# 6 State Machines

State machines are a simple, abstract model of step-by-step processes. Since computer programs can be understood as defining step-by-step computational processes, it’s not surprising that state machines come up regularly in computer science. They also come up in many other settings such as designing digital circuits and modeling probabilistic processes. This section introduces *Floyd’s Invariant Principle* which is a version of induction tailored specifically for proving properties of state machines.

One of the most important uses of induction in computer science involves proving one or more desirable properties continues to hold at every step in a process. A property that is preserved through a series of operations or steps is known as a *preserved invariant*.

Examples of desirable invariants include properties such as the altitude of a plane remaining above 1,000 feet unless the wingflaps are deployed, or the temperature of a nuclear reactor remaining below the meltdown threshold.

## 6.1 States and Transitions

Formally, a *state machine* is nothing more than a binary relation on a set, except that the elements of the set are called “states,” the relation is called the transition relation, and an arrow in the graph of the transition relation is called a *transition*. A transition from state $q$ to state $r$ will be written $q \longrightarrow r$. The transition relation is also called the *state graph* of the machine. A state machine also comes equipped with a designated *start state*.

A simple example is a bounded counter, which counts from 0 to 99 and overflows at 100. This state machine is pictured in Figure 6.1, with states pictured as circles, transitions by arrows, and with start state 0 indicated by the double circle. To be

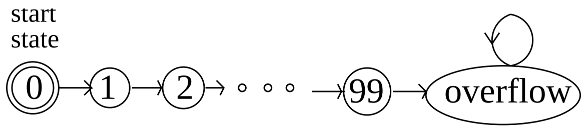


**Figure 6.1** *State transitions for the 99-bounded counter.*

precise, what the picture tells us is that this bounded counter machine has

$$\begin{aligned} \text{states} &::= \{0, 1, \ldots, 99, \text{overflow}\}, \\ \text{start state} &::= 0, \\ \text{transitions} &::= \{n \longrightarrow n + 1 \mid 0 \leq n < 99\} \\ &\qquad \cup \{99 \longrightarrow \text{overflow}, \text{overflow} \longrightarrow \text{overflow}\}. \end{aligned}$$

This machine isn't much use once it overflows, since it has no way to get out of its overflow state.

State machines for digital circuits and string pattern matching algorithms, for instance, usually have only a finite number of states. Machines that model continuing computations typically have an infinite number of states. For example, instead of the 99-bounded counter, we could easily define an "unbounded" counter that just keeps counting up without overflowing. The unbounded counter has an infinite state set, the nonnegative integers, which makes its state diagram harder to draw.

State machines are often defined with labels on states and/or transitions to indicate such things as input or output values, costs, capacities, or probabilities. Our state machines don't include any such labels because they aren't needed for our purposes. We do name states, as in Figure 6.1, so we can talk about them, but the names aren't part of the state machine.

## 6.2 The Invariant Principle

### 6.2.1 A Diagonally-Moving Robot

Suppose we have a robot that starts at the origin and moves on an infinite 2-dimensional integer grid. The *state* of the robot at any time can be specified by the integer coordinates $(x, y)$ of the robot's current position. So the *start state* is $(0, 0)$. At each step, the robot may move to a diagonally adjacent grid point, as illustrated in Figure 6.2.

To be precise, the robot's transitions are:

$$\{(m, n) \longrightarrow (m \pm 1, n \pm 1) \mid m, n \in \mathbb{Z}\}.$$

For example, after the first step, the robot could be in states $(1, 1)$, $(1, -1)$, $(-1, 1)$ or $(-1, -1)$. After two steps, there are 9 possible states for the robot, including $(0, 0)$. The question is, can the robot ever reach position $(1, 0)$?

If you play around with the robot a bit, you'll probably notice that the robot can only reach positions $(m, n)$ for which $m + n$ is even, which of course means that it

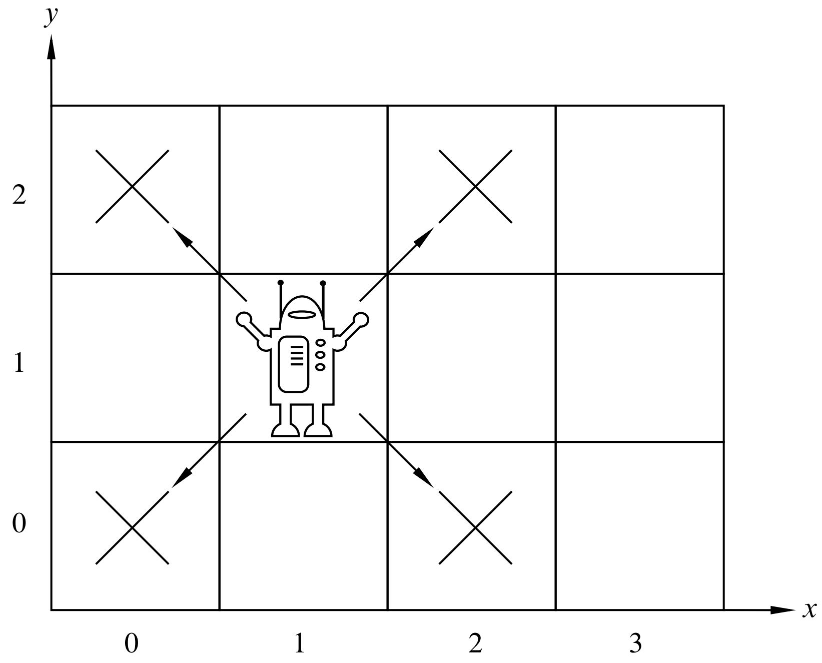


**Figure 6.2** *The Diagonally Moving Robot.*

can't reach $(1,0)$. This follows because the evenness of the sum of the coordinates is a property that is *preserved* by transitions. This is an example of a *preserved invariant*.

This once, let's go through this preserved invariant argument, carefully highlighting where induction comes in. Specifically, define the even-sum property of states to be:

$$\text{Even-sum}((m,n)) ::= [m+n \text{ is even}].$$

**Lemma 6.2.1.** *For any transition $q \longrightarrow r$ of the diagonally-moving robot, if Even-sum($q$), then Even-sum($r$).*

This lemma follows immediately from the definition of the robot's transitions: $(m,n) \longrightarrow (m \pm 1, n \pm 1)$. After a transition, the sum of coordinates changes by $(\pm 1) + (\pm 1)$, that is, by 0, 2, or -2. Of course, adding 0, 2 or -2 to an even number gives an even number. So by a trivial induction on the number of transitions, we can prove:

**Theorem 6.2.2.** *The sum of the coordinates of any state reachable by the diagonally-moving robot is even.*

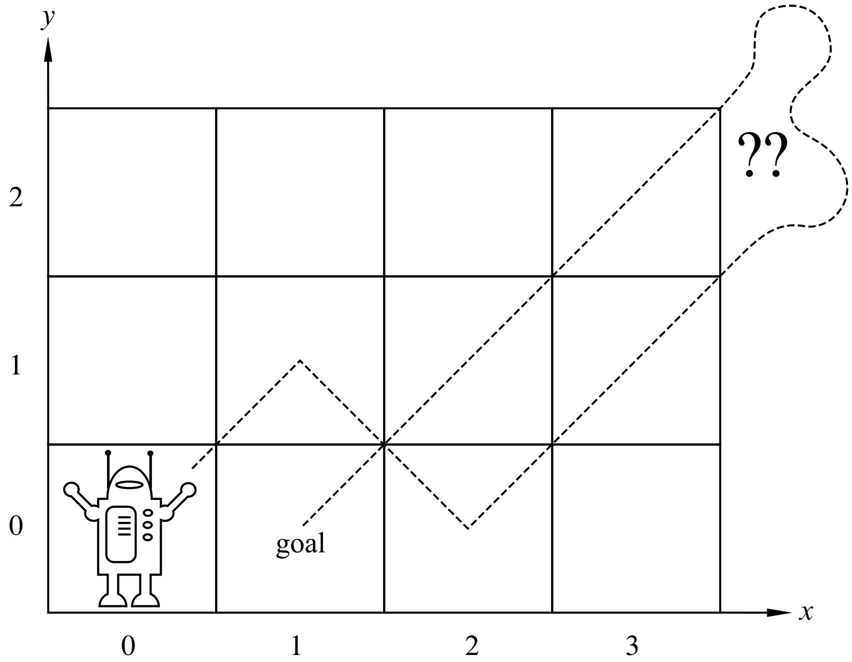


**Figure 6.3** *Can the Robot get to* $(1, 0)$*?*

*Proof.* The proof is induction on the number of transitions the robot has made. The induction hypothesis is

$$P(n) ::= \text{if } q \text{ is a state reachable in } n \text{ transitions, then Even-sum}(q).$$

**Base case**: $P(0)$ is true since the only state reachable in 0 transitions is the start state $(0,0)$, and $0 + 0$ is even.

**Inductive step**: Assume that $P(n)$ is true, and let $r$ be any state reachable in $n + 1$ transitions. We need to prove that Even-sum($r$) holds.

Since $r$ is reachable in $n + 1$ transitions, there must be a state $q$ reachable in $n$ transitions such that $q \longrightarrow r$. Since $P(n)$ is assumed to be true, Even-sum($q$) holds, and so by Lemma 6.2.1, Even-sum($r$) also holds. This proves that $P(n)$ IMPLIES $P(n + 1)$ as required, completing the proof of the inductive step.

We conclude by induction that for all $n \geq 0$, if $q$ is reachable in $n$ transitions, then Even-sum($q$). This implies that every reachable state has the Even-sum property. ■

**Corollary 6.2.3.** *The robot can never reach position* $(1,0)$.

*Proof.* By Theorem 6.2.2, we know the robot can only reach positions with coordinates that sum to an even number, and thus it cannot reach position $(1,0)$. ■

### 6.2.2 Statement of the Invariant Principle

Using the Even-sum invariant to understand the diagonally-moving robot is a simple example of a basic proof method called The Invariant Principle. The Principle summarizes how induction on the number of steps to reach a state applies to invariants.

A state machine *execution* describes a possible sequence of steps a machine might take.

**Definition 6.2.4.** An *execution* of the state machine is a (possibly infinite) sequence of states with the property that

- it begins with the start state, and
- if $q$ and $r$ are consecutive states in the sequence, then $q \longrightarrow r$.

A state is called *reachable* if it appears in some execution.

**Definition 6.2.5.** A *preserved invariant* of a state machine is a predicate $P$ on states, such that whenever $P(q)$ is true of a state $q$ and $q \longrightarrow r$ for some state $r$ then $P(r)$ holds.

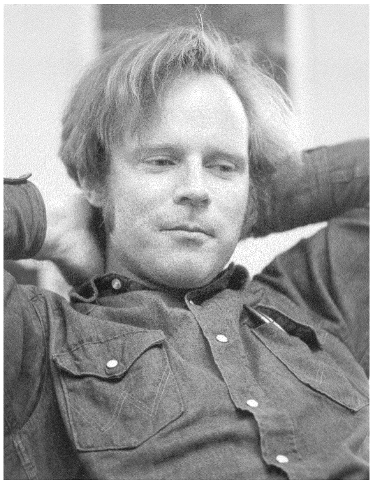
**The Invariant Principle**

If a preserved invariant of a state machine is true for the start state,
then it is true for all reachable states.

The Invariant Principle is nothing more than the Induction Principle reformulated in a convenient form for state machines. Showing that a predicate is true in the start state is the base case of the induction, and showing that a predicate is a preserved invariant corresponds to the inductive step.[1]

[1] Preserved invariants are commonly just called “invariants” in the literature on program correctness, but we decided to throw in the extra adjective to avoid confusion with other definitions. For example, other texts (as well as another subject at MIT) use “invariant” to mean “predicate true of all reachable states.” Let’s call this definition “invariant-2.” Now invariant-2 seems like a reasonable definition, since unreachable states by definition don’t matter, and all we want to show is that a desired property is invariant-2. But this confuses the *objective* of demonstrating that a property is invariant-2 with the *method* of finding a *preserved* invariant—which is preserved even at unreachable states—to *show* that it is invariant-2.

**Robert W. Floyd**



The Invariant Principle was formulated by Robert W. Floyd at Carnegie Tech in 1967. (Carnegie Tech was renamed Carnegie-Mellon University the following year.) Floyd was already famous for work on the formal grammars that transformed the field of programming language parsing; that was how he got to be a professor even though he never got a Ph.D. (He had been admitted to a PhD program as a teenage prodigy, but flunked out and never went back.)

In that same year, Albert R. Meyer was appointed Assistant Professor in the Carnegie Tech Computer Science Department, where he first met Floyd. Floyd and Meyer were the only theoreticians in the department, and they were both delighted to talk about their shared interests. After just a few conversations, Floyd's new junior colleague decided that Floyd was the smartest person he had ever met.

Naturally, one of the first things Floyd wanted to tell Meyer about was his new, as yet unpublished, Invariant Principle. Floyd explained the result to Meyer, and Meyer wondered (privately) how someone as brilliant as Floyd could be excited by such a trivial observation. Floyd had to show Meyer a bunch of examples before Meyer understood Floyd's excitement —not at the truth of the utterly obvious Invariant Principle, but rather at the insight that such a simple method could be so widely and easily applied in verifying programs.

Floyd left for Stanford the following year. He won the Turing award—the "Nobel prize" of computer science—in the late 1970's, in recognition of his work on grammars and on the foundations of program verification. He remained at Stanford from 1968 until his death in September, 2001. You can learn more about Floyd's life and work by reading the eulogy at

http://oldwww.acm.org/pubs/membernet/stories/floyd.pdf

written by his closest colleague, Don Knuth.

### 6.2.3 The Die Hard Example

The movie *Die Hard 3: With a Vengeance* includes an amusing example of a state machine. The lead characters played by Samuel L. Jackson and Bruce Willis have to disarm a bomb planted by the diabolical Simon Gruber:

**Simon:** On the fountain, there should be 2 jugs, do you see them? A 5-gallon and a 3-gallon. Fill one of the jugs with exactly 4 gallons of water and place it on the scale and the timer will stop. You must be precise; one ounce more or less will result in detonation. If you're still alive in 5 minutes, we'll speak.

**Bruce:** Wait, wait a second. I don't get it. Do you get it?

**Samuel:** No.

**Bruce:** Get the jugs. Obviously, we can't fill the 3-gallon jug with 4 gallons of water.

**Samuel:** Obviously.

**Bruce:** All right. I know, here we go. We fill the 3-gallon jug exactly to the top, right?

**Samuel:** Uh-huh.

**Bruce:** Okay, now we pour this 3 gallons into the 5-gallon jug, giving us exactly 3 gallons in the 5-gallon jug, right?

**Samuel:** Right, then what?

**Bruce:** All right. We take the 3-gallon jug and fill it a third of the way...

**Samuel:** No! He said, "Be precise." Exactly 4 gallons.

**Bruce:** Sh - -. Every cop within 50 miles is running his a - - off and I'm out here playing kids games in the park.

**Samuel:** Hey, you want to focus on the problem at hand?

Fortunately, they find a solution in the nick of time. You can work out how.

#### The Die Hard 3 State Machine

The jug-filling scenario can be modeled with a state machine that keeps track of the amount $b$ of water in the big jug, and the amount $l$ in the little jug. With the 3 and 5 gallon water jugs, the states formally will be pairs $(b, l)$ of real numbers such that $0 \leq b \leq 5, 0 \leq l \leq 3$. (We can prove that the reachable values of $b$ and $l$ will be nonnegative integers, but we won't assume this.) The start state is $(0, 0)$, since both jugs start empty.

Since the amount of water in the jug must be known exactly, we will only consider moves in which a jug gets completely filled or completely emptied. There are several kinds of transitions:

1. Fill the little jug: $(b, l) \longrightarrow (b, 3)$ for $l < 3$.
2. Fill the big jug: $(b, l) \longrightarrow (5, l)$ for $b < 5$.
3. Empty the little jug: $(b, l) \longrightarrow (b, 0)$ for $l > 0$.
4. Empty the big jug: $(b, l) \longrightarrow (0, l)$ for $b > 0$.
5. Pour from the little jug into the big jug: for $l > 0$,
$$(b, l) \longrightarrow \begin{cases} (b + l, 0) & \text{if } b + l \leq 5, \\ (5, l - (5 - b)) & \text{otherwise.} \end{cases}$$
6. Pour from big jug into little jug: for $b > 0$,
$$(b, l) \longrightarrow \begin{cases} (0, b + l) & \text{if } b + l \leq 3, \\ (b - (3 - l), 3) & \text{otherwise.} \end{cases}$$

Note that in contrast to the 99-counter state machine, there is more than one possible transition out of states in the Die Hard machine. Machines like the 99-counter with at most one transition out of each state are called *deterministic*. The Die Hard machine is *nondeterministic* because some states have transitions to several different states.

The Die Hard 3 bomb gets disarmed successfully because the state (4,3) is reachable.

### Die Hard Permanently

The *Die Hard* series is getting tired, so we propose a final *Die Hard Permanently*. Here, Simon's brother returns to avenge him, posing the same challenge, but with the 5 gallon jug replaced by a 9 gallon one. The state machine has the same specification as the Die Hard 3 version, except all occurrences of "5" are replaced by "9."

Now, reaching any state of the form $(4, l)$ is impossible. We prove this using the Invariant Principle. Specifically, we define the preserved invariant predicate $P((b, l))$ to be that $b$ and $l$ are nonnegative integer multiples of 3.

To prove that $P$ is a preserved invariant of Die-Hard-Once-and-For-All machine, we assume $P(q)$ holds for some state $q ::= (b, l)$ and that $q \longrightarrow r$. We have to show that $P(r)$ holds. The proof divides into cases, according to which transition rule is used.

One case is a "fill the little jug" transition. This means $r = (b, 3)$. But $P(q)$ implies that $b$ is an integer multiple of 3, and of course 3 is an integer multiple of 3, so $P(r)$ still holds.

Another case is a "pour from big jug into little jug" transition. For the subcase when there isn't enough room in the little jug to hold all the water, that is, when $b + l > 3$, we have $r = (b - (3 - l), 3)$. But $P(q)$ implies that $b$ and $l$ are integer multiples of 3, which means $b - (3 - l)$ is too, so in this case too, $P(r)$ holds.

We won't bother to crank out the remaining cases, which can all be checked just as easily. Now by the Invariant Principle, we conclude that every reachable state satisifies $P$. But since no state of the form $(4, l)$ satisifies $P$, we have proved rigorously that Bruce dies once and for all!

By the way, notice that the state (1,0), which satisfies $\text{NOT}(P)$, has a transition to (0,0), which satisfies $P$. So the negation of a preserved invariant may not be a preserved invariant.

## 6.3 Partial Correctness & Termination

Floyd distinguished two required properties to verify a program. The first property is called *partial correctness*; this is the property that the final results, if any, of the process must satisfy system requirements.

You might suppose that if a result was only partially correct, then it might also be partially incorrect, but that's not what Floyd meant. The word "partial" comes from viewing a process that might not terminate as computing a *partial relation*. Partial correctness means that *when there is a result*, it is correct, but the process might not always produce a result, perhaps because it gets stuck in a loop.

The second correctness property, called *termination*, is that the process does always produce some final value.

Partial correctness can commonly be proved using the Invariant Principle. Termination can commonly be proved using the Well Ordering Principle. We'll illustrate this by verifying a Fast Exponentiation procedure.

### 6.3.1 Fast Exponentiation

#### Exponentiating

The most straightforward way to compute the $b$th power of a number $a$ is to multiply $a$ by itself $b-1$ times. But the solution can be found in considerably fewer multiplications by using a technique called *Fast Exponentiation*. The register machine program below defines the fast exponentiation algorithm. The letters $x, y, z, r$ denote registers that hold numbers. An *assignment statement* has the form “$z := a$” and has the effect of setting the number in register $z$ to be the number $a$.

**A Fast Exponentiation Program**

Given inputs $a \in \mathbb{R}, b \in \mathbb{N}$, initialize registers $x, y, z$ to $a, 1, b$ respectively, and repeat the following sequence of steps until termination:

- if $z = 0$ **return** $y$ and terminate
- $r :=$ remainder$(z, 2)$
- $z :=$ quotient$(z, 2)$
- if $r = 1$, then $y := xy$
- $x := x^2$

We claim this program always terminates and leaves $y = a^b$.

To begin, we’ll model the behavior of the program with a state machine:

1. states $::= \mathbb{R} \times \mathbb{R} \times \mathbb{N}$,
2. start state $::= (a, 1, b)$,
3. transitions are defined by the rule

$$(x, y, z) \longrightarrow \begin{cases} (x^2, y, \text{quotient}(z, 2)) & \text{if } z \text{ is nonzero and even,} \\ (x^2, xy, \text{quotient}(z, 2)) & \text{if } z \text{ is nonzero and odd.} \end{cases}$$

The preserved invariant $P((x, y, z))$ will be

$$z \in \mathbb{N} \text{ AND } yx^z = a^b. \tag{6.1}$$

To prove that $P$ is preserved, assume $P((x, y, z))$ holds and that $(x, y, z) \longrightarrow (x_t, y_t, z_t)$. We must prove that $P((x_t, y_t, z_t))$ holds, that is,

$$z_t \in \mathbb{N} \text{ AND } y_t x_t^{z_t} = a^b. \tag{6.2}$$

Since there is a transition from $(x, y, z)$, we have $z \neq 0$, and since $z \in \mathbb{N}$ by (6.1), we can consider just two cases:

If $z$ is even, then we have that $x_t = x^2, y_t = y, z_t = z/2$. Therefore, $z_t \in \mathbb{N}$ and

$$\begin{aligned} y_t x_t^{z_t} &= y(x^2)^{z/2} \\ &= yx^{2 \cdot z/2} \\ &= yx^z \\ &= a^b && \text{(by (6.1))} \end{aligned}$$

If $z$ is odd, then we have that $x_t = x^2, y_t = xy, z_t = (z-1)/2$. Therefore, $z_t \in \mathbb{N}$ and

$$\begin{aligned} y_t x_t^{z_t} &= xy(x^2)^{(z-1)/2} \\ &= yx^{1+2 \cdot (z-1)/2} \\ &= yx^{1+(z-1)} \\ &= yx^z \\ &= a^b && \text{(by (6.1))} \end{aligned}$$

So in both cases, (6.2) holds, proving that $P$ is a preserved invariant.

Now it’s easy to prove partial correctness: if the Fast Exponentiation program terminates, it does so with $a^b$ in register $y$. This works because $1 \cdot a^b = a^b$, which means that the start state $(a, 1, b)$ satisifies $P$. By the Invariant Principle, $P$ holds for all reachable states. But the program only stops when $z = 0$. If a terminated state $(x, y, 0)$ is reachable, then $y = yx^0 = a^b$ as required.

Ok, it’s partially correct, but what’s fast about it? The answer is that the number of multiplications it performs to compute $a^b$ is roughly the length of the binary representation of $b$. That is, the Fast Exponentiation program uses roughly $\log b$[2] multiplications, compared to the naive approach of multiplying by $a$ a total of $b-1$ times.

More precisely, it requires at most $2(\lceil \log b \rceil + 1)$ multiplications for the Fast Exponentiation algorithm to compute $a^b$ for $b > 1$. The reason is that the number in register $z$ is initially $b$, and gets at least halved with each transition. So it can’t be halved more than $\lceil \log b \rceil + 1$ times before hitting zero and causing the program to terminate. Since each of the transitions involves at most two multiplications, the total number of multiplications until $z = 0$ is at most $2(\lceil \log b \rceil + 1)$ for $b > 0$ (see Problem 6.6).

[2] As usual in computer science, $\log b$ means the base two logarithm $\log_2 b$. We use, $\ln b$ for the natural logarithm $\log_e b$, and otherwise write the logarithm base explicitly, as in $\log_{10} b$.

### 6.3.2 Derived Variables

The preceding termination proof involved finding a nonnegative integer-valued measure to assign to states. We might call this measure the "size" of the state. We then showed that the size of a state decreased with every state transition. By the Well Ordering Principle, the size can't decrease indefinitely, so when a minimum size state is reached, there can't be any transitions possible: the process has terminated.

More generally, the technique of assigning values to states—not necessarily nonnegative integers and not necessarily decreasing under transitions—is often useful in the analysis of algorithms. *Potential functions* play a similar role in physics. In the context of computational processes, such value assignments for states are called *derived variables*.

For example, for the Die Hard machines we could have introduced a derived variable $f : \text{states} \to \mathbb{R}$ 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 $x$-coord, where $x\text{-coord}((i,j)) ::= i$.

There are a few standard properties of derived variables that are handy in analyzing state machines.

**Definition 6.3.1.** A derived variable $f : \text{states} \to \mathbb{R}$ is *strictly decreasing* iff

$$q \longrightarrow q' \text{ IMPLIES } f(q') < f(q).$$

It is *weakly decreasing* iff

$$q \longrightarrow q' \text{ IMPLIES } f(q') \le f(q).$$

*Strictly increasingweakly increasing* derived variables are defined similarly.[3]

We confirmed termination of the Fast Exponentiation procedure by noticing that the derived variable $z$ was nonnegative-integer-valued and strictly decreasing. We can summarize this approach to proving termination as follows:

**Theorem 6.3.2.** *If $f$ is a strictly decreasing $\mathbb{N}$-valued derived variable of a state machine, then the length of any execution starting at state $q$ is at most $f(q)$.*

Of course, we could prove Theorem 6.3.2 by induction on the value of $f(q)$, but think about what it says: "If you start counting down at some nonnegative integer $f(q)$, then you can't count down more than $f(q)$ times." Put this way, it's obvious.

[3] Weakly increasing variables are often also called *nondecreasing*. We will avoid this terminology to prevent confusion between nondecreasing variables and variables with the much weaker property of *not* being a decreasing variable.

### 6.3.3 Termination with Well ordered Sets (Optional)

Theorem 6.3.2 generalizes straightforwardly to derived variables taking values in a well ordered set (Section 2.4.

**Theorem 6.3.3.** *If there exists a strictly decreasing derived variable whose range is a well ordered set, then every execution terminates.*

Theorem 6.3.3 follows immediately from the observation that a set of numbers is well ordered iff it has no infinite decreasing sequences (Problem 2.23).

Note that the existence of a *weakly* decreasing derived variable does not guarantee that every execution terminates. An infinite execution could proceed through states in which a weakly decreasing variable remained constant.

### 6.3.4 A Southeast Jumping Robot (Optional)

Here’s a simple, contrived example of a termination proof based on a variable that is strictly decreasing over a well ordered set. Let’s think about a robot that travels around the nonnegative integer quadrant $\mathbb{N}^2$.

If the robot is at some position $(x, y)$ different from the origin $(0, 0)$, the robot must make a move, which may be

- a unit distance West—that is, $(x, y) \longrightarrow (x - 1, y)$ for $x > 0$, or
- a unit distance South combined with an arbitrary jump East—that is, $(x, y) \longrightarrow (z, y - 1)$ for $z \geq x$,

providing the move does not leave the quadrant.

**Claim 6.3.4.** *The robot will always get stuck at the origin.*

If we think of the robot as a nondeterministic state machine, then Claim 6.3.4 is a termination assertion. The Claim may seem obvious, but it really has a different character than termination based on nonnegative integer-valued variables. That’s because, even knowing that the robot is at position $(0, 1)$, for example, there is no way to bound the time it takes for the robot to get stuck. It can delay getting stuck for as many seconds as it wants by making its next move to a distant point in the Far East. This rules out proving termination using Theorem 6.3.2.

So does Claim 6.3.4 still seem obvious?

Well it is if you see the trick. Define a derived variable $v$ mapping robot states to the numbers in the well ordered set $\mathbb{N} + \mathbb{F}$ of Lemma 2.4.6. In particular, define $v : \mathbb{N}^2 \to \mathbb{N} + \mathbb{F}$ as follows

$$v(x, y) ::= y + \frac{x}{x + 1}.$$

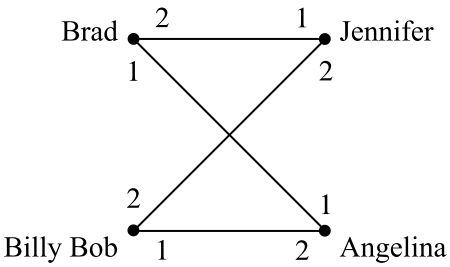


**Figure 6.4** Preferences for four people. Both men like Angelina best and both women like Brad best.

Now it's easy to check that if $(x, y) \longrightarrow (x', y')$ is a legitimate robot move, then $v((x', y')) < v((x, y))$. In particular, $v$ is a strictly decreasing derived variable, so Theorem 6.3.3 implies that the robot always get stuck—even though we can't say how many moves it will take until it does.

## 6.4 The Stable Marriage Problem

Suppose we have a population of men and women in which each person has preferences of the opposite-gender person they would like to marry: each man has his preference list of all the women, and each woman has her preference list of all of the men.

The preferences don't have to be symmetric. That is, Jennifer might like Brad best, but Brad doesn't necessarily like Jennifer best. The goal is to marry everyone: every man must marry exactly one woman and *vice versa*—no polygamy and heterosexual marriages only.[4] Moreover, we would like to find a matching between men and women that is *stable* in the sense that there is no pair of people who prefer one another to their spouses.

For example, suppose Brad likes Angelina best, and Angelina likes Brad best, but Brad and Angelina are married to other people, say Jennifer and Billy Bob. Now *Brad and Angelina prefer each other to their spouses*, which puts their marriages at risk. Pretty soon, they're likely to start spending late nights together working on problem sets!

This unfortunate situation is illustrated in Figure 6.4, where the digits "1" and "2" near a man shows which of the two women he ranks first and second, respectively, and similarly for the women.

[4] Same-sex marriage is an interesting but separate case.

More generally, in any matching, a man and woman who are not married to each other and who like each other better than their spouses is called a *rogue couple*. In the situation shown in Figure 6.4, Brad and Angelina would be a rogue couple.

Having a rogue couple is not a good thing, since it threatens the stability of the marriages. On the other hand, if there are no rogue couples, then for any man and woman who are not married to each other, at least one likes their spouse better than the other, and so there won't be any mutual temptation to start an affair.

**Definition 6.4.1.** A *stable matching* is a matching with no rogue couples.

The question is, given everybody's preferences, can you find a stable set of marriages? In the example consisting solely of the four people in Figure 6.4, we could let Brad and Angelina both have their first choices by marrying each other. Now neither Brad nor Angelina prefers anybody else to their spouse, so neither will be in a rogue couple. This leaves Jen not-so-happily married to Billy Bob, but neither Jen nor Billy Bob can entice somebody else to marry them, and so this is a stable matching.

It turns out there *always* is a stable matching among a group of men and women. We don't know of any immediate way to recognize this, and it seems surprising. In fact, in the apparently similar same-sex or "buddy" matching problem where people are supposed to be paired off as buddies, regardless of gender, a stable matching *may not* be possible. An example of preferences among four people where there is no stable buddy match is given in Problem 6.22. But when men are only allowed to marry women, and *vice versa*, then there is a simple procedure to produce a stable matching and the concept of preserved invariants provides an elegant way to understand and verify the procedure.

### 6.4.1 The Mating Ritual

The procedure for finding a stable matching can be described in a memorable way as a *Mating Ritual* that takes place over several days. On the starting day, each man has his full preference list of all the women, and likewise each woman has her full preference list of all the men. Then following events happen each day:

**Morning**: Each man stands under the balcony of the woman on the top of his list, that is the woman he prefers above all the other remaining women. Then he serenades her. He is said to be her *suitor*. If a man has no women left on his list, he stays home and does his math homework.

**Afternoon**: Each woman who has one or more suitors says to her favorite among them, "We might get engaged. Please stay around." To the other suitors, she says, "No. I will never marry you! Take a hike!"

**Evening**: Any man who is told by a woman to take a hike crosses that woman off his preference list.

**Termination condition**: When a day arrives in which every woman has at most one suitor, the ritual ends with each woman marrying her suitor, if she has one.

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

- The Ritual eventually reaches the termination condition.
- Everybody ends up married.
- The resulting marriages are stable.

To prove these facts, it will be helpful to recognize the Ritual as the description of a state machine. The state at the start of any day is determined by knowing for each man, which woman, if any, he will serenade that day—that is, the woman at the top of his preference list after he has crossed out all the women who have rejected him on earlier days.

**Mating Ritual at Akamai**

The Internet infrastructure company Akamai, cofounded by Tom Leighton, also uses a variation of the Mating Ritual to assign web traffic to its servers.

In the early days, Akamai used other combinatorial optimization algorithms that got to be too slow as the number of servers (over 65,000 in 2010) and requests (over 800 billion per day) increased. Akamai switched to a Ritual-like approach, since a Ritual is fast and can be run in a distributed manner. In this case, web requests correspond to women and web servers correspond to men. The web requests have preferences based on latency and packet loss, and the web servers have preferences based on cost of bandwidth and co-location.

### 6.4.2 There is a Marriage Day

It’s easy to see why the Mating Ritual has a terminal day when people finally get married. Every day on which the ritual hasn’t terminated, at least one man crosses a woman off his list. (If the ritual hasn’t terminated, there must be some woman serenaded by at least two men, and at least one of them will have to cross her off his list). If we start with $n$ men and $n$ women, then each of the $n$ men’s lists initially has $n$ women on it, for a total of $n^2$ list entries. Since no woman ever gets added to a list, the total number of entries on the lists decreases every day that the Ritual continues, and so the Ritual can continue for at most $n^2$ days.

### 6.4.3 They All Live Happily Ever After…

We will prove that the Mating Ritual leaves everyone in a stable marriage. To do this, we note one very useful fact about the Ritual: if on some morning a woman has any suitor, then her favorite suitor will still be serenading her the next morning—his list won’t have changed. So she is sure to have today’s favorite suitor among her suitors tomorrow. That means she will be able to choose a favorite suitor tomorrow who is at least as desirable to her as today’s favorite. So day by day, her favorite suitor can stay the same or get better, never worse. This sounds like an invariant, and it is. Namely, let $P$ be the predicate

> For every woman $w$ and man $m$, if $w$ is crossed off $m$’s list, then $w$ has a suitor whom she prefers over $m$.

**Lemma 6.4.2.** *$P$ is a preserved invariant for The Mating Ritual.*

*Proof.* Woman $w$ gets crossed off $m$’s list only when $w$ has a suitor she prefers to $m$. Thereafter, her favorite suitor doesn’t change until one she likes better comes along. So if her favorite suitor was preferable to $m$, then any new favorite suitor will be as well. ■

Notice that the invariant $P$ holds vacuously at the beginning since no women are crossed off to start. So by the Invariant Principle, $P$ holds throughout the Ritual. Now we can prove:

**Theorem 6.4.3.** *Everyone is married at the end of the Mating Ritual.*

*Proof.* Assume to the contrary that someone is unmarried on the last day of the Mating Ritual. There are an equal number of men and women, so some some man—call him Bob—as well as some woman—call her Alice—are unmarried. This means Bob isn’t serenading anybody on the last day, that is, his list is empty.

In particular, Alice has been crossed off Bob's list and, since $P$ is true, Alice has a suitor on the last day. By definition, Alice is married to this suitor, a contradiction. ■

**Theorem 6.4.4.** *The Mating Ritual produces a stable matching.*

*Proof.* Let Brad and Jen be any man and woman, respectively, that are *not* married to each other on the last day of the Mating Ritual. We will prove that Brad and Jen are not a rogue couple, and thus that all marriages on the last day are stable. There are two cases to consider.

**Case 1:** Jen is not on Brad's list by the end. Then by invariant $P$, we know that Jen has a suitor (and hence a husband) whom she prefers to Brad. So she's not going to run off with Brad—Brad and Jen cannot be a rogue couple.

**Case 2:** Jen is on Brad's list. Since Brad picks women to serenade by working down his list, his wife must be higher on his preference list than Jen. So he's not going to run off with Jen—once again, Brad and Jen are not a rogue couple. ■

### 6.4.4 . . . Especially the Men

Who is favored by the Mating Ritual, the men or the women? The women *seem* to have all the power: each day they choose their favorite suitor and reject the rest. What's more, we know their suitors can only change for the better as the Ritual progresses. Similarly, a man keeps serenading the woman he most prefers among those on his list until he must cross her off, at which point he serenades the next most preferred woman on his list. So from the man's perspective, the woman he is serenading can only change for the worse. Sounds like a good deal for the women.

But it's not! We will show that the men are by far the favored gender under the Mating Ritual.

While the Mating Ritual produces one stable matching, stable matchings need not be unique. For example, reversing the roles of men and women will often yield a different stable matching among them. So a man may have different wives in different sets of stable marriages. In some cases, a man can stably marry every one of the women, but in most cases, there are some women who cannot be a man's wife in any stable matching. For example, given the preferences shown in Figure 6.4, Jennifer cannot be Brad's wife in any stable matching because if he was married to her, then he and Angelina would be a rogue couple. It is not feasible for Jennifer to be stably married to Brad.

**Definition 6.4.5.** Given a set of preferences for the men and women, one person is a *feasible spouse* for another person when there is a stable matching in which these two people are married.

**Definition 6.4.6.** Let $Q$ be the predicate: for every woman $w$ and man $m$, if $w$ is crossed off $m$'s list, then $w$ is not a feasible spouse for $m$.

**Lemma 6.4.7.** *$Q$ is a preserved invariant*[5] *for The Mating Ritual.*

*Proof.* Suppose $Q$ holds at some point in the Ritual and some woman Alice is about to be crossed off some man's, Bob's, list. We claim that Alice must not be feasible for Bob. Therefore $Q$ will still hold after Alice is crossed off, proving that $Q$ is invariant.

To verify the claim, notice that when Alice gets crossed of Bob's list, it's because Alice has a suitor, Ted, she prefers to Bob. What's more, since $Q$ holds, all Ted's feasible wives are still on his list, and Alice is at the top. So Ted likes Alice better than all his other feasible spouses. Now if Alice could be married to Bob in some set of stable marriages, then Ted must be married to a wife he likes less than Alice, making Alice and Ted a rogue couple and contradicting stability. So Alice can't be married to Bob, that is, Alice is not a feasible wife for Bob, as claimed. ■

**Definition 6.4.8.** Given a set of preferences for the men and women, a person's *optimal spouse* is their most preferred feasible spouse. A person's *pessimal spouse* is their least preferred feasible spouse.

Everybody has an optimal and a pessimal spouse, since we know there is at least one stable matching, namely, the one produced by the Mating Ritual. Lemma 6.4.7 implies a key property the Mating Ritual:

**Theorem 6.4.9.** *The Mating Ritual marries every man to his optimal wife.*

*Proof.* If Bob is married to Alice on the final day of the Ritual, then everyone above Alice on Bob's preference list was crossed off, and by property $Q$, all these crossed off women were infeasible for Bob. So Alice is Bob's highest ranked feasible spouse, that is, his optimal wife. ■

**Lemma 6.4.10.** *Given a set of preferences for the men and women, everyone is the pessimal spouse of their optimal spouse.*

[5] We appeal to $P$ in justifying $Q$, so technically it is $P$ AND $Q$ which is actually the preserved invariant. But let's not be picky.

*Proof.* By symmetry, it is enough to prove that every man is the pessimal husband of his optimal wife.

Suppose Alice is Bob's optimal wife. Then in any stable set of marriages, if Alice liked her husband less than Bob, then Bob must be married to someone not optimal, and therefore Alice and Bob would be a rogue couple. So Alice must like all her feasible husbands at least as much as Bob. That is, Bob is Alice's pessimal husband. ■

**Corollary 6.4.11.** *The Mating Ritual marries every woman to her pessimal husband.*

### 6.4.5 Applications

The Mating Ritual was first announced in a paper by D. Gale and L.S. Shapley in 1962, but ten years before the Gale-Shapley paper was published, and unknown to them, a similar algorithm was being used to assign residents to hospitals by the National Resident Matching Program (NRMP). The NRMP has, since the turn of the twentieth century, assigned each year's pool of medical school graduates to hospital residencies (formerly called "internships"), with hospitals and graduates playing the roles of men and women.[6] Before the Ritual-like algorithm was adopted, there were chronic disruptions and awkward countermeasures taken to preserve unstable assignments of graduates to residencies. The Ritual resolved these problems so successfully, that it was used essentially without change at least through 1989.[7] For this and related work, Shapley was awarded the 2012 Nobel prize in Economics.

Not surprisingly, the Mating Ritual is also used by at least one large online dating agency. Of course there is no serenading going on—everything is handled by computer.

## Problems for Section 6.3

### Practice Problems

**Problem 6.1.**
Which states of the Die Hard 3 machine below have transitions to exactly two states?

[6] In this case there may be multiple women married to one man, but this is a minor complication, see Problem 6.23.

[7] Much more about the Stable Marriage Problem can be found in the very readable mathematical monograph by Dan Gusfield and Robert W. Irving, [27].

**Die Hard Transitions**

1. Fill the little jug: $(b, l) \longrightarrow (b, 3)$ for $l < 3$.
2. Fill the big jug: $(b, l) \longrightarrow (5, l)$ for $b < 5$.
3. Empty the little jug: $(b, l) \longrightarrow (b, 0)$ for $l > 0$.
4. Empty the big jug: $(b, l) \longrightarrow (0, l)$ for $b > 0$.
5. Pour from the little jug into the big jug: for $l > 0$,

$$(b, l) \longrightarrow \begin{cases} (b + l, 0) & \text{if } b + l \le 5, \\ (5, l - (5 - b)) & \text{otherwise.} \end{cases}$$

6. Pour from big jug into little jug: for $b > 0$,

$$(b, l) \longrightarrow \begin{cases} (0, b + l) & \text{if } b + l \le 3, \\ (b - (3 - l), 3) & \text{otherwise.} \end{cases}$$

## Homework Problems

### Problem 6.2.

In the late 1960s, the military junta that ousted the government of the small republic of Nerdia completely outlawed built-in multiplication operations, and also forbade division by any number other than 3. Fortunately, a young dissident found a way to help the population multiply any two nonnegative integers without risking persecution by the junta. The procedure he taught people is:

```
procedure multiply($x$, $y$: nonnegative integers)
$r := x$;
$s := y$;
$a := 0$;
while $s \neq 0$ do
    if $3 \mid s$ then
        $r := r + r + r$;
        $s := s/3$;
    else if $3 \mid (s - 1)$ then
        $a := a + r$;
        $r := r + r + r$;
        $s := (s - 1)/3$;
    else
```

```
        $a := a + r + r$;
        $r := r + r + r$;
        $s := (s - 2)/3$;
**return** $a$;
```

We can model the algorithm as a state machine whose states are triples of nonnegative integers $(r, s, a)$. The initial state is $(x, y, 0)$. The transitions are given by the rule that for $s > 0$:

$$(r, s, a) \to \begin{cases} (3r, s/3, a) & \text{if } 3 \mid s \\ (3r, (s-1)/3, a + r) & \text{if } 3 \mid (s - 1) \\ (3r, (s-2)/3, a + 2r) & \text{otherwise.} \end{cases}$$

**(a)** List the sequence of steps that appears in the execution of the algorithm for inputs $x = 5$ and $y = 10$.

**(b)** Use the Invariant Method to prove that the algorithm is partially correct—that is, if $s = 0$, then $a = xy$.

**(c)** Prove that the algorithm terminates after at most $1 + \log_3 y$ executions of the body of the **do** statement.

**Problem 6.3.**

A robot named Wall-E wanders around a two-dimensional grid. He starts out at (0, 0) and is allowed to take four different types of steps:

1. $(+2, -1)$
2. $(+1, -2)$
3. $(+1, +1)$
4. $(-3, 0)$

Thus, for example, Wall-E might walk as follows. The types of his steps are listed above the arrows.

$$(0,0) \xrightarrow{1} (2,-1) \xrightarrow{3} (3,0) \xrightarrow{2} (4,-2) \xrightarrow{4} (1,-2) \to \dots$$

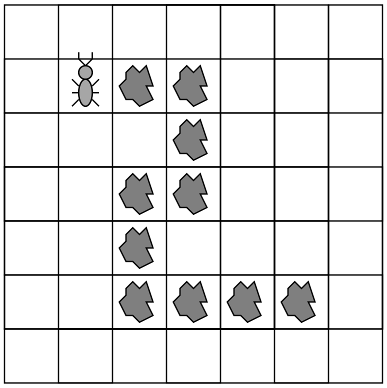
Wall-E's true love, the fashionable and high-powered robot, Eve, awaits at (0, 2).

**(a)** Describe a state machine model of this problem. What are the states? The transitions?

**(b)** Will Wall-E ever find his true love? If yes, find a path from Wall-E to Eve. If no, use the Invariant Principle to prove that no such path exists, being sure to clearly state and prove your preserved invariant. *Hint*: The value $x - y$ is not preserved, but how can it change?

### Problem 6.4.

A hungry ant is placed on an unbounded grid. Each square of the grid either contains a crumb or is empty. The squares containing crumbs form a path in which, except at the ends, every crumb is adjacent to exactly two other crumbs. The ant is placed at one end of the path and on a square containing a crumb. For example, the figure below shows a situation in which the ant faces North, and there is a trail of food leading approximately Southeast. The ant has already eaten the crumb upon which it was initially placed.



The ant can only smell food directly in front of it. The ant can only remember a small number of things, and what it remembers after any move only depends on what it remembered and smelled immediately before the move. Based on smell and memory, the ant may choose to move forward one square, or it may turn right or left. It eats a crumb when it lands on it.

The above scenario can be nicely modelled as a state machine in which each state is a pair consisting of the “ant’s memory” and “everything else”—for example, information about where things are on the grid. Work out the details of such a model state machine; design the ant-memory part of the state machine so the ant will eat all the crumbs on *any* finite path at which it starts and then signal when it is done. Be sure to clearly describe the possible states, transitions, and inputs and outputs (if any) in your model. Briefly explain why your ant will eat all the crumbs.

Note that the last transition is a self-loop; the ant signals done for eternity. One could also add another end state so that the ant signals done only once.

**Problem 6.5.**
Suppose that you have a regular deck of cards arranged as follows, from top to bottom:

$$A\heartsuit\ 2\heartsuit \dots K\heartsuit\ A\spadesuit\ 2\spadesuit \dots K\spadesuit\ A\clubsuit\ 2\clubsuit \dots K\clubsuit\ A\diamondsuit\ 2\diamondsuit \dots K\diamondsuit$$

Only two operations on the deck are allowed: *inshuffling* and *outshuffling*. In both, you begin by cutting the deck exactly in half, taking the top half into your right-hand and the bottom into your left. Then you shuffle the two halves together so that the cards are perfectly interlaced; that is, the shuffled deck consists of one card from the left, one from the right, one from the left, one from the right, etc. The top card in the shuffled deck comes from the right-hand in an outshuffle and from the left-hand in an inshuffle.

**(a)** Model this problem as a state machine.

**(b)** Use the Invariant Principle to prove that you cannot make the entire first half of the deck black through a sequence of inshuffles and outshuffles.

*Note*: Discovering a suitable invariant can be difficult! This is the part of a correctness proof that generally requires some insight, and there is no simple recipe for finding invariants. A standard initial approach is to identify a bunch of reachable states and then look for a pattern—some feature that they all share.

**Problem 6.6.**
Prove that the fast exponentiation state machine of Section 6.3.1 will halt after

$$\lceil \log_2 n \rceil + 1 \tag{6.3}$$

transitions starting from any state where the value of $z$ is $n \in \mathbb{Z}^+$.

*Hint:* Strong induction.

## Class Problems

**Problem 6.7.**
In this problem you will establish a basic property of a puzzle toy called the *Fifteen Puzzle* using the method of invariants. The Fifteen Puzzle consists of sliding square

tiles numbered $1, \ldots, 15$ held in a $4 \times 4$ frame with one empty square. Any tile adjacent to the empty square can slide into it.

The standard initial position is

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | |

We would like to reach the target position (known in the oldest author's youth as "the impossible"):

| | | | |
|---|---|---|---|
| 15 | 14 | 13 | 12 |
| 11 | 10 | 9 | 8 |
| 7 | 6 | 5 | 4 |
| 3 | 2 | 1 | |

A state machine model of the puzzle has states consisting of a $4 \times 4$ matrix with 16 entries consisting of the integers $1, \ldots, 15$ as well as one "empty" entry—like each of the two arrays above.

The state transitions correspond to exchanging the empty square and an adjacent numbered tile. For example, an empty at position $(2, 2)$ can exchange position with tile above it, namely, at position $(1, 2)$:

$$\begin{array}{|c|c|c|c|}\hline n_1 & n_2 & n_3 & n_4 \\ \hline n_5 & & n_6 & n_7 \\ \hline n_8 & n_9 & n_{10} & n_{11} \\ \hline n_{12} & n_{13} & n_{14} & n_{15} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|}\hline n_1 & & n_3 & n_4 \\ \hline n_5 & n_2 & n_6 & n_7 \\ \hline n_8 & n_9 & n_{10} & n_{11} \\ \hline n_{12} & n_{13} & n_{14} & n_{15} \\ \hline \end{array}$$

We will use the invariant method to prove that there is no way to reach the target state starting from the initial state.

We begin by noting that a state can also be represented as a pair consisting of two things:

1. a list of the numbers $1, \ldots, 15$ in the order in which they appear—reading rows left-to-right from the top row down, ignoring the empty square, and

2. the coordinates of the empty square—where the upper left square has coordinates $(1, 1)$, the lower right $(4, 4)$.

**(a)** Write out the "list" representation of the start state and the "impossible" state.

Let $L$ be a list of the numbers $1, \ldots, 15$ in some order. A pair of integers is an *out-of-order pair* in $L$ when the first element of the pair both comes *earlier* in the list and *is larger*, than the second element of the pair. For example, the list $1, 2, 4, 5, 3$ has two out-of-order pairs: (4,3) and (5,3). The increasing list $1, 2 \ldots n$ has no out-of-order pairs.

Let a state $S$ be a pair $(L, (i, j))$ described above. We define the *parity* of $S$ to be 0 or 1 depending on whether the sum of the number of out-of-order pairs in $L$ and the row-number of the empty square is even or odd. that is

$$\text{parity}(S) ::= \begin{cases} 0 & \text{if } p(L) + i \text{ is even,} \\ 1 & \text{otherwise.} \end{cases}$$

**(b)** Verify that the parity of the start state and the target state are different.

**(c)** Show that the parity of a state is preserved under transitions. Conclude that "the impossible" is impossible to reach.

By the way, if two states have the same parity, then in fact there *is* a way to get from one to the other. If you like puzzles, you'll enjoy working this out on your own.

**Problem 6.8.**

The Massachusetts Turnpike Authority is concerned about the integrity of the Zakim bridge. Their consulting architect has warned that the bridge may collapse if more than 1000 cars are on it at the same time. The Authority has also been warned by their traffic consultants that the rate of accidents from cars speeding across bridges has been increasing.

Both to lighten traffic and to discourage speeding, the Authority has decided to make the bridge *one-way* and to put tolls at *both* ends of the bridge (don't laugh, this is Massachusetts). So cars will pay tolls both on entering and exiting the bridge, but the tolls will be different. In particular, a car will pay \$3 to enter onto the bridge and will pay \$2 to exit. To be sure that there are never too many cars on the bridge, the Authority will let a car onto the bridge only if the difference between the amount of money currently at the entry toll booth and the amount at the exit toll booth is strictly less than a certain threshold amount of $\$T_0$.

The consultants have decided to model this scenario with a state machine whose states are triples $(A, B, C)$ of nonnegative integers, where

- $A$ is an amount of money at the entry booth,
- $B$ is an amount of money at the exit booth, and
- $C$ is a number of cars on the bridge.

Any state with $C > 1000$ is called a *collapsed* state, which the Authority dearly hopes to avoid. There will be no transition out of a collapsed state.

Since the toll booth collectors may need to start off with some amount of money in order to make change, and there may also be some number of “official” cars already on the bridge when it is opened to the public, the consultants must be ready to analyze the system started at *any* uncollapsed state. So let $A_0$ be the initial number of dollars at the entrance toll booth, $B_0$ the initial number of dollars at the exit toll booth, and $C_0 \leq 1000$ the number of official cars on the bridge when it is opened. You should assume that even official cars pay tolls on exiting or entering the bridge after the bridge is opened.

**(a)** Give a mathematical model of the Authority’s system for letting cars on and off the bridge by specifying a transition relation between states of the form $(A, B, C)$ above.

**(b)** Characterize each of the following derived variables

$$A, B, A + B, A - B, 3C - A, 2A - 3B, B + 3C, 2A - 3B - 6C, 2A - 2B - 3C$$

as one of the following

| | |
|---|---|
| constant | C |
| strictly increasing | SI |
| strictly decreasing | SD |
| weakly increasing but not constant | WI |
| weakly decreasing but not constant | WD |
| none of the above | N |

and briefly explain your reasoning.

The Authority has asked their engineering consultants to determine $T$ and to verify that this policy will keep the number of cars from exceeding 1000.

The consultants reason that if $C_0$ is the number of official cars on the bridge when it is opened, then an additional $1000 - C_0$ cars can be allowed on the bridge. So as long as $A - B$ has not increased by $3(1000 - C_0)$, there shouldn’t more than 1000 cars on the bridge. So they recommend defining

$$T_0 ::= 3(1000 - C_0) + (A_0 - B_0), \tag{6.4}$$

where $A_0$ is the initial number of dollars at the entrance toll booth, $B_0$ is the initial number of dollars at the exit toll booth.

**(c)** Use the results of part (b) to define a simple predicate $P$ on states of the transition system which is satisfied by the start state—that is $P(A_0, B_0, C_0)$ holds—is not satisfied by any collapsed state, and is a preserved invariant of the system. Explain why your $P$ has these properties. Conclude that the traffic won’t cause the bridge to collapse.

**(d)** A clever MIT intern working for the Turnpike Authority agrees that the Turnpike’s bridge management policy will be *safe*: the bridge will not collapse. But she warns her boss that the policy will lead to *deadlock*—a situation where traffic can’t move on the bridge even though the bridge has not collapsed.

Explain more precisely in terms of system transitions what the intern means, and briefly, but clearly, justify her claim.

**Problem 6.9.**
Start with 102 coins on a table, 98 showing heads and 4 showing tails. There are two ways to change the coins:

(i) flip over any ten coins, or

(ii) let $n$ be the number of heads showing. Place $n + 1$ additional coins, all showing tails, on the table.

For example, you might begin by flipping nine heads and one tail, yielding 90 heads and 12 tails, then add 91 tails, yielding 90 heads and 103 tails.

**(a)** Model this situation as a state machine, carefully defining the set of states, the start state, and the possible state transitions.

**(b)** Explain how to reach a state with exactly one tail showing.

**(c)** Define the following derived variables:[8]

| | | | | | |
|---|---|---|---|---|---|
| $C$ | ::= | the number of coins on the table, | $H$ | ::= | the number of heads, |
| $T$ | ::= | the number of tails, | $C_2$ | ::= | $\text{parity}(C)$, |
| $H_2$ | ::= | $\text{parity}(H)$, | $T_2$ | ::= | $\text{parity}(T)$. |

Which of these variables is

1. strictly increasing?
2. weakly increasing?
3. strictly decreasing?

[8] The function parity : $\mathbb{Z} \to \{0, 1\}$ is defined as follows: $\text{parity}(n) = 0$ when $n$ is even, and $\text{parity}(n) = 1$ when $n$ is odd.

4. weakly decreasing?
5. constant?

No need to prove these answers.

**(d)** Prove using the Invariant principle that it is not possible to reach a state in which there is exactly one head showing. If you wish to use a fact from part c, please prove it carefully here.

**Problem 6.10.**
A classroom is designed so students sit in a square arrangement. An outbreak of beaver flu sometimes infects students in the class; beaver flu is a rare variant of bird flu that lasts forever, with symptoms including a yearning for more quizzes and the thrill of late night problem set sessions.

Here is an illustration of a $6 \times 6$-seat classroom with seats represented by squares. The locations of infected students are marked with an asterisk.

| | | | | | |
|---|---|---|---|---|---|
| * | | | | * | |
| | * | | | | |
| | | * | * | | |
| | | | | | |
| | | * | | | |
| | | | * | | * |

Outbreaks of infection spread rapidly step by step. A student is infected after a step if either

- the student was infected at the previous step (since beaver flu lasts forever), or
- the student was adjacent to *at least two* already-infected students at the previous step.

Here *adjacent* means the students’ individual squares share an edge (front, back, left or right); they are not adjacent if they only share a corner point. So each student is adjacent to 2, 3 or 4 others.

In the example, the infection spreads as shown below.

| | | | | | |
|---|---|---|---|---|---|
| * | | | | * | |
| | * | | | | |
| | | * | * | | |
| | | | | | |
| | | * | | | |
| | | | * | | * |

⇒

| | | | | | |
|---|---|---|---|---|---|
| * | * | | | * | |
| * | * | * | | | |
| | * | * | * | | |
| | | * | | | |
| | | * | * | | |
| | | * | * | * | * |

⇒

| | | | | | |
|---|---|---|---|---|---|
| * | * | * | | * | |
| * | * | * | * | | |
| * | * | * | * | | |
| | * | * | * | | |
| | | * | * | * | |
| | | * | * | * | * |

In this example, over the next few time-steps, all the students in class become infected.

**Theorem.** *If fewer than n students among those in an n×n arrangment are initially infected in a flu outbreak, then there will be at least one student who never gets infected in this outbreak, even if students attend all the lectures.*

Prove this theorem.

*Hint:* Think of the state of an outbreak as an $n \times n$ square above, with asterisks indicating infection. The rules for the spread of infection then define the transitions of a state machine. Find a weakly decreasing derived variable that leads to a proof of this theorem.

## Exam Problems

### Problem 6.11.

*Token replacing-1-2* is a single player game using a set of tokens, each colored black or white. Except for color, the tokens are indistinguishable. In each move, a player can replace one black token with two white tokens, or replace one white token with two black tokens.

We can model this game as a state machine whose states are pairs $(n_b, n_w)$ where $n_b \geq 0$ equals the number of black tokens, and $n_w \geq 0$ equals the number of white tokens.

**(a)** List the numbers of the following predicates that are preserved invariants.

$$n_b + n_w \operatorname{rem}(n_b + n_w,\ 3) \neq 2 \tag{6.5}$$

$$n_w - n_b \operatorname{rem}(n_w - n_b,\ 3) = 2 \tag{6.6}$$

$$n_b - n_w \operatorname{rem}(n_b - n_w,\ 3) = 2 \tag{6.7}$$

$$n_b + n_w > 5 \tag{6.8}$$

$$n_b + n_w < 5 \tag{6.9}$$

Now assume the game starts with a single black token, that is, the start state is $(1, 0)$.

**(b)** List the numbers of the predicates above are true for all reachable states:

**(c)** Define the predicate $T(n_b, n_w)$ by the rule:

$$T(n_b, n_w) ::= \text{rem}(n_w - n_b,\ 3) = 2.$$

We will now prove the following:
**Claim.** *If $T(n_b, n_w)$, then state $(n_b, n_w)$ is reachable.*

Note that this claim is different from the claim that $T$ is a preserved invariant.

The proof of the Claim will be by induction in $n$ using induction hypothesis $P(n)$::=

$$\forall (n_b, n_w).\, [(n_b + n_w = n) \text{ AND } T(n_b, n_w)] \text{ IMPLIES } (n_b, n_w) \text{ is reachable.}$$

The base cases will be when $n \leq 2$.

- Assuming that the base cases have been verified, complete the **Inductive Step**.
- Now verify the **Base Cases**: $P(n)$ for $n \leq 2$.

**Problem 6.12.**
*Token Switching* is a process for updating a set of black and white tokens. The process starts with a single black token. At each step,

i. one black token can be replaced with two white tokens, or

ii. if the numbers of white and black tokens are not the same, the colors of all the tokens can be switched: all the black tokens become white, and the white tokens become black.

We can model Token Switching as a state machine whose states are pairs $(b, w)$ of nonnegative integers, where $b$ equals the number of black tokens, and $w$ equals the number of white tokens. So the start state is $(1, 0)$.

**(a)** Indicate which of the following states can be reached from the start state in *exactly* two steps:

$$(0,0),\ (1,0),\ (0,1),\ (1,1),\ (0,2),\ (2,0),\ (2,1),\ (1,2),\ (0,3),\ (3,0)$$

**(b)** Define the predicate $F(b, w)$ by the rule:

$$F(b, w) ::= \ (b - w) \text{ is not a multiple of 3.}$$

Prove the following
**Claim.** *If $F(b, w)$, then state $(b, w)$ is reachable from the start state.*

**(c)** Explain why state $(11^{6^{7777}}, 5^{10^{88}})$ is *not* a reachable state.

*Hint:* Do not assume $F$ is a preserved invariant without proving it.

**Problem 6.13.**
*Token replacing-1-3* is a single player game using a set of tokens, each colored black or white. In each move, a player can replace a black token with three white tokens, or replace a white token with three black tokens. We can model this game as a state machine whose states are pairs $(b, w)$ of nonnegative integers, where $b$ is the number of black tokens and $w$ the number of white ones.

The game has two possible start states: $(5, 4)$ or $(4, 3)$.

We call a state $(b, w)$ *eligible* when

$$\text{rem}(b - w, \ 4) = 1 \tag{6.10}$$

This problem examines the connection between eligible states and states that are *reachable* from either of the possible start states.

**(a)** Prove that the predicate

$$P(b, w) ::= \text{rem}(b - w, \ 4) = 1,$$

is a Preserved Invariant.

**(b)** Conlude that every reachable state is eligible.

**(c)** Prove that the eligible state $(3, 2)$ is not reachable. *Hint:* $b + w$ is strictly increasing.

Say that a state $(b, w)$ is *big enough* when $\min(b, w) > 2$. We now prove the

**Claim.** *Every big enough eligible state is reachable.*

To begin, observe the

**Fact.** If $\max(b, w) \leq 5$ and $(b, w)$ is eligible and big enough, then $(b, w)$ is reachable.

Among the nine states with $3 \leq \min(b, w)$ and $\max(b, w) \leq 5$, it is easy to check that only the start state $(5, 4)$ satisfies the preserved invariant. Now the Fact holds because by definition the start state is reachable.

**(d)** Prove the Claim by strong induction using the induction hypothesis:

$$Q(n) ::= \forall (b, w).[b + w = n \text{ AND } (b, w) \text{ is big enough and eligible}] \text{ IMPLIES } (b, w) \text{ is reachable.} \tag{6.11}$$

You may assume the Fact given above.

*Hint:* $Q(n-2)$ IMPLIES $Q(n)$.

**Problem 6.14.**
There is a bucket containing more blue balls than red balls. As long as there are more blues than reds, any one of the following rules may be applied to add and/or remove balls from the bucket:

(i) Add a red ball.

(ii) Remove a blue ball.

(iii) Add two reds and one blue.

(iv) Remove two blues and one red.

**(a)** Starting with 10 reds and 16 blues, what is the largest number of balls the bucket will contain by applying these rules?

Let $b$ be the number of blue balls and $r$ be the number of red balls in the bucket at any given time.

**(b)** Prove that $b - r \geq 0$ is a preserved invariant of the process of adding and removing balls according to rules (i)–(iv).

**(c)** Find a nonnegative integer-valued derived variable that is strictly decreasing, and use it to prove that no matter how many balls the bucket contains, repeatedly applying rules (i)–(iv) will eventually lead to a state where no further rule can be applied.

**Problem 6.15.**
The following problem is a twist on the Fifteen-Puzzle considered earlier in Problem 6.7.

Let $A$ be a sequence consisting of the numbers $1, \dots, n$ in some order. A pair of integers in $A$ is called an *out-of-order pair* when the first element of the pair both comes *earlier* in the sequence, and *is larger*, than the second element of the pair. For example, the sequence $(1,2,4,5,3)$ has two out-of-order pairs: $(4,3)$ and $(5,3)$. We let $t(A)$ equal the number of out-of-order pairs in $A$. For example, $t((1,2,4,5,3)) = 2$.

The elements in $A$ can be rearranged using the *Rotate-Triple* operation, in which three consecutive elements of $A$ are rotated to move the smallest of them to be first.

For example, in the sequence $(2,4,1,5,3)$, the *Rotate-Triple* operation could rotate the consecutive numbers $4,1,5$, into $1,5,4$ so that

$$(2,4,1,5,3) \longrightarrow (2,1,5,4,3).$$

The *Rotate-Triple* could also rotate the consecutive numbers $2,4,1$ into $1,2,4$ so that

$$(2,4,1,5,3) \longrightarrow (1,2,4,5,3).$$

We can think of a sequence $A$ as a state of a state machine whose transitions correspond to possible applications of the *Rotate-Triple* operation.

**(a)** Argue that the derived variable $t$ is *weakly decreasing*.

**(b)** Prove that having an even number of out-of-order pairs is a preserved invariant of this machine.

**(c)** Starting with

$$S ::= (2014, 2013, 2012, \dots, 2, 1),$$

explain why it is impossible to reach

$$T ::= (1, 2, \dots, 2012, 2013, 2014).$$

## Problems for Section 6.4

### Practice Problems

**Problem 6.16.**
Four Students want separate assignments to four VI-A Companies. Here are their preference rankings:

| Student | Companies |
|---:|---|
| Albert: | HP, Bellcore, AT&T, Draper |
| Sarah: | AT&T, Bellcore, Draper, HP |
| Tasha: | HP, Draper, AT&T, Bellcore |
| Elizabeth: | Draper, AT&T, Bellcore, HP |

| Company | Students |
|---:|---|
| AT&T: | Elizabeth, Albert, Tasha, Sarah |
| Bellcore: | Tasha, Sarah, Albert, Elizabeth |
| HP: | Elizabeth, Tasha, Albert, Sarah |
| Draper: | Sarah, Elizabeth, Tasha, Albert |

**(a)** Use the Mating Ritual to find *two* stable assignments of Students to Companies.

**(b)** Describe a simple procedure to determine whether any given stable marriage problem has a unique solution, that is, only one possible stable matching. Briefly explain why it works.

### Problem 6.17.

Suppose that Harry is one of the boys and Alice is one of the girls in the *Mating Ritual*. Which of the properties below are preserved invariants? Why?

a. Alice is the only girl on Harry's list.

b. There is a girl who does not have any boys serenading her.

c. If Alice is not on Harry's list, then Alice has a suitor that she prefers to Harry.

d. Alice is crossed off Harry's list, and Harry prefers Alice to anyone he is serenading.

e. If Alice is on Harry's list, then she prefers Harry to any suitor she has.

### Problem 6.18.

Prove that whatever the marriage preferences among the men and women, every man is the pessimal husband of his optimal wife.

*Hint:* Follows directly from the definition of "rogue couple."

**Problem 6.19.**
In the Mating Ritual for stable marriages between an equal number of boys and girls, explain why there must be a girl to whom no boy proposes (serenades) until the last day.

## Class Problems

**Problem 6.20.**
The preferences among 4 boys and 4 girls are partially specified in the following table:

| | | | | |
|---|---|---|---|---|
| B1: | G1 | G2 | – | – |
| B2: | G2 | G1 | – | – |
| B3: | – | – | G4 | G3 |
| B4: | – | – | G3 | G4 |
| G1: | B2 | B1 | – | – |
| G2: | B1 | B2 | – | – |
| G3: | – | – | B3 | B4 |
| G4: | – | – | B4 | B3 |

**(a)** Verify that

$$(B1, G1), (B2, G2), (B3, G3), (B4, G4)$$

will be a stable matching whatever the unspecified preferences may be.

**(b)** Explain why the stable matching above is neither boy-optimal nor boy-pessimal and so will not be an outcome of the Mating Ritual.

**(c)** Describe how to define a set of marriage preferences among $n$ boys and $n$ girls which have at least $2^{n/2}$ stable assignments.

*Hint:* Arrange the boys into a list of $n/2$ pairs, and likewise arrange the girls into a list of $n/2$ pairs of girls. Choose preferences so that the $k$th pair of boys ranks the $k$th pair of girls just below the previous pairs of girls, and likewise for the $k$th pair of girls. Within the $k$th pairs, make sure each boy's first choice girl in the pair prefers the other boy in the pair.

**Problem 6.21.**
The Mating Ritual of Section 6.4.1 for finding stable marriages works even when the numbers of men and women are not equal. As before, a set of (monogamous) marriages between men and women is called stable when it has no “rogue couples.”

**(a)** Extend the definition of *rogue couple* so it covers the case of unmarried men and women. Verify that in a stable set of marriages, either all the men are married or all the women are married.

**(b)** Explain why even in the case of unequal numbers of men and women, applying the Mating Ritual will yield a stable matching.

## Homework Problems

**Problem 6.22.**
Suppose we want to assign pairs of “buddies,” who may be of the same gender, where each person has a preference rank for who they would like to be buddies with. The meaning of a *stable* buddy assignment is the same as for the familiar (gender segregated) stable matching problem. For the preference ranking given in Figure 6.5, show that there is no stable buddy assignment. In this figure Mergatroid’s preferences aren’t shown because they don’t affect the outcome.

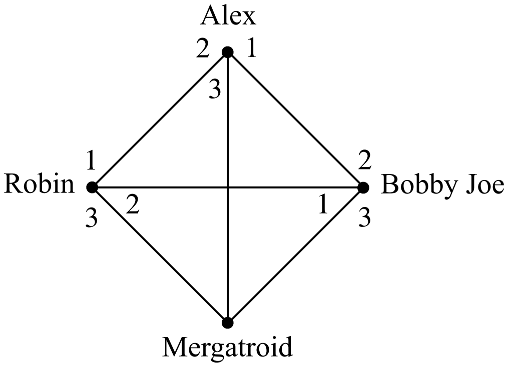


**Figure 6.5** Some preferences with no stable buddy matching.

**Problem 6.23.**
The most famous application of stable matching was in assigning graduating medical students to hospital residencies. Each hospital has a preference ranking of students, and each student has a preference ranking of hospitals, but unlike finding stable marriages between an equal number of boys and girls, hospitals generally have differing numbers of available residencies, and the total number of residencies may not equal the number of graduating students.

Explain how to adapt the Stable Matching problem with an equal number of boys and girls to this more general situation. In particular, modify the definition of stable matching so it applies in this situation, and explain how to adapt the Mating Ritual to handle it.

**Problem 6.24.**
Give an example of a stable matching between 3 boys and 3 girls where no person gets their first choice. Briefly explain why your matching is stable. Can your matching be obtained from the Mating Ritual or the Ritual with boys and girls reversed?

**Problem 6.25.**
In a stable matching between an equal number of boys and girls produced by the Mating Ritual, call a person *lucky* if they are matched up with someone in the top half of their preference list. Prove that there must be at least one lucky person.

*Hint:* The average number of times a boy gets rejected by girls.

**Problem 6.26.**
Suppose there are two stable sets of marriages. So each man has a first wife and a second wife , and likewise each woman has a first husband and a second husband.

Someone in a given marriage is a *winner* when they prefer their current spouse to their other spouse, and they are a *loser* when they prefer their other spouse to their current spouse. (If someone has the same spouse in both of their marriages, then they will be neither a winner nor a loser.)

We will show that

In each of the marriages, someone is a winner iff their spouse is a loser. (WL)

**(a)** The left to right direction of (WL) is equivalent to the assertion that married partners cannot both be winners. Explain why this follows directly from the definition of rogue couple.

The right to left direction of (WL) is equivalent to the assertion that a married couple cannot both be losers. This will follow by comparing the number of winners and losers among the marriages.

**(b)** Explain why the number of winners must equal the number of losers among the two sets of marriages.

**(c)** Complete the proof of (WL) by showing that if some married couple were both losers, then there must be another couple who were both winners.

**(d)** Conclude that in a stable set of marriages, someone’s spouse is optimal iff they are pessimal for their spouse.

**Problem 6.27.**
Suppose there are two stable sets of marriages, a first set and a second set. So each man has a first wife and a second wife (they may be the same), and likewise each woman has a first husband and a second husband. We can form a third set of marriages by matching each man with the wife he prefers among his first and second wives.

**(a)** Prove that this third set of marriages is an exact matching: no woman is married to two men.

**(b)** Prove that this third marriage set is stable.

*Hint:* You may assume the following fact from Problem 6.26.

$$\text{In every marriage, someone is a winner iff their spouse is a loser.} \tag{SL}$$

**Problem 6.28.**
A state machine has *commuting transitions* if for any states $p, q, r$

$$(p \longrightarrow q \text{ AND } p \longrightarrow r) \text{ IMPLIES } \exists t.\, q \longrightarrow t \text{ AND } r \longrightarrow t.$$

The state machine is *confluent* if

$$(p \longrightarrow^* q \text{ AND } p \longrightarrow^* r) \text{ IMPLIES } \exists t.\, q \longrightarrow^* t \text{ AND } r \longrightarrow^* t.$$

**(a)** Prove that if a state machine has commuting transitions, then it is confluent.

*Hint:* By induction on the number of moves from $p$ to $q$ plus the number from $p$ to $r$.

**(b)** A *final state* of a state machine is one from which no transition is possible. Explain why, if a state machine is confluent, then at most one final state is reachable from the start state.

**Problem 6.29.**
According to the day-by-day description of the Mating Ritual of Section 6.4.1, at the end of each day, *every* man’s list is updated to remove the name of the woman

he who rejected him. But it's easier, and more flexible, simply to let one women reject one suitor at a time.

In particular, the states of this Flexible Mating Ritual state machine will be the same as for the day-by-day Ritual: a state will be a list, for each man, of the women who have not rejected him. But now a transition will be to choose two men who are serenading the same woman—that is, who have the same woman at the top of their current lists—and then have the woman reject whichever of the two she likes less. So the only change in state is that the name of the serenaded woman gets deleted from the top of the list of the man she liked less among two of her serenaders—everything else stays the same.

It's a worthwhile review to verify that the same preserved invariants used to establish the properties of the Mating Ritual will apply to the Flexible Mating Ritual. This ensures that the Flexible Ritual will also terminate with a stable set of marriages.

But now a new issue arises: we know that there can be many sets of possible sets of stable marriages for the same set of men/women preferences. So it seems possible that the Flexible Ritual might terminate with different stable marriage sets, depending on which choice of transition was made at each state. But this does not happen: the Flexible Ritual will always terminate with the same set of stable marriages as the day-by-day Ritual.

To prove this, we begin with a definition: a state machine has *commuting transitions* if for any states $p, q, r$,

$$(p \longrightarrow q \text{ AND } p \longrightarrow r) \text{ IMPLIES } \exists t.\, q \longrightarrow t \text{ AND } r \longrightarrow t.$$

**(a)** Verify that the Flexible Mating Ritual has commuting transitions.

**(b)** Now conclude from Problem 6.28 that the Flexible Mating Ritual always terminate with the same set of stable marriages as the day-by-day Ritual.

## Exam Problems

### Problem 6.30.

Four unfortunate children want to be adopted by four foster families of ill repute. A child can only be adopted by one family, and a family can only adopt one child. Here are their preference rankings (most-favored to least-favored):

| Child | Families |
|---:|:---|
| Bottlecap: | Hatfields, McCoys, Grinches, Scrooges |
| Lucy: | Grinches, Scrooges, McCoys, Hatfields |
| Dingdong: | Hatfields, Scrooges, Grinches, McCoys |
| Zippy: | McCoys, Grinches, Scrooges, Hatfields |

| Family | Children |
|---|---|
| Grinches: | Zippy, Dingdong, Bottlecap, Lucy |
| Hatfields: | Zippy, Bottlecap, Dingdong, Lucy |
| Scrooges: | Bottlecap, Lucy, Dingdong, Zippy |
| McCoys: | Lucy, Zippy, Bottlecap, Dingdong |

**(a)** Exhibit two different stable matching of Children and Families.

| Family | Child in 1st match | Child in 2nd match |
|---|---|---|
| Grinches: | | |
| Hatfields: | | |
| Scrooges: | | |
| McCoys: | | |

**(b)** Examine the matchings from part (a), and explain why these matchings are the only two possible stable matchings between Children and Families.

*Hint:* In general, there may be many more than two stable matchings for the same set of preferences.

**Problem 6.31.**
The Mating Ritual 6.4.1 for finding stable marriages works without change when there are at least as many, and possibly more, men than women. You may assume this. So the Ritual ends with all the women married and no rogue couples for these marriages, where an unmarried man and a married woman who prefers him to her spouse is also considered to be a "rogue couple."

Let Alice be one of the women, and Bob be one of the men. Indicate which of the properties below that are preserved invariants of the Mating Ritual 6.4 when there are **at least as many** men as women. Briefly explain your answers.

**(a)** Alice has a suitor (man who is serenading her) whom she prefers to Bob.

**(b)** Alice is the only woman on Bob's list.

**(c)** Alice has no suitor.

**(d)** Bob prefers Alice to the women he is serenading.

**(e)** Bob is serenading Alice.

**(f)** Bob is not serenading Alice.

**(g)** Bob's list of women to serenade is empty.

**Problem 6.32.**
We want a stable matching between $n$ boys and $n$ girls for a positive integer $n$.

**(a)** Explain how to define preference rankings for the boys and the girls that allow only *one* possible stable matching. Briefly justify your answer.

**(b)** Mark each of the following predicates about the Stable Marriage Ritual **P** if it is a **P**reserved Invariant, **N** if it is **n**ot, and "**U**" if you are very unsure. "Bob's list" refers to the list of the women he has not crossed off.

(i) Alice is not on Bob's list.

(ii) No girl is on Bob's list.

(iii) Bob is the only boy serenading Alice.

(iv) Bob has fewer than 5 girls on his list.

(v) Bob prefers Alice to his favorite remaining girl.

(vi) Alice prefers her favorite current suitor to Bob.

(vii) Bob is serenading his optimal spouse.

(viii) Bob is serenading his pessimal spouse.

(ix) Alice's optimal spouse is serenading her.

(x) Alice's pessimal spouse is serenading her.