Lecture Notes

In the previous lecture, we introduced two main tools for analyzing a procedure. The first tool, the state machine, models all possible behaviors of a procedure or system. This serves as a basis for formal proofs. Invariants, the second tool, are statements that are true for all reachable states. Useful theorems about the behavior of a system can often obtained by proving that the corresponding state machine has a certain invariant.

This lecture introduces a third tool for analysis of algorithms, derived variables, with a focus on applications to your love life.

1 Derived Variables

Definition A *derived variable* for state machine $\mathcal{M} = (Q, Q_0, \delta)$ is a function $f : Q \mapsto S$ where S is a set (usually $\mathcal{N}$).

Derived variables are also called *variant functions* and *potential functions.*

As an example, in the Die Hard problem from last lecture we could have introduced a derived variable for the amount of water in both buckets, by setting $f((a, b)) = a + b$. Similarly, in the robot problem, the position of the robot along the x-axis would be given by the derived variable $f((i, j)) = i$.

More generally, let $P(q)$ be any predicate. Then we can define a derived variable as follows.

$$f_P(q) = \begin{cases} 0 & if\ P(q)\ is\ false \\ 1 & if\ P(q)\ is\ true \end{cases}$$

This allows an invariant to be characterized in terms of a derived variable. In particular, the predicate $P(q)$ is an invariant if and only if $f_P(q) = 1$ for all reachable q.

The most interesting derived variables reveal something about the execution of a procedure. For example, suppose that a natural-valued derived variable always decreases.

Definition A derived variable $f : Q \mapsto \mathcal{N}$ is *decreasing* if for every execution $q_0, q_1, \ldots,$ we have $f(q_0) > f(q_1) > \ldots,$

For example, suppose a state machine starts in state q_0 and $f(q_0)$ is 100. During each step of an execution, the derived variable decreases in value— something like $100, 99, 96, 91, 72, 71, 65, \ldots$. Since the variable can never be negative, this implies that all executions are finite length; that is, the procedure terminates.

Theorem 1.1 If $f : Q \mapsto \mathcal{N}$ is a decreasing derived variable of a state machine, then the length of any execution starting at q_0 is at most $f(q_0)$.

In other words, if we know that *any* decreasing derived variable exists for a state machine with a single start state, we know that *every* execution terminates.

Proof. The proof is by contradiction. Suppose that there exists an execution $q_0, q_1, \ldots, q_t, \ldots$ where $t > f(q_0)$. Then $f(q_t) \leq f(q_0) - t < 0$, since the value of f decreased t times by at least 1 each time. This contradicts the assertion that f is natural-valued. ■

Even if there are many possible starting states, the length of executions can still often be bounded with the following corollary:

Corollary 1.2 The length of any execution of a state machine $\mathcal{M} = (Q, Q_0, \delta)$ is at most $\max_{q_0 \in Q_0} f(q_0)$

Of course, one can define other flavors of derived variable, such as increasing, non-decreasing, non-increasing, etc.

Definition A derived variable $f : Q \mapsto \mathcal{N}$ is *non-increasing* if for every execution $q_0, q_1, \ldots$, we have $f(q_0) \geq f(q_1) \geq \ldots$.

The existence of a non-increasing derived variable does not guarantee that every execution terminates. During an execution, the value of a non-increasing derived variable might decrease for a time, but then reach some constant and stay there. For example, $f(q) = 1$ is non-increasing derived variable for every state machine, but not all executions of all state machines terminate!

Here is another type of derived variable that will be useful later in this lecture.

Definition A derived variable $f : Q \mapsto \mathcal{N}$ is *non-skipping* if for every execution $q_0, q_1, \ldots$, we have $|f(q_i) - f(q_{i+1})| \leq 1$ for all $i \geq 0$.

Non-skipping derived variables are useful when we want to be sure that we have covered all the possibilities within some range. For example, if f is non-skipping, and there is an execution where f is 1 at some point and 100 at another point, then we know that in between the machine enters at least one state with each value $2, 3, \ldots, 99$. This idea is captured by the following theorem:

Theorem 1.3 (Non-Skipping Theorem) If f is non-skipping and $q_0, q_1, \ldots$ is an execution, then for all $0 \leq i \leq j$ and for all $r \in \{f(q_i), f(q_i)+1, \ldots, f(q_j)\}$, there exists a $t \in \{i, i+1, \ldots, j\}$ such that $f(q_t) = r$.

The proof is by induction on $j - i$, but omitted here.

2 The Mating Problem

Okay, frequent public reference to derived variables may not help your mating prospects. But they can help with the analysis!

2.1 The Problem

Suppose that there are N boys and N girls. Each boy ranks all of the girls according to his preference, and each girl ranks all of the boys. For example, Bob might like Alice most, Carol second, Hildegard third, etc. There are no ties in anyone's rankings; Bob can not like Carol and Hildegard equally. Furthermore, rankings are known at the start and stay fixed for all time.

The general goal is to pair off boys and girls so that everyone is happy; we'll be more precise in a moment. Every boy must paired with exactly one girl and vice-versa– no polygamy. For now we'll also not allow boys to be paired with boys or girls with girls. We're not making any social judgement here, its just that this makes the mathematical problem a lot harder!

If we want these pairings to last, then we want to avoid an unstable arrangement:

Definition A set of pairings is *unstable* if there exist a boy and a girl that both prefer each other to the people with whom they are paired.

For example, suppose that Bob and Carol are paired, and Ted and Alice are paired. The situation is shown in Figure 1. Unfortunately, Carol likes Ted more than Bob *and* Ted likes Carol more than Alice. Trouble! The initial set of pairings is unstable. Carol and Ted are going to start having late nights at the office. In this case, we say that Carol and Ted are a *rogue couple*. That is, they are not paired, but would prefer to leave their mates and form a pair.

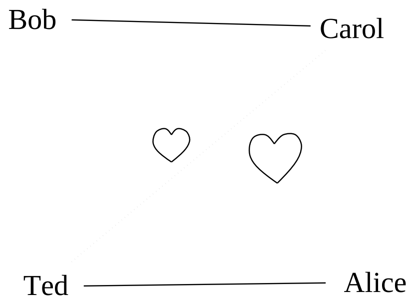

Figure 1: *Bob is paired with Carol, and Ted is paired with Alice. We will not formally define the term, "has the hots", but this reflects Ted's outlook toward Carol. More precisely, Ted prefers Carol his mate, and Carol prefers Ted to her mate. The initial pairing is therefore unstable, and Ted and Carol form a rogue couple.*

Definition A set of pairings is *stable* if there are no rogue couples or, equivalently, it the set of pairings is not unstable.

Now we can state the mating problem precisely: find a stable pairing of the boys and girls. It is not obvious that this goal is achievable! In fact, in the bisexual version of the problem, where boys can pair with boys and girls with girls, there may be no stable pairing!

However, for the "hetero" problem a stable pairing does always exist. Let's find a stable pairing in one possible situation, and then try to translate our method to a general algorithm. The table below shows the preferences of each girl and boy in decreasing order.

boys	*girls*
$1 : CBEAD$	$A : 35214$
$2 : ABECD$	$B : 52143$
$3 : DCBAE$	$C : 43512$
$4 : ACDBE$	$D : 12345$
$5 : ABDEC$	$E : 23415$

How about we try a greedy strategy? We take each boy in turn and pack him off with his favorite among the girls still available. This gives the following assignment.

$$
\begin{aligned}
1 &\to C \\
2 &\to A \\
3 &\to D \\
4 &\to B \\
5 &\to E
\end{aligned}
$$

To determine whether this pairing is stable, we have to check whether there are any rogue couples. Boys 1, 2, and 3 all got their top pick among the girls; none would even think of running off. Boy 4 may be a problem because he likes girl A better than his mate, but she ranks him dead last. However, boy 4 also likes girl C better than his mate, and she rates him above her own mate. Therefore, boy 4 and girl C form a rogue couple! The pairing is not stable. We could try to make ad hoc repairs, but we're really trying to develop a general strategy.

Another approach would be to use induction. Suppose we pair boy 1 with his favorite girl, C, show that these two will not join a rogue couple, and then solve the remaining problem by induction. Clearly boy 1 will never leave his top pick, girl C. But girl C might very well dump him– she might even rate him last!

This turns out to be a tricky problem. The best approach is to use the dating protocol that was popular back in the 50's.

2.2 The Mating Algorithm

The mating ritual takes place over several days. The following events happen each day:

Morning: Each girl stands on her balcony. Each boy stands under the balcony of his favorite girl among those he has not yet ruled out and serenades her. If a boy has been rejected by every girl, he stays home and does his 6.042 homework.

Afternoon: Girls who have at least one suitor say to their favorite, "Maybe... come back tomorrow." To the others, they say, "No. I will never marry you! Take a hike!"

Evening: Any boy who hears a "no" crosses that girl off his list, ruling her out.

Termination condition: When every girl has at most one suitor, each girl marries her suitor, if there is one.

There are a number of facts about this algorithm that we would like to prove:

- The algorithm terminates.
- The algorithm terminates quickly.
- The resulting marriages are stable.
- Everybody ends up married.

Furthermore, we would like to know if the algorithm is fair. Do both boys and girls end up equally happy?

2.3 Termination

We can prove that the Mating Algorithm terminates by modeling the procedure with a state machine and then showing that there exists a decreasing derived variable.

A state is defined by the set of names crossed off on the boys' lists. A natural candidate for the decreasing derived variable is then the total number of uncrossed names over all lists. Initially, there are N uncrossed names on each of the N boys' lists for a total of N^2. Then, while the algorithm is running, there are two possibilities to consider. First, if some girl has more than one suitor, then all but one of these will cross her off that evening, and so the derived variable will decrease. Otherwise, if no girl has more than one suitor, then the algorithm terminates. These observations are summed up in the following theorem and proof.

Theorem 2.1 The Mating Algorithm always terminates within N^2 days.

Proof. We will prove that the Mating Algorithm terminates by modeling the procedure with a state machine and then showing that there exists a decreasing derived variable $f(q)$.

Define a state machine $\mathcal{M} = (Q, Q_0, \delta)$ as follows. Let Q contain one state for each possible way of crossing off girls' names on the boys' lists. Let Q_0 be the state with no names crossed off on any list. For each state $q \in Q$ that does not cause the Mating Algorithm to terminate, compute q' with the algorithm, and add the transition $q \to q'$ to δ.

Let $f(q)$ be the number of uncrossed girl's names in state q. Initially, every boy's list has N uncrossed girl's names and so $f(q_0) = N^2$. To prove that $f(q)$ is a decreasing derived variable, we must show that for every execution $q_0, q_1, \ldots$, we have $f(q_0) > f(q_1) > \ldots$. For this purpose, it is sufficient to show that $f(q) > f(q')$ for every transition $(q, q') \in \delta$.

In every transition (q, q'), some girl is serenaded by more than one boy. All but one of these boys crosses off the girl that evening. This means that $f(q)$ decreases by at least 1.

Since there exists a decreasing derived variable $f(q)$ and $f(q_0) = N^2$, Theorem 1.1 implies that the algorithm terminates within N^2 steps. ■

Notice that while the state machine used in the proof is by no means trivial, it is nevertheless deterministic. Given the boys' current lists and the boys' and girls' rankings, the lists for the next day are completely determined.

2.4 They All Live Happily Every After...

We still have to prove that the Mating Algorithm leaves everyone in a stable marriage. Two additional derived variables will be useful:

- Let $f_i(q)$ be the original rank that boy i assigned to the girl he is serenading in state q. If the boy is serenading no one, the value of the variable is $N + 1$.
- Let $g_i(q)$ be the original rank that girl i assigned to the boy to whom she says "maybe" in state q. If she has no suitors in state q, then the value of the variable is $N + 1$.

Here are some useful facts about the derived variables $f_i(q)$ and $g_i(q)$ that follow immediately from the algorithm.

Facts

1. *f_i is a non-decreasing derived variable. As a boy crosses off names, the rank of the girl he serenades increases; that is, he is forced to pursue girls that he desires less.*
2. *f_i is non-skipping. Each boy crosses off at most one name each day.*
3. *$f_i(q_0) = 1$. Initially, each boy serenades his favorite gal.*
4. *The preceding two facts together with the Non-Skipping Theorem imply that if boy i never marries, then at least he serenaded every girl. ("Better to have loved and lost...")*
5. *Similarly, if boy i does marry, then he courted every girl that he ranked higher than his mate.*
6. *g_i is a non-increasing derived variable. The girl keeps stringing along her top choice by saying "maybe" (holding g_i constant) until someone better comes along (decreasing g_i) or until the algorithm terminates.*

We can now piece these facts together to prove two theorems that say everyone lives happily ever after.

Theorem 2.2 Everyone is married by the Mating Algorithm.

Proof. The proof is by contradiction.

Assume for the purpose of contradiction that some boy B_i is not married. Fact 4 implies that B_i visited every girl. This implies that every girl was courted at least once. This means that every g_i was less than $N + 1$ at some point. This, together with Fact 6 implies that every g_i is less than $N + 1$ at termination, and so every girl is married. This implies that every boy is also married, and contradicts the assumption that B_i is not married. ∎

Theorem 2.3 The Mating Algorithm produces stable marriages.

Proof. The proof is by contradiction.

Assume for the purpose of contradiction that there is a rogue couple $B - G$. Let G' be the girl that married B, and let B' be the boy who married G.

Since $B - G$ is a rogue couple, B must like G better than G'. This implies that B visited G first. However, she must have said "no" to him. This, together with Fact 6, implies that G found someone she likes even better than B. In particular, this implies that G likes B' better than B. But this contradicts the assumption that $B - G$ is a rogue couple. ∎

2.5 ...And the Boys Live Especially Happily

Who is favored by the Mating Algorithm, the boys or the girls? The girls seem to have all the power; they stand on their balconies choosing the finest among their suitors and spurning the rest. If you're a girl, this probably sounds pretty good!

We can study this problem formally. First, we need some definitions.

Definition The *realm of possibility* for a person P (boy or girl) is the set of people with whom P is paired in some stable pairing.

This captures the idea that Ben Bitdiddle is just never going to pair up with Christie Brinkley; she is not in his realm of possibility. No matter how the cookie crumbles, there will always be some guy that she likes better than Ben and that likes her more than his own mate. No pairing of Ben and Christie can be stable.

Note that since the Mating Algorithm always produces one stable pairing, the realm of possibility for a person– even Ben– is never the empty set.

Definition A person's *optimal* mate is her or her favorite from the realm of possibility.

Definition A person's *pessimal* mate is her or her least favorite from the realm of possibility.

Here is the shocking truth about the Mating Algorithm:

Theorem 2.4 The Mating Algorithm pairs every boy with his optimal mate and every girl with her pessimal mate.

Proof. The proof is in two parts. First, we show that every boy is paired with his optimal mate. The proof is by contradiction.

Assume for the purpose of contradiction that some boy does not get his optimal girl. Look at the first day that a boy is rejected by his optimal girl. (We are implicitly using the well-ordering principle here; induction in disguise!) Let B be one of these rejected boys, and let G be his optimal girl.

Girl G must have rejected B in favor of some boy B' that she likes better. Since this is the first day that a boy was rejected by his optimal girl, B' has not yet been rejected by his own optimal mate. This means that B' likes G more than any other mate in his realm of possibility. (Of course, she may not be in his realm of possibility; she may dump him later.)

Consider a stable pairing where B marries G and B' marries someone else. Such a stable pairing must exist, since G is optimal for B and therefore is in his realm of possibility. Now a problem arises, because B' and G form a rogue couple; G likes B' more than her mate, B, and B' likes G more any other mate in his realm of possibility. This contradicts the assertion that the pairing was stable.

Now for the second part of the proof: showing that every girl is paired with her pessimal mate. Again, the proof is by contradiction.

Suppose for the purpose of contradiction that there exists a stable pairing $\mathcal{M}$ where there is a girl G who fares worse than in the Mating Algorithm. Let B be her mate in the Mating Algorithm. By the preceding argument, G is optimal for B. Let B' be her mate in $\mathcal{M}$, a boy that she likes even less than B. Then B and G form a rogue couple in $\mathcal{M}$; B prefers G to his mate, because she is optimal for him, and G prefers B to her mate, B', by assumption. This contradicts the assertion that $\mathcal{M}$ was a stable pairing. ∎