Lecture 3

Invariants/Strong Induction

Prep

* Candy
* 8-puzzle
* 16 Bricks
* Top 10 List
* TA’s Game Strategy (have them snack on candy after 1st round and think about it a bit) & Names
* 15 Puzzle
* 15 Puzzle on PS2
* Write larger on board

Take

* Candy x 2
* 15-puzzle
* 8 puzzle letters & bd.
* 16 Bricks
* Top 10 list – for class
* 5 pocket protectors

Handout

* Top 10 List

Reminders

* PS2 due Monday 7:30 pm\*\* – list collaborators and sources.
  + Remember to write
* 10 pt bonus for going to office hours
  + Try to see your TA but you can see anyone, including me

From looking at the problem sets, it’s clear that many of you know how to do a proof. It’s also clear that many of you are still learning. Sometimes it takes a little while to get in synch with what we expect from you when writing a proof & your TA will be working with you on this over next few weeks.

Generally speaking, it’s best when a proof has 7 characteristics.

**Good proofs are:**

* **Write in line**
* **correct –** obviously important
* **complete –** all key details there
* **clear** – well explained & not a messy collection of arrows
* **brief** – not writing an essay, just the key steps – don’t want to crush the reader with details.
* **elegant** – this is the mathematicians’ notion of “beautiful”. For example, an artist might say “wow, that’s a beautiful painting!” The highest compliment you can get from a mathematician is for them to say your proof is “very elegant”. Of course, lots of judgment here.
* **well organized** – use lemmas, just like subroutines
* **in order** – sometimes you’ll see a proof where all the pieces are there but in a haphazard or confusing order. One especially common mistake is putting the steps in reverse order. For example, if you want to prove A=B, don’t start there and work backwards until get 1 = 1 or some other truth:

**Ok if**

**A = B ⇑**

**.**

**. ⇑**

**.**

**1 = 1**

Sometimes, taught in high school, but poor way to go. Often ok .. not always. Ok if **⇑** between each start but often this is not checked & sometimes not true. And very easy to make an error.

Good proofs are very much like good code! In fact, one of the reasons that we care so much about proofs in computer science is to be able to prove that **programs** do what they are supposed to do! This is important because our lives are increasingly dependent on software, and if the software doesn’t work right, very bad things can happen.

There are some famous examples where important software did not do what it was supposed to do:

The Airbus A300 was one of the first passenger jets to be run by software. Major advance -- only problem was that software glitch on one of the first A300s caused rear door to open a few minutes before landing and plane crashed.

The Therac 25 was a famous computer-driven machine for providing radiation therapy to cancer patients. It was famous because the software sometimes got into a race condition which ended up killing patients with massive overdoses of radiation.

Are any of you old enough to remember the 2000 election? Most too young but maybe you heard about it later. Al Gore ran against George Bush. Very close election. Most think Gore actually won, but there was some funny business with chads in florida. There were also bugs in the computerized voting systems. In fact, poor al gore got negative 16k votes in one county. **☺**

Several years ago, a single faulty command to a computer system used by United and AA grounded the entire fleets of both airlines for most of a day.

There are many more examples.

We worry a lot about correct software at **Akamai**. Akamai is an internet content delivery company that Danny Lewin and I started with a dozen MIT undergraduates in late 90’s. **Explain Akamai:** If our software doesn’t work right, we could take down 250K most popular sites on web, including Facebook, 97 of the top 100 commerce sites like Ebay and Amazon, and most banks.

To make the importance of good proofs a little more personal, think about the fact that someday, we’ll all be at mercy of critical computer systems designed by some of you. So I want you to look at the person sitting next to you – someday your life could depend on software they wrote doing what it’s supposed to do! That’s a little scary, right? That is why we are very motivated to help you develop the ability to formulate rock-solid logical arguments – so when you write code, it won’t nuke your classmate or cause a problem with their flight.

Unfortunately, doing good proofs is hard. Even the world’s best mathematicians mess them up from time to time. In fact, it is estimated that 1/3 of all math papers contain errors.

The trouble often arises because we get lazy and don’t write down all the steps, details or cases. In order to save time and energy, we write up less formal sketches or skip proofs or intermediate steps altogether. This can be ok, but it dramatically increases chances for error.

There are some famous examples:

* Gauss’s 1799 Ph. D. theses is usually referred to as being the first rigorous proof of the Fundamental Theorem of Algebra (every polynomial has a zero over the complex numbers). Guass is very famous in math. But his thesis contains quotes like –

“If a branch of an algebraic curve enters a bounded region, it must necessarily leave it again. .. Nobody, to my knowledge, has ever doubted [this fact].” (This is a warning sign –buzzers should be going off in your brain when you hear something like this.) Gauss continues: “But if anybody desires it, then on another occasion I intend to give a demonstration which will leave no doubt.”  **Now if Gauss did this in 6.042- homework, we’d zap him. ☺**

Fields Medalist Steve Smale writes about this, calling it an “immense gap” in the proof that was not filled in until 1920, more than a hundred years later. Turns out Gauss could not give a proof after all!

* Remember the Poincare Conjecture from last week? The Colbertvideo. In 1900, Poincare claimed it was a simple fact. Four years later, he decided it was not so obvious & he demoted the claim to the status of a “conjecture”. Of course, this became the infamous The Poincare Conjecture – which took over 100 years to solve.

If you see yourself doing something like this, stop. There’s a good chance it will lead to an error or gap.

There are other ways that you can mess up a proof – to help you understand them, we’ve prepared a Top 10 list of proof techniques that you should not use.

**Hold up Handout**

**Top 10 List of Proof Techniques**

**that Should NOT be Used in 6.042**

10. **Proof by throwing in the kitchen sink:** The author writes down every theorem or result known to mankind. When questioned later, the author correctly observes that the proof contains all the key facts needed to actually prove the result. Very popular strategy on 6.042 exams. **Explain Cribsheets** – just copy down all the lemmas from the term and try to wangle some credit later.

9. **Proof by example:** The author gives only the case n = 2 and suggests that it contains most of the ideas of the general proof.

8. **Proof by vigorous handwaving:** One of my favorites – if I wave my hands enough, it must be true.

7. **Proof by cumbersome notation:** Best done with access to at least four alphabets and special symbols. ***Reader gets hopelessly confused. I once had grad student, now a famous professor at berkeley – we called him the encryptor – he could make even the simplest proof impossible to follow using this technique.***

6. **Proof by exhaustion:** This is a variant of proof by throwing in the kitchen sink and proof by cumbersome notation.

5. **Proof by omission:** Very common and a faculty favorite. Telltale signs are phrases like:

“The reader may easily supply the details,”

“The other 253 cases are analogous,” or

***“Trivial”*** *– I’m guilty of this one. You can probably find some copies of my old papers on line where I’ve done this in lieu of the proof – turned out that every once in a while it wasn’t so trivial after all…*

**4. Proof by picture:** *Saw one on Tuesday. Proof by picture is a* more convincing form of proof by example. Combines well with proof by omission.

**3. Proof by vehement assertion:** *Similar to proof by intimidation.* *Helps to raise your voice and start the proof by saying something like “Any moron knows that” – the goal is to intimidate the listener into submission. Doesn’t make it true, of course.*

2. **Proof by appeal to intuition:** Cloud-shaped drawings frequently help here. Can be seen on 6.042 exams when there was not time to include a complete proof by throwing in the kitchen sink.

1. **Proof by reference to eminent authority:** *This includes statements like “The professor said so,” or,* “I saw Fermat in the elevator and he said he had a proof …”

*I don’t think so. Aside from fact that he’s been dead for over 300 years – he was not too reliable in the first place.*

Does everyone know the story of Fermat’s last thm?

**Fermat’s Last Thm**

**∀n >2 ¬∃ x, y, z ∈N + xn +yn = zn**

i.e., for n > 2, no positive integer solutions to

**Relate to Pythagorean Thm with n=2**

In 1637, Fermat wrote that he had a proof of this result, but that the proof was too long to be put in margin of the book he was writing in. Fermat did supply the details of the alleged proof.

After 350 years and many attempts and 100s of pages of deductions, result was finally proved by Andrew Wiles. It took Wiles over 10 years to nonstop focus to do it. Maybe Fermat right about too long for margin! **☺**

Questions on proof technique?

We will pound a lot on this over the next few weeks.

Next we’re going to look at a class of puzzles that was very popular in the late 1800s.

In these puzzles, you have a grid with letters in each position of the grid except one and the goal is to get all the letters in alphabetical order by moving one letter at a time.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **E** | **F** |
| **H** | **G** |  |

**Problem: Find a sequence of moves**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **E** | **F** |
| **G** | **H** |  |

**to go from to**

**SAVE**

In other words, you are supposed to get the G & H in order without messing up order of any other letters. Just swap two letters.

**Legal move: slide a letter into adjacent blank square: up, down, left, or right.** No diagonal moves allowed.

**Example on White Board**

Everyone understand? Raise your hand if you have played a game like this.

Ok – anyone think they can do it in 2 minutes or less? Who is good at these sorts of puzzles under pressure? Come on down. Need a team of 3 volunteers to solve the puzzle. You can get help from the class. If you can do it within 2 minutes – give you some candy. Consolation prize if you can’t beat the clock. Probably harder with 3 people vs 1 because then you mess each other up.

**Get volunteers – show candy**

Good at puzzles? How confident are you?

No picking up letters - only legal moves

Can’t do it? Ok – didn’t win candy but I have some very fashionable plastic nerd-pride pocket protectors for you instead for trying. I’ll give you another chance to win candy later.

**Q**. I was a little cruel to these folks. Any idea why?

**A**. Puzzle can’t be solved at all, never mind in 2 minutes.

Can’t be done? Prove it to me.

Well – let’s see if we can prove that.

**THM: The puzzle is not solvable.** I.e., there is no sequence of legal moves to invert G & H and return all other letters to their original position. You can’t get there from here.

**SAVE**

To prove this we’ll make use of what is called an “invariant” – very powerful and commonly-used concept in computer science and very closely tied to induction. The goal will be to use the invariant to show that no matter what sequence of moves we make, we can never get to a state where the letters are in sorted order.

In general, if we want to show that a system can never reach some special state, it is sufficient to show that there is some property (called the invariant) that

1) holds for the initial position or state,

2) never changes as a consequence of a legal move, and

3) does not hold for the special state.

If you have this magic invariant property and it holds at the start & is maintained by every move, then it must hold for any reachable state. If invariant does not hold for some special state, then it must be that the special state can never occur – because you can’t get there by legal moves.

Let’s see if we can figure out such a property for our puzzle. To do that, we need to see what happens when we make a move.

2 kinds of moves: row & column.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **G** |  |
| **E** | **F** | **H** |

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** |  | **G** |
| **E** | **F** | **H** |

**Row move**

**Ex:**

**⇒**

Row moves don’t do a whole lot. The space moves left or right by one, but the relative order of the items does not change. Now to make that precise, we have to define what we mean by relative order. In this case, we will use what is called row-major order.

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

**Row-major order**

**Fact: A row move does not change the order of the items.**

**Explain why: group before unchanged and group after unchanged. Show overall order ABCDGEFH is unchanged.**

A little risky since sort of like proof by picture or proof by example, but we have given a fair argument here—not trying to overburden you in doing a proof. Might be safer to state as a lemma and then give a more detailed proof. Judgment call.

Questions so far?

OK, if we only had row moves, this would be a pretty silly puzzle since nothing really changes, so let’s look at what happens during a column move.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **F** |  |
| **H** | **E** | **G** |

**Column Move:**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **F** | **G** |
| **H** | **E** |  |

**Ex: ⇒**

Column moves are more interesting.

**Q:** For example, did the ordering change?

**A:** Yes

**Q:** How so? Which pairs changed order?

**A**: G changed order with two letters before:

G was after E and is now before E

G was after H and is now before H

So we inverted the relative order of G & E and the relative order of G & H.

Let’s try another column move to see what happens. We are looking for an invariant and a good way to find it is to look at examples of moves and see what happens.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** |  | **G** |
| **H** | **E** | **F** |

|  |  |  |
| --- | --- | --- |
| **A** |  | **C** |
| **D** | **B** | **G** |
| **H** | **E** | **F** |

**Ex:** **⇒**

B moves down in this case.

**Q:** What changes occurred in the ordering?

**A:** Swapped relative order of B & C and B & D.

**Q:** Any guess about Lemma we will prove for column move?

**A:** The moving letter changes order with two before or two after.

By the way, when we tell you to not do a “proof by example,” we don’t mean to not do examples. Often you need to do examples to figure out what is going on and also to figure out how to do the proof. And by doing examples in this case, it will lead us to hypothesize that:

**Lemma 1: A column move changes the relative order of precisely 2 pairs of items.**

Of course, we are not done yet—we need to prove the lemma.

**Proof: In a column move, we move a letter from cell i to a blank spot in cell i+ 3 or i-3.** Nothing else moves.

Let’s check that:

** 1 2 3**

All Column moves 3 apart

** 4 5 6**

**7 8 9**

**When a letter moves 3 positions, it changes relative order with the 2 letters in between.**

Questions?

**Q:** Anyone have an idea about how we can use these lemmas to show original puzzle can’t be solved?

**A:** Maybe we can analyze this puzzle by looking at the number of pairs of items that are out of order. A pair of items that is out of order is said to be inverted.

**Def:** A pair of letters **L1 and L2 are inverted if L1 precedesL2 in the alphabet but L1 appears after L2 in the puzzle order.**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **F** | **D** | **G** |
| **E** | **H** |  |

**Q:** What are the inverted pairs?

**A: (D, F), (E, F), (E, G)**

**EX:**

**Q:** How many inverted pairs are in the start state of the original puzzle?

**A:** 1, just G & H **A B C**

**D E F**

**H G**

**Q:** How many inverted pairs in desired end state?

**A:** 0 – all in order **A B C**

**D E F**

**G H**

So to solve the puzzle, we need to go from the start state with 1 pair inverted (G/H) to a state with no inversions.

**Q/A:** To show it’s not possible, we’ll look at how # inverted pairs can change with each move.

**Q:** How can the number of inverted pairs change w/ a move?

**A:** Up by 2, down by 2, or no change.

Let’s state as a Lemma and prove it.

**Lemma 2: During a legal move, the number of inverted pairs can only increase by 2, decrease by 2, or remain the same.**

**Proof: Row move: no changes (Fact)**

**Column move: 2 pairs change order (Lemma 1)**

**3 cases:**

**A. both were in order ⇒ Q/A: # inverted pairs up by 2**

**B. both were inverted ⇒ Q/A: # inverted pairs down by 2**

**C. one each ⇒ Q/A: # inverted pairs is same.**

So the number of inverted pairs can change during a legal move, but if it does change, it can only change by 2.

**Q:** What property of the number of inverted pairs can’t change during a move? Remember, we are looking for an invariant—which means something that does not change.

**A:** The parity of the number of inverted pairs can’t change. I.e., if you begin with an odd number of inverted pairs, you will still have an odd number of inverted pairs after a legal move. This is our invariant! And we will state it as a lemma.

**Lemma 3 (Invariant): during a move, the parity (even/odd) of the number of inverted pairs does not change.**

**Proof: adding or subtracting 2 from a number does not change its parity.**

Almost done now.

**Q:** If parity of # inverted pairs never changes, is it possible to go from a start state with 1 inversion to a final state with no inversions?

**A:** No

**Q:** Why not?

**A:** Parity of # inversions will always be odd – it can never change during a move.

This seems pretty obvious but it we are going to prove it since it is so fundamental in computer science.

**Lemma 4: In every configuration reachable from**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **C** |
| **D** | **E** | **F**  **the parity of the number of inverted pairs is odd.** |
| **H** | **G** |  |

**Invariant**

**Proof: by induction**

**P (n): After any sequence of n moves from**

**Q:** what do we check first?

**A: Base case: n = 0. # inverted pairs = 1 ⇒ parity odd.**

**Inductive step: For n ≥ 0, show P (n) ⇒ P (n + 1).**

**Assume P(n) is true for purposes of induction.**

**Consider any sequence of n + 1 moves M1, …, Mn+1.**

**By I.H.** (i.e., P(n)), **we know that parity after M1,…, Mn is odd. By Lemma 3, we know parity does not change during Mn+1 ⇒ parity after M1 , …, Mn+1 is odd ⇒ P(n+1).**

Now we can prove original Thm – that the puzzle is not solvable.

**Proof of Thm: The parity of the # inverted pairs in the desired end state is even ( 0 ). So by Lemma 4, the desired end state can’t be reached from the start state** which means that the puzzle is not solvable.

Questions?

**Q:** Could we get the letters in order if we allowed location of blank to be different?

**A:** No

**Q:** Why not?

**A:** Parity still even and no even parity state is reachable.

This kind of puzzle was originally invented by Noyes Chapman in 1874. Most popular version – called 15 puzzle – was 4 x 4 grid with 15 letters or numbers & one blank. Same goal: swap the last two letters.

**SHOW 15 – Puzzle**

Puzzle was enormously popular in 1880’s – a $1000 prize was offered to anyone who could find a way to swap last two letters. That would be worth quarter-million $ today – but, just like 8 puzzle, it turns out there is no solution. The proof you can’t solve 15 – puzzle is a little trickier than 3 X 3 version & we left it for homework. These Lemmas will be useful but you will need a slightly more complicated invariant.

Proofs using invariants are important in computer science – since there are many cases where you want to be able to prove that you don’t get into a bad state.

For example, suppose you are writing software for a nuclear reactor control system. You would like to be sure that whatever steps you perform – you don’t wind up in a meltdown state.

Or suppose you are designing software for radiation therapy machine - don’t want to ever be in a state where you fry the patient.Airbus etc.

Questions?

For the rest of today, we are going to talk about another variation of induction called strong induction. Very similar to ordinary induction, but sometimes easier to use when solving problems.

Like ordinary induction, strong induction can be expressed as an axiom.

**Strong Induction**

**Axiom: Let P(n) be any predicate. If P (0) is true and ∀n ≥ 0  then P(n) is true for all n ≥ 0.**

**Explain correspondence with dominos and why it makes sense – if/then formulation of dominos going down**

**Explain difference with ordinary induction in axiom**

Ordinary: P(n) implies P(n+1)

Strong: P(1) and P(2) and … and P(n) implies P(n+1)

Because of how implication works, this means that in ordinary induction, we can only assume P(n) to prove P(n+1). But with strong induction, we get to assume a lot more – we can assume P(0), P(1), … , P(n) are all true to prove P(n+1) is true.

Sometimes this makes the proof easier since we get to assume more. And you would think that you could prove stronger things as a result—in fact, that’s why it is called strong induction. But that turns out to not be true since a proof using strong induction can be converted to a proof using regular induction. So whatever you can do with one method, you can do with the other – it just is easier using one or the other depending on what you are trying to prove.

There is an interesting game where strong induction makes the analysis easier. In this game, you have stack of 8 blocks.

**Show Block Stack**

First step, split the stack of blocks any way you want (say 5 – 3). **Show with blocks** Your score for the step is the product of the sizes of the two resulting stacks.

**Unstacking Game Points**

**8**

1. **3 15**

In each subsequent step, you split another stack and your score for that step is computed the same way.

**5**

**4 1 4**

Keep on going until all bricks are in stacks of size 1. Score for game is sum of all your points.

Goal is to get max score for game.

Questions?

Just so it’s clear, we are going to play this game in class. We will pit some volunteers from the class against the TAs to see who is better. If volunteers from class can beat the teaching assistants then they win prize. Candy for everyone in class; otherwise give candy to teaching assistants.

**Get 3 volunteers**

Make some noise if you think 3 6.042 students can beat TA’s!

How many think the TA’s will win?

Students go first – alternate moves.

Class can help with advice.

**Do on 2 boards**

**Class TA’s**

**Strategy Points Strategy Points**

**8 8** (pathetic first move)

**7**

**1**

**6**

**1**

**5**

**1**

**4**

**1**

**3**

**1**

**1**

**1**

**2**

**1**

**6**

**7**

**4**

**5**

**2**

**3**

**28**

**1**

We know sum of 1+2+…+7 is 7x8/2 = 28 from last time.

Tie: Class started strong but finished weak! TA’s looked pessimal but they caught up at end.

Hmmmm tie – not sure whether to give candy to teaching assistants or class. Tell you what – I’ll give you another chance to beat 28.

* **Redo –**

Whoa! Another tie – what are the odds of that?!

They say you can’t beat 28 – that all strategies give 28. I wouldn’t do that to you … would I? Well yeah, I would. ☺

Ok, given that we have another tie, I guess I’ll have to put it to a vote to see who wins the candy.

Make some noise if you think I should give it to class?

Make some noise if you think I should give it to TAs?

Sorry guys. **☺**

**Give candy to class**

**Erase boards**

Ok, back to the game. This was strange. All 3 strategies got the same score.

**Q:** Any thoughts about a possible Thm here

**A:** All strategies produce the same score! 28 is the max and the min.

Let’s try to prove that all strategies give the same score – but for any # blocks.

**THM: All strategies for the n – block game produce the same score S(n).**

**SAVE**

**Ex: S(8) = 28**

**Pf:** What strategy?

**By strong induction**, of course -- remember to write this down

**Q:** What is the next step?

**A:** Identify I.H.

**I.H: P(n)**

**Q:** Next step

**A: Base Case: n=1 S(1) = 0** (no score **–** bricks already unstacked)

Q: Next Step

A: **Assume P(1), P(2), ... , P(n) to prove P(n+1).**

**Look at n+1 bricks n + 1**

First move: **1 ≤ k ≤ n**

**k n+1- k**

**score = k(n+1-k) + S(k) + S(n+1-k)**

**Q/A**

**by P(k) & P(n+1 - k)**

**Show ordinary induction works for TA strategy since we went down by 1, but we need strong induction for a general strategy.**

**Q:** What do we need to show to prove P(n+1), i.e. that all strategies give same score for n+1 bricks?

**A:** Score does not depend on k

Uh-oh – does not appear to be true – far from clear why.

Hmmm – seems like dead end

**Q:** What should you do when can’t get your proof by induction to work? Remember from last time?

**A:** Try to prove something harder.

**Q:** Any ideas for stronger IH? (Stronger used in diff context here – stronger IH, diff than strong induction).

**Q:** What do you need to know to show score does not depend on k?

**A:** Formula for value of S(n)

OK. Let’s see of we can figure out what S(n) should be. Since we think all strategies result in the same score, we could get the score by looking at the TA strategy, and see if we can figure it out from that.

**Q/A**

**A:** **S(n) = n-1 + n-2+ …….+3+2+1 =** 

From last class

Ok, so let’s plug this into our inductive hypothesis and see if it works.

**S(n) =** in Thm **Go back & modify Thm & proof**

Base case ****

**Score = k (n+1-k) + S(k) + S(n+1-k)**

**= kn + k – k2 +  + **

****

**= **

**= S(n+1)**

**So  True**

**The score is always S(n) =** 

Questions?

**Q:** Why did strong induction come in handy here?

**A:** needed to cover all recursive cases depending on first split of stack

(i.e. value of k).

Just assuming P(n) only good enough to cover TA’s strategy.

There is a proof by ordinary induction, but more complicated and less natural.

That’s it for today. Next week, we will apply this material to study number theory and cryptography.

**DO IF TIME**

To finish up today, let’s see an example of how not to use strong induction.

**Thm (Not!): All natural numbers are even**

**“Proof”: by strong induction**

**IH: P(n): n is even**

**Base Case: n = 0 is even ⇒ P(0) true**

**Induction Step: ∀ n ≥ 0 assume P(0), P(1) … , P(n) to prove P(n+1)**

**i.e. Assume 0, 1, ….., n are even**

**Examine n+1**

**P(n) ⇒ n is even**

**P(1) ⇒ 1 is even**

**⇒ n+1 is even** (sum of evens is even)

**⇒ P (n+1)**

**Q:** Where is the bug?

**Q/A** Indeed **P(5) follows from P(4) & P(1)**

**Bug: n=0 “Assume P(0), P(1), …. P(n), to prove P(1)”**

Cannot assume P(1) when n = 0. Can only assume P(0).

Bug – trouble with “…” Again!

Just like in proof that all horses are the same color.

Here is what we really did:

**P(0) is T**

**P(1) ⇒ P(2)**

All are true

**P(1) Λ P(2) ⇒ P(3)**

**.**

**.**

**.**

**∀n≥1**

**P(1) Λ … Λ P(n) ⇒ P(n+1)**

**Q:** What is missing?

**A: Missing P(0) ⇒ P(1) Key Domino**

What if we modified proof to start at n = 1 instead of n = 0.

**Q:** Why doesn’t that proof work?

No “…” problem anymore.

**A:** Base case fails!

Someone asked last time why you need the base case. Here is a great example of what goes wrong if you don’t check it.