Mistakes, Alternative Conceptions and Prior Knowledge

The following is a short discussion paper that aims to develop a framework you can use to think about some of the causes of recurring mistakes that your pupils make in Computing Science. It draws on recent research in cognitive science, educational psychology and CS education research. While reading through this paper try to think how this relates to your own experience, and what common types of error you see students making each year in different Computing courses.

Mistakes you may have seen before

Picture the scene, we've just spent several weeks gradually building up pupils' knowledge and skills of the various programming concepts and how they can be applied. The class seems fairly confident so we set a slightly more open ended programming challenge. For some, the wheels just seem to fall off; much staring at a blank screen ensues and we end up with a portion of the class looking like the next set of extras in a zombie movie.

When they do get going we end up with several mistakes in their code that don't look like careless typos. Three mistakes I've seen crop up over the years are

1. Being unable to convert a condition in English into the correct complex condition in code. So when a pupil sees in a problem description "Check age is 18 or over or 65 and under" that means Age >=18 or < 65 rather than Age >=18 and Age <= 65.
2. Getting variables or expressions with variables round the wrong way in assignment statements. So we end up with current_value = old_value rather than old_value = current_value or score + 1 = new_score.
3. Failing to realise that the computer will compute the value of an expression and store the result rather than the expression itself. So when asked what the value of X is when X = Y + 1 they respond Y + 1 or just give a shrug of the shoulders.

Explaining these in terms of alternative conceptions

Mistakes can be an opportunity for pupils to really learn something they didn't know before but only if they can work through why something they believe doesn't apply, rather than just feeling stupid and giving up in frustration. To help them through this muddle we really need to be aware of what alternative conceptions they have so that we can guide them through their confusion to a more refined understanding of a concept or idea.

What exactly is an alternative conception?

Alternative conceptions are also more commonly known as misconceptions. They are ways of thinking about a particular phenomenon in a less familiar area that lead to novices coming to the wrong conclusion. However a misconception implies a novice has done something wrong when in reality they don't have enough experience in the new area to make much sense of what is correct or incorrect. For this reason many researchers have started to refer to them as alternative conceptions instead.

Not all novice mistakes are because of alternative conceptions. The ones that are tend to crop up more than once and can be justified by the pupil and are fairly resistant to change. Just telling a novice that they're wrong when they have an alternative conception really doesn't help that much. The other tricky thing about alternative conceptions is they're hard to detect using regular tests or other assessments.

Why do they exist?

In general, alternative conceptions exist because, as human beings, we're constantly trying to come up with explanations for why things are the way they are or why they've changed. Even the most switched off learner in the classroom will engage in this kind of behaviour with a subject or interest they have. If they don't have detailed knowledge in a particular area then they'll try to compensate by drawing on other prior knowledge or trying to extend what they do know about it instead.

More specifically in Computing, pupils' day to day experience with using digital technology can lead to them to fail to realise that all of the devices they use are really just general purpose information processing systems. After all, computers are the ultimate masters of disguise, embedded in almost every technology we use! In these circumstances What You See Is What You Get can all too often become What You See Is All There Is for most people. This obscures the underlying principles that apply to all of these devices and gives plenty of scope for alternative conceptions to arise.

How can alternative conceptions help us?

Alternative conceptions give us a way of being able to explain the reasons for students' faulty thinking which is often the first step in being able to lead them through it to a more productive way of thinking. It also allows us to try to determine whether a particular mistake is likely to be a one off event or will crop up in other circumstances.

Coming back to the three mistakes I shared with you earlier we can think about them in terms of what alternative conceptions pupils may have.

1. Check age is 18 or over or 65 and under" that means Age >=18 or < 65 rather than Age >=18 and Age <= 65. There are several possible causes for this with the three most likely alternative conceptions feeding into this mistake being
    a. Natural language "or" works in the same way as a logical "or".
    b. The computer will infer anything that's missing from the information provided so there's no need to provide Age again in the second comparison.
    c. Less than means up to and including the end of the comparison value so 65 is acceptable but 66 will not.
2. current_value = old_value rather than old_value = current_value. Two equally plausible explanations would be
    a. Like equations in Maths, the order of the sides doesn't matter because = means the two things are equivalent.
    b. The computer carries out parts of an instruction in standard, left to right, reading order.
3. The value of X in X = Y + 1 is Y + 1. The most likely suspect for this alternative conception is Maths related.
    a. X = Y + 1 means that X is equivalent to Y + 1 rather than X is storing the result of the current value of Y with 1 added to it.

Performance of teachers who know the common alternative conceptions

Although we don't have specific large scale experimental data for CS yet there has been a recent large-scale study that looked at the influences of teacher subject matter knowledge and knowledge of misconceptions in the teaching of middle school physical science. It provides some useful indications of the size of effect that being aware of the most common alternative conceptions has on pupils learning.

For pupils with reasonable literacy and numeracy skills, in tricky areas with many potential alternative conceptions they achieved an effect size 0.7 if their teacher also had knowledge of what these alternative conceptions were likely to be (an effect size of 0.7-0.9 is roughly equivalent to an

improvement of one whole grade). What is really interesting is that teachers' subject matter knowledge made almost no extra difference in these areas compared to poor subject matter knowledge, an effect size of 0.52 compared to 0.44 (roughly equivalent to half a grade which is the usual difference that having a teacher makes).

For difficult areas with alternative conceptions it seems that good subject matter knowledge alone is not enough to improve student learning over and above the normal effect that having a teacher has. To improve learning of more challenging concepts we need to understand what the alternative conceptions are likely to be.

Problem solved or is it?

Understanding pupils' alternative conceptions seems to be fairly important if we want them to make better progress in the more difficult areas of Computing Science. If we had a list of the alternative conceptions we could adapt our teaching to make sure that these either didn't develop or we addressed them when we taught a new concept. Unfortunately there are three rather large flies in the alternative conceptions ointment which mean we have to dig a bit deeper. These are

1. Conceptual change is difficult.
2. There are lots of potential alternative conceptions
3. Lack of enough research in non-programming specific CS topics

Conceptual change is difficult

Research in how novices understanding changes in science education has discovered that many alternative conceptions are resistant to change. One popular, and regularly used, model of conceptual change indicates that four conditions must be met in order for a change to take place. These are dissatisfaction with the existing conception and the new conception being intelligible, plausible and fruitful. Fruitful being that the new idea can explain or resolve problems the original concept could not and potentially be extended in new ways. If any of these conditions fail to be met then learners will maintain their original conception and no change will occur.

Lots of potential alternative conceptions

Although there are probably less alternative conceptions in an area such as Computing Science than mistakes they still form a considerable collection. For example the list of alternative conceptions we'll provide for you in your next session just covers programming up to Higher but has over 80 entries in it. Multiply that by the number of other areas we teach and you'd end up with a fairly daunting list. Such a list unfortunately doesn't exist due to the third and final problem below.

Lack of a body research in many CS topics

There's plenty of research into programming misconceptions spanning nearly four decades however there isn't as large a volume of studies for other topics. This is problematic as the alternative conceptions identified in CS Education research that do exist only appear in one or two studies making it difficult to assess how credible and widespread they are.

Understanding the causes to avoid papering over the cracks

If we can't just rely on a fully developed body of knowledge of CS alternative conceptions then what can we do? Two approaches are to try to understand some of the root causes of the alternative conceptions and address these in our teaching and try to avoid pupils developing them in the first place.

Pupils build on prior knowledge

One observation we can use to help us is that novices try to build on prior knowledge even in a new topic where they don't have much experience. They can draw this knowledge from

1. Prior knowledge in another subject domain
2. General day to day experience
3. Analogies and metaphors the teacher uses
4. Trying to extend existing CS knowledge they do know.

Prior knowledge in another subject domain

Although we may believe that developing the underlying understanding may be important, many of us have limited time with students in the BGE phase in S1-3 compared to other subjects. When students do join us they bring with them a wealth of experience that we can make use of but that may also interfere with understanding specific concepts in Computing. Maths has often been considered an important pre-requisite for success in Computing Science, in spite of mixed research evidence on what impact it actually has, but it can also hinder the understanding of key concepts.

As we mentioned earlier, the meaning of = in Mathematics implies that the two sides of an equation are equivalent and the terms can appear on either side. Compare this to the process of assignment and = means something very different indeed; in this case it means execute the expression on the right hand side and then store the result in a named location in the computer memory on the left hand side.

Another potential source of confusion are the symbols used for common operations such as multiplication and division and the ability in Maths to use proximity of a number and a term such as y or a bracket to indicate multiplication. Although Mathematics may be considered to be closest in similarity to Computing Science other subject domains such as English and Modern Languages also provide experiences that novices may try to draw on such as the meaning of keywords in a high level language.

General day to day experience

Giving instructions to someone else in a conversational setting is something that many of our pupils will have direct experience of. Whether it's from supervising younger children or groups in school or explaining the rules of a playground game the experience is largely the same. Instructions are given in an interactive fashion with the participants able to ask for clarification or further explanation if they need it. The instructions don't have to be very precise as the person can also demonstrate what they mean or the person following them can fill in some of the gaps from their background knowledge. When a similar approach is used in the unforgiving world of the computer the result can be frustration and confusion about why a particular piece of code or complex action is not working correctly particularly if the novice thinks the machine has enough information to be able to figure it out.

Analogies and metaphors the teacher uses

Sometimes an analogy or metaphor helps us to bridge the gap between a new idea and something students can relate to. Many other teachers we've worked with, or I've spoken to, use the analogy of storing a value in a box when explaining variables. However students can, and often do, overextend the analogy or metaphor in inappropriate ways which can cause more problems than the analogy or metaphor originally solved.

For example, boxes in the real world often have multiple items stored in them and aren't always labelled. When you place a new item in the box the existing item is pushed to the side instead of being overwritten. Finally when we want to store an item in another box, we don't copy it but instead remove it from the first storage location and then place it in the second one. All of those experiences can end up being transferred across to the new concept with potentially disastrous consequences.

Students may end up believing that simple variables store a history of different values instead of just one value and that previous values are kept rather than overwritten. When a variable is on the right hand side of an expression its value is transferred to the new location rather than being copied to the new location. All of which are completely incorrect assumptions to make about how variables and variable assignment work but understandable given the ways in which the analogy and their prior experience can mislead them.

Trying to extend existing CS knowledge they do know

Students may have plenty of experience using particular commands but may end up overgeneralising or undergeneralising what they know. The print function is one of the most common instructions that pupils may use but I've seen students overgeneralising by trying to print a whole array, placing just its name inside of a print statement instead of iterating through each value with a loop. When they have more than two pieces of data to display on one line they may become stuck because they've under generalised and don't realise that multiple pieces of text and variables can be provided as arguments to the print statement instead of always one or two actual parameters.

What is the common trigger?

A large part of Computing Science involves teaching how various information processes work. To do this effectively pupils have to eventually develop a dynamic understanding of the process in action whether it's the Fetch Execute cycle or what happens when the computer executes a particular high level programming instruction. This understanding is not of the actual machine itself but rather a simplified model that contains enough detail to be able to explain what the computer is doing at a particular level of abstraction. When students are unaware of how the process works many of them will try to come up with an explanation of their own often resulting in alternative conceptions. In Computing Science these are likely to cause difficulties and block progress in areas with a tightly connected set of concepts such as programming or in the creation of information systems.

Food for thought

We've discussed common mistakes that pupils make and a way of understanding them in terms of alternative conceptions and the prior knowledge they spring from. Several things you might like to think about and discuss on your hub course forum with other members are

1. How might tracing (possibly using TRACS) help novices avoid developing strongly held alternative conceptions of programming concepts?
2. What other ways may you use knowledge of alternative conceptions and their causes to inform your teaching?
3. What steps might you need to take in order to be able to change a strongly held alternative conception?
4. What persistent and common mistakes have you seen your own pupils make in other topics and what do you think their alternative conception or conceptions could be?

In the upcoming session we will be using an N5 level programming concepts test to highlight possible alternative conceptions you may have and which you will be able to use with other colleagues, your Higher class as a baseline of what they understand from previous years and later on with N5 pupils.