length. The constraints would say that two intersecting words must have the same letter in the intersecting box. Solving a problem in this formulation requires fewer steps, but the domains are larger (assuming a big dictionary) and there are fewer constraints. Both formulations are feasible.

5.5  a. For rectilinear floor-planning, one possibility is to have a variable for each of the small rectangles, with the value of each variable being a 4-tuple consisting of the $x$ and $y$ coordinates of the upper left and lower right corners of the place where the rectangle will be located. The domain of each variable is the set of 4-tuples that are the right size for the corresponding small rectangle and that fit within the large rectangle. Constraints say that no two rectangles can overlap; for example if the value of variable $R_1$ is $[0, 0, 5, 8]$, then no other variable can take on a value that overlaps with the $0, 0$ to $5, 8$ rectangle.

b. For class scheduling, one possibility is to have three variables for each class, one with times for values (e.g. MWF8:00, TuTh8:00, MWF9:00, ...), one with classrooms for values (e.g. Wheeler110, Evans330, ...) and one with instructors for values (e.g. Abelson, Bibel, Canny, ...). Constraints say that only one class can be in the same classroom at the same time, and an instructor can only teach one class at a time. There may be other constraints as well (e.g. an instructor should not have two consecutive classes).

5.6  The exact steps depend on certain choices you are free to make; here are the ones I made:

a. Choose the $X_3$ variable. Its domain is $\{0, 1\}$.

b. Choose the value 1 for $X_3$. (We can’t choose 0; it wouldn’t survive forward checking, because it would force $F$ to be 0, and the leading digit of the sum must be non-zero.)

c. Choose $F$, because it has only one remaining value.

d. Choose the value 1 for $F$.

e. Now $X_2$ and $X_1$ are tied for minimum remaining values at 2; let’s choose $X_2$.

f. Either value survives forward checking, let’s choose 0 for $X_2$.

g. Now $X_1$ has the minimum remaining values.

h. Again, arbitrarily choose 0 for the value of $X_1$.

i. The variable $O$ must be an even number (because it is the sum of $T + T$ less than 5 (because $O + O = R + 10 \times 0$). That makes it most constrained.

j. Arbitrarily choose 4 as the value of $O$.

k. $R$ now has only 1 remaining value.

l. Choose the value 8 for $R$.

m. $T$ now has only 1 remaining value.

n. Choose the value 7 for $T$.

o. $U$ must be an even number less than 9; choose $U$.

p. The only value for $U$ that survives forward checking is 6.

q. The only variable left is $W$.

r. The only value left for $W$ is 3.
s. This is a solution.

This is a rather easy (under-constrained) puzzle, so it is not surprising that we arrive at a solution with no backtracking (given that we are allowed to use forward checking).

5.7 There are implementations of CSP algorithms in the Java, Lisp, and Python sections of the online code repository; these should help students get started. However, students will have to add code to keep statistics on the experiments, and perhaps will want to have some mechanism for making an experiment return failure if it exceeds a certain time limit (or number-of-steps limit). The amount of code that needs to be written is small; the exercise is more about running and analyzing an experiment.

5.8 We’ll trace through each iteration of the while loop in AC-3 (for one possible ordering of the arcs):

a. Remove $SA - WA$, delete $R$ from $SA$.
b. Remove $SA - V$, delete $B$ from $SA$, leaving only $G$.
c. Remove $NT - WA$, delete $R$ from $NT$.
d. Remove $NT - SA$, delete $G$ from $NT$, leaving only $B$.
e. Remove $NSW - SA$, delete $G$ from $NSW$.
f. Remove $NSW - V$, delete $B$ from $NSW$, leaving only $R$.
g. Remove $Q - NT$, delete $B$ from $Q$.
h. Remove $Q - SA$, delete $G$ from $Q$.
i. remove $Q - NSW$, delete $R$ from $Q$, leaving no domain for $Q$.

5.9 On a tree-structured graph, no arc will be considered more than once, so the AC-3 algorithm is $O(ED)$, where $E$ is the number of edges and $D$ is the size of the largest domain.

5.10 The basic idea is to preprocess the constraints so that, for each value of $X_i$, we keep track of those variables $X_k$ for which an arc from $X_k$ to $X_i$ is satisfied by that particular value of $X_i$. This data structure can be computed in time proportional to the size of the problem representation. Then, when a value of $X_i$ is deleted, we reduce by 1 the count of allowable values for each $(X_k, X_i)$ arc recorded under that value. This is very similar to the forward chaining algorithm in Chapter 7. See (?!) for detailed proofs.

5.11 The problem statement sets out the solution fairly completely. To express the ternary constraint on $A, B$ and $C$ that $A + B = C$, we first introduce a new variable, $AB$. If the domain of $A$ and $B$ is the set of numbers $N$, then the domain of $AB$ is the set of pairs of numbers from $N$, i.e. $N \times N$. Now there are three binary constraints, one between $A$ and $AB$ saying that the value of $A$ must be equal to the first element of the pair-value of $AB$; one between $B$ and $AB$ saying that the value of $B$ must equal the second element of the value of $AB$; and finally one that says that the sum of the pair of numbers that is the value of $AB$ must equal the value of $C$. All other ternary constraints can be handled similarly.

Now that we can reduce a ternary constraint into binary constraints, we can reduce a 4-ary constraint on variables $A, B, C, D$ by first reducing $A, B, C$ to binary constraints as