones (Figure 5), each player, on their turn, takes one or two stones. They cannot pass on their turn. The player who takes the last stone wins.

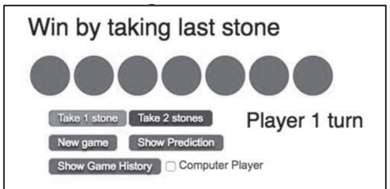
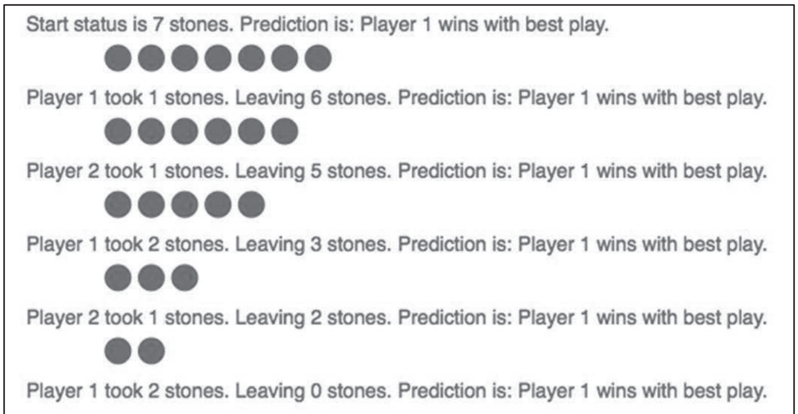


Figure 5: Seven stone Nim start state.

This visual representation is intended to provide an interface that reflects what might be expected for an implementation of the game. It is a visualisation of underlying observables (i.e. the number of stones remaining (*numStones*). In the construal environment, the learner could manipulate this value directly by using the statement *numStones = 4;* to see the impact on the visual representation. An ability to influence the number of starting stones is provided but this is only visible in the visualisation when a new game is started.

With this interface, two learners can play the game or the learner can play against the computer. Using the "Show Prediction" button, the learner can see what the construal predicts the outcome should be if the best playing strategy is used. The show game history allows the learner to see the play sequence history and reflect on the game strategy. The history for a best strategy game play is given in Figure 6.


Start status is 7 stones. Prediction is: Player 1 wins with best play.

Player 1 took 1 stones. Leaving 6 stones. Prediction is: Player 1 wins with best play.

Player 2 took 1 stones. Leaving 5 stones. Prediction is: Player 1 wins with best play.

Player 1 took 2 stones. Leaving 3 stones. Prediction is: Player 1 wins with best play.

Player 2 took 1 stones. Leaving 2 stones. Prediction is: Player 1 wins with best play.

Player 1 took 2 stones. Leaving 0 stones. Prediction is: Player 1 wins with best play.

Figure 6: A sample game showing the play history

We want the learner to develop a strategy for understanding the game through identifying the winning game invariant. Applying the principles of variation theory and the sequence to foster awareness and understanding of what an invariant is, we applied the following sequence. This strategy was initially used in tutorials with students using toothpicks as the stones.

Note: There are other invariant possibilities with the game such as the number of stones remaining is between the initial number of stones and zero stones but these are not the focus of this exercise.

Instantiation

The first step is to have the students play the game having asked the question "is there a way to ensure that you can win?" In many instances, the students will play the game and one or other player will win. Can they determine who will win? Generally, not. Part of the issue here is that the student often does not understand the nature of the game or the mathematical principle that underlies the winning of the game.

Contrast

To show contrast, we want to vary the critical aspect while keeping the other aspects constant. The problem with invariance is that we want to expose something that does not change. However, there is a pattern of variance with respect to the game state that does expose the invariance and this is the prediction of who will win in relation to the number of stones remaining (see Figure 6). If we pause the game at critical points, can the players determine who will win?

When there are just one or two stones left, the answer to who will win is relatively obvious. If you pause at three stones, often the person whose turn it is realises they cannot win but not always. At four stones left, can they now reason about how many stones they should take if they want to win? Hopefully, they reason that it is one so they leave their opponent with three. If they have that clear, can they reason about how many stones to take if five stones are left? Can they now predict who is likely to win and why they are likely to win? How general is their theory of prediction? Will it work with six stones and seven stones? How do they word their rule for their prediction? Although the invariant "*number of stones left after my turn must be a multiple of 3 if I am to win*" is important, it is the reasoning that gets to that invariant that is important for computational reasoning. The rule for my move is to take the maximum of 1 stone or the remainder of dividing the number of stones by three.

An alternative here would be to have the learner to alter the number of stones remaining and endeavour to predict the outcome. This would not require the viewing of the history in the construal but the prediction could be used to confirm the learner's prediction.

It may take several times of playing the game with pauses or manipulating the number of stones remaining and questions for them to come to this conclusion. As the facilitator of their learning, you want to pause and challenge them to think at each stage of the game so the invariant is exposed (multiple of three). The critical aspect from the variation theory perspective is the number of stones remaining to predict a win.

Generalisation

Changing the number of stones at the start of the game confirms the invariant but we contend that it is not exposing the process. In fact, our contrast sequence may not be showing the process. Is there a variation of the game or a very similar game for which the same process of deriving an invariant applies?

What happens if we change the win rule to the winner is the person who forces the other player to take the last stone? When can we begin to predict a winner? Do the learner's follow the same process as for the previous version of Nim?

In terms of the process of revealing the concept of an invariant, we have begun the process of generalisation but in terms of the computational reasoning process for discerning invariants, we are still in contrast mode.

To generalise the process of discerning an invariant, we want a variation of the game that still has an invariant but has greater complexity. This is possible using the generic Nim game.

The seven stone Nim construal

The version of the game implemented by Thompson (2017) enables the learner to play the game. It also provides the opportunity to observe the prediction of who will win for each state, and the ability to view the history. To encourage the learner to try and construct an invariant rule, it has the capability for the learner to create a formula for deciding how many stones to take. There is also the option to change the winning rule so that the player who takes the last stone loses. This changes the behaviour of the construal while leaving the features of the game in place.

A return to generalisation

Nim is what is known as a mesére game (Siegel, 2008). The generic Nim game starts with any number of piles of coins. In their turn, a player can take as many coins as they want from a single pile. The person who takes the last stone wins. A pile can have any number of coins at the start of the game. For our study of the game, we will use a maximum of three piles.

Beynon (2017) implements this variant as a construal and uses a very similar strategy to that described for seven stone Nim to help the learner understand how to win the game. He instantiates the game so that the learner can play the game. He also raises some questions and provides some sample starting points that try to encourage understanding of the winning strategy invariant. What he makes visible is a number of key observables and the Nim sum (exclusive or) calculation.

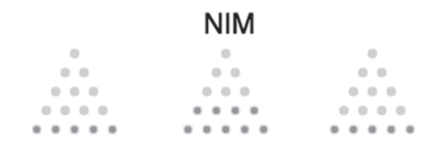


Figure 7: Nim Coins

In Beynon's construal (Figure 7), the red dots represent coins while the grey dots are empty locations. The states of the three piles can be represented by three numbers (i.e. [5, 9, 5] for Figure 7).

Following the strategy used for the seven stone Nim game, the learner should be exposed to an initial state [n, 0, 0] that would give a win. This establishes that in order to win, they need to be presented with a state where only one pile has coins. The next starting state would be to start with two piles such as [1, 1] which they would hopefully see as a losing state and then with [2, 1] followed by [2, 2] to allows them to explore how they can leave their opposition in a losing state. With [2, 1], they want to leave their opponent with [1, 1] which they know forces a loss. But what of [2, 2], can they get their opposition to a guaranteed losing state? They cannot get to [1, 1] so that means they need a state which might allow their opponent to make a mistake. The only safe possibility is [2, 1].

Encouraging experimentation with [n, n] and [m, n] combinations will reinforce the idea that if there are only two piles in play then you want to leave your opposition with two piles with the same number of coins.

The strategy that we are encouraging for experimentation is to start with known outcomes and then explore options that would enable them to reach those known outcomes. Taking this approach, a step further would suggest that if you add another column of coins then maybe you can predict what will happen if two columns have the same number of coins (i.e. [n, n, m]). Do they see that removing the m coins from the third column gives the opposition a losing state?

Now, we are ready to look at cases where the number of coins in each column is different (i.e. [3, 2, 1]). Regardless of what they do, they are going to end up with an [n, n, m] state which is a potential win for the opposition. They might see that they cannot get to a state that would force the opposition to lose.

Move to a maximum of 4 coins in a pile. They should be aware that [4, n, n] is a potentially winning state since they can leave their opposition with [0, n, n]. What about combinations where each pile has a different number of coins (i.e. [4, 2, 1], [4, 3, 1], and [4, 3, 2]). Can they see that these all reduce to [3, 2, 1]? Can they see that when the maximum number of stones in a pile is 4 that only [4, 4, 0] has the characteristics of a losing position?

The table in the appendix was developed by the author to try and determine how you arrive and the Nim sum conclusion that the numbers for a losing state when using their binary representation exclusive or ($\oplus$) to binary zero. It uses a similar strategy to that used for the seven coin Nim described above. It starts with the easily predictable outcomes (i.e. [1, 0, 0] and [1, 1, 0]) and builds up the number of coins used. It recognises consistent patterns to avoid having to experiment with every option.

It may be possible that the learner may see that there is something unique about numbers that are powers of 2 (i.e. 2, 4, 8, …) in that only the [n, n, 0] combination is a potentially losing position. It is important that they also see that once in a losing position, it is difficult to force the opposition into a losing position but from a potentially winning position, there is only one unique reduction that will leave the opposition in a losing state. That is you can lose the game but unless you know the losing states, you cannot force a win. What they are looking for is the rule that will enable them to know how to determine those states. In seven stone Nim, this was the multiple of 3. By exposing of the Nim sum as they play, does the learner determine that a Nim sum of zero determines a losing state?

Having completed at least two variants of the game, we take some time to review the process for coming to define the invariant through seeing whether the learner can describe the process and why it worked?

Siegel (2008) says that the Nim sum applies to any number of columns. The next step is to see whether the learner can determine that this is true for a version of the game with more than three columns before moving to a different type of problem. At this point, we have not conducted an experiment to see whether the learner can apply the process to another version of the game.

Expanding the context – real generalisation and fusion

To this point, we have been using very similar games but we want our learners to be able to apply the problem solving process to a much broader context including programming and real world situations. This is what is missing from many of our strategies of using programming to teach computational ideas.

There are other games with invariants such as tower of Hanoi, missionaries and cannibals, jealous husbands, or getting soldiers across the river. Like the set of Nim games, these all follow a particular pattern to arrive at a solution and have a very simple invariant rule. Although we have used the strategy with some of these problems with some positive outcomes, we are not yet convinced that the learners have developed a problem solving strategy.

Discussion

For the transfer of the computational thinking technique, it is not simply discerning the invariant concept in a particular type of problem space (i.e. the writing of a code segment) but in a range of different problems possibly from different problem domains. We start with games but transfer the concept to programming. We contend that we should include some everyday real world problems to encourage transfer and to verify that transfer is occurring.

The construit environment is an attempt to apply the principles of Papert's (1980) constructivism theory. The learner constructs a model as a way of building an understanding of a phenomenon. With the models discussed in this paper, we are not asking the learner to start constructing a model from scratch. Rather, we are asking them to manipulate part of the model to build an understanding of the problem that they are working with.

The construit language is designed to aid the learner build a model. Both Piaget and Papert contend that a child constructs knowledge through interaction with their world (Ackermann, 2001). By endeavouring to ensure that the learner is exposed to the critical aspects of the desired knowledge then we are exposing them to opportunities to construct the desired knowledge. Variation theory (Marton, 2015) helps us identify those critical aspects and guides us in how to expose them to the learner.

In the case of seven stone Nim, the variations used in the construal were developed through interaction with learners in tutorials where they played the game using toothpicks. It was observed that stopping the game at specific points and asking them to reason about the outcome helped them develop an invariant rule to determine who would win and then to a game play strategy to win the game. Similar strategies have been used with noughts and crosses (tic-tac-toe) and tower of Hanoi. Beynon's (2017) Nim coins utilises a similar approach but instead of allowing the learner to draw their own conclusion as to the winning strategy informs the learner of the invariant rule. We see it as important that the learner develops their understanding of the invariant rule and the process to arrive at a rule.

Conclusion

We argue that providing learners with construals that encourage them to focus on experimentation around the critical aspects of the problem can aid their learning. Asking the learner to develop a model of the problem space is problematic as the learner has to learn the modelling environment and how to represent the problem in that environment before they can focus on developing the knowledge to solve the problem. This is a major difficulty with teaching programming.

This paper endeavours to reveal how we might be able to expose the computational reasoning ideas through using variation theory and construals but the paper also attempts show how we might use variation theory to teach computational reasoning ideas. The work is still in preliminary form and needs further verification to ensure that the strategies described produce the desired outcomes.

Appendix: **Detailed Analysis of the Nim Game**

Game State	Nim Sum	Prediction	Justification for prediction
[n, 0, 0]		Win	Simply take all n coins.
[n. n, 0]	n n 000 000	Lose	[1, 1, 0] is losing position and all [n, n, 0] are reduced to an [m, n, 0] followed by a new [n, n, 0] until it reaches [1, 1, 0]. Nim sum is always zero when two columns have the same number of stones and the third column is empty.

Game State	Nim Sum	Prediction	Justification for prediction
[n, n, m]	n n m m	Win	Take all m coins reduces to [n, n, 0], a losing position for the opposition. All other reductions lead to a potentially winning position for the opposition although this needs further proof at this point. The Nim sum is always the binary representation of m since the Nim sum of n $\oplus$ n = 0.
[m, n, 0]		Win	n < m. Remove m – n stones from m pile and you have [n, n, 0], a losing position for the opposition. Failure to give the opposition an [n, n, 0] potentially puts you in a losing position.
[2, 2, 0]	10 10 00 00	Lose	**Note:** Only [2, 2, 0] is a losing position where the maximum number of coins is 2. The next state is either [2, 1, 0] or [2, 0, 0]. If your opponent knows how to play, they can keep you in a losing state.
[3, 2, 1]	11 10 01 00	Lose	All possible move combinations end with a possible winning state for the opposition. All other combinations with a maximum of 3 coins are covered by the previous cases. For [3, 2, 1] this would be an [n, n, m] combination or an [m, n, 0] combination. For [3, 3, 0] it would be an [3, n, 0] combination where n is less than 3.
[3, 3, 0]	11 11 00 00		
[4, 4, 0]	100 100 000 000	Lose	**Note:** Only [4, 4, 0] is a losing position where the maximum number of coins is 4. No other number has a 1 in the most significant column of its binary representation.
[4, 2, 1]	100 010 001 111	Win	All can be reduced to [3, 2, 1] which is a losing position for opposition. Other reductions potentially leave your opposition in a winning position.
[4, 3, 2]	100 011 010 101		
[4, 3, 1]	100 011 001 110		**Note:** 4 = 3 + 1 but we are in a losing state and not a winning state so this refutes any rule based on decimal arithmetic. It may be indicating that we should be looking at binary notation since 4 requires 3 binary digits to represent while 3 requires 2 and 1 requires only 1.
[5, 4, 1]	101 100 001 000	Lose	Reduce to [5, n, 1] where n < 4 or [4, n, 1] where n < 5 of [5, 4, 0] which are all win situations for the opposition.
[5, m, n]	101 0yy 0zz 1xx	Win	Where m and n are less than 4 and n < m since they can be reduced to [3, 2, 1], a losing position for the opposition. The possible initial combinations are [5, 3, 2], [5, 3, 1], [5, 2, 1]. All other possible reductions leave the opposition in a potentially winning state. The most significant digit on the Nim sum is always 1.
[5, 4, n]	101 100 01z 01x	Win	Where n = 3 or 2. Reduces to [5, 4, 1] which are is a losing position for the opposition. Other reductions leave the opposition in a potentially winning state. The middle digit of the Nim sum is always 1.

Game State	Nim Sum	Prediction	Justification for prediction
[6, 4, 2]	110 100 010 000	Lose	All [4, 2, n] where n < 5, [6, 4, n] where n < 2, and [6, 2, n] where n < 4 are win positions for the opposition.
[6, 5, 3]	110 101 011 000		All [6, 5, n] where n < 3, [6, 3, n] where n < 5, and [5, 3, n] where n < 6 are win positions for the opposition. **Note:** [6, 5, 3] is the first combination for a loss where the decimal sum of the two smaller numbers does not equal the larger number (i.e. 6 ≠ 5 +3).
[6, 6, 0]	110 110 000 000		
[6, 4, 1]	110 100 001 011	Win	Reduces to [5, 4, 1] which is a losing position for the opposition. **Note:** This is the best play from this state other options potentially led to a possible win state for the opposition.
[6, m, n]	110 0yy 0zz 1xx	Win	Where m and n are less than 4 and n < m since they can be reduced to [3, 2, 1], a losing position for the opposition. All other reductions leave the opposition in a potentially winning position. The most significant digit of the Nim sum is always 1.
[6, 4, n]	110 100 yyy xxx	Win	Where n = 3, 4, or 5 reduced to [6, 4, 2] which is a losing position for the opposition. All other reductions leave the opposition in a potentially winning position. If n is 3 then the last digit of the Nim sum is 1. If n is 4 or 5 then first digit of the Nim sum is 1.
[6, 5, n]	110 101 yyy xxx	Win	Where n < 5 but not equal to 3. Cases are [6, 5, 4] (Nim sum 111) -> [1, 5, 4], [6, 5, 2] (001) -> [6, 4, 2], [6, 5, 1] (010) -> [4, 5, 1]. All other reductions leave the opposition in a potentially winning position. yyy would need to be 3 [011] in order for the Nim sum to be zero.
[7, 4, 3]	111 100 011 000	Lose	All [4, 3, n] options are wins for the opposition . All other [7, 4, n] and [7, 3, n] combinations that can be reached from this starting position are covered below and are potential win states for the opposition.
[7, 5, 2]	111 101 010 000		All [5, 2, n] options are wins for the opposition . All other [7, 5, n] and [7, 2, n] combinations that can be reached from this starting position are covered below and are potential win states for the opposition.
[7, 6, 1]	111 110 001 000		All [6, 1, n] options are wins for the opposition. All other [7, 6, n] and [7, 1, n] combinations that can be reached from this starting position are covered below and are potential win states for the opposition.
[7, 7, 0]	111 111 000 000		

Game State	Nim Sum	Prediction	Justification for prediction
[7, m, n]	111 0yy 0zz 1xx	Win	Where m and n are less than 4 and n < m since they can be reduced to [3, 2, 1], a losing position for the opposition. All other reductions leave the opposition in a potentially winning position. The most significant digit of the Nim sum is always 1.
[7, 4, 1]	111 100 001 010	Win	[7, 4, 1] reduces to [5, 4, 1]. All other reductions leave the opposition in a potentially winning position.
[7, 4, 2]	111 010 001 100		[7, 4, 2] reduces to [6, 4, 2]. All other reductions leave the opposition in a potentially winning position.
[7, 5, 1]	111 101 001 011		[7, 5, 1] reduces to [5, 4, 1]. All other reductions leave the opposition in a potentially winning position.
[7, 5, n]	111 101 yyy xxx		[7, 5, n] where n > 2 reduce to [7, 5, 2]. All other reductions leave the opposition in a potentially winning position. **Note:** Only n = 2 will give a Nim sum of zero.
[7, 6, n]	111 110 yyy xxx		[7, 6, n] where n > 1 reduce to [7, 6, 1]. All other reductions leave the opposition in a potentially winning position. **Note:** Only n = 1 will give a Nim sum of zero.
[8, 8, 0]		Lose	**Note:** Only [8, 8, 0] is a losing position where the maximum number of coins is 8
[8, 2, 1]		Win	[8, 2, 1] reduces to [3, 2, 1], a losing position for the opposition
[8, 3, n]			[8, 3, 1] and [8, 3, 2] both reduce to [3, 2, 1]
[8, 4, n]			[8, 4, 1] reduces to [5, 4, 1]. [8, 4, 2] reduces to [6, 4, 2]. [8, 4, 3] reduces to [7, 4, 3].
[8, 5, n]			[8, 5, 1] reduces to [5, 4, 1]. [8, 5, 2] reduces to [7, 5, 2]. [8, 5, 3] reduces to [6, 5, 3]. [8, 5, 4] reduces to [5, 4, 1].
[8, 6, n]			[8, 6, 1] reduces to [7, 6, 1]. [8, 6, 2] reduces to [6, 4, 2]. [8, 6, 3] reduces to [6, 5, 3]. [8, 6, 4] reduces to [6, 4, 2]. [8, 6, 5] reduces to [6, 5, 3].
[8, 7, n]			[8, 7, 1] reduces to [7, 6, 1]. [8, 7, 2] reduces to [7, 5, 2]. [8, 7, 3] reduces to [7, 4, 3]. [8, 7, 4] reduces to [7, 4, 3]. [8, 7, 5] reduces to [7, 5, 2]. [8, 7, 6] reduces to [7, 6, 1].
[9, 8, 1] [9, 9, 0]		Lose	
[9, m, n]		Win	Where m and n are less than 8 and n < m since they can be reduced to a losing position for the opposition
[9, 8, n]		Win	Where n > 1 and n < 8 reduce to [9, 8, 1]

References

Ackermann, E. (2001). *Piaget's constructivism, Papert's constructionism: What's the difference?* Paper presented at the Constructivism: Uses and perspetives in education, Geneva. Retrieved from: http://learning.media.mit.edu/content/publications/EA.Piaget _ Papert.pdf

Aho, A. V. (2011). Ubiquity symposium: Computation and Computational Thinking. *Ubiquity, 2011*(January). doi:10.1145/1922681.1922682

Beynon, M. (2017). Nim Coins. Construit!: University of Warwick. Retrieved from: http://jseden.dcs.warwick.ac.uk/construit/?load=166

Beynon, M. (2009). Constructivist Computer Science Education Reconstructed. *Innovation in Teaching and Learning in Information and Computer Sciences*, *8*(2), 73–90. https://doi.org/10.11120/ital.2009.08020073

Beynon, M., Foss, J., Hudnott, E., Russ, S., Hall, C., Boyatt, R., . . . Winczer, M. (2015). *Making Construals as a New Digital Skill: Dissolving the Program – and the Programmer – Interface*. Paper presented at the International Conference on Interactive Technologies and Games (ITAG). https://doi.org/10.1109/iTAG.2015.10

Costa, A. L., & Liebmann, R. M. (1996). *Envisioning process as content: Toward a renaissance curriculum*. Thopusand Oaks, CA: Corwin Press Inc.

James, W. (1897). *The will to believe and other essays in popular philosophy* (1912 / Project Gutenberg). New York: Longmans, Green and Co. Retrieved from: http://www.gutenberg.org/ebooks/26659

James, W. (1909). *A Pluralistic Universe Hibbert Lectures at Manchester College on the Present Situation in Philosophy*. London, Bombay, and Calcutta: Longmans, Green and Co. Retrieved from: http://www.gutenberg.org/ebooks/11984

James, W. (1912). *Essays in Radical Empiricism*. (R. B. Perry, Ed.). New York: Longmans, Green, and Co. Retrieved from: http://www.gutenberg.org/ebooks/32547

Kowalski, R. (2011). *Computational logic and human thinking: how to be artificially intelligent*. Cambridge; New York: Cambridge University Press.

Marton, F. (2015). *Necessary conditions of learning*. New York and London: Routledge.

Marton, F., & Booth, S. A. (1997). *Learning and awareness*. Mahwah, NJ: Lawrence Erlbaum Associates.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*: Basic Books, Inc.

Siegel, A. N. (2008). *Misére games and misére quotients*. Course Notes. Weizmann Institute of Science. Rehovot, Israel. Retrieved from: https://arxiv.org/pdf/math/0612616.pdf

Thompson, E. (2017). Seven Stone Nim. Construit!: University of Warwick. Retrieved from: http://jseden.dcs.warwick.ac.uk/construit/?load=236

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35. Retrieved from: https://doi.org/10.1145/1118178.1118215

Zingaro, D. (2008). *Invariants: A Generative approach to programming*. London: College Publications.