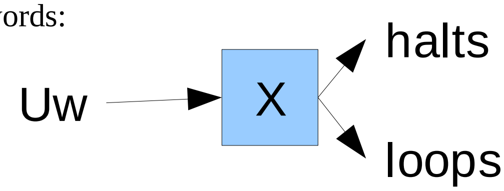
J Bolton ch. 12

Languages and Machines

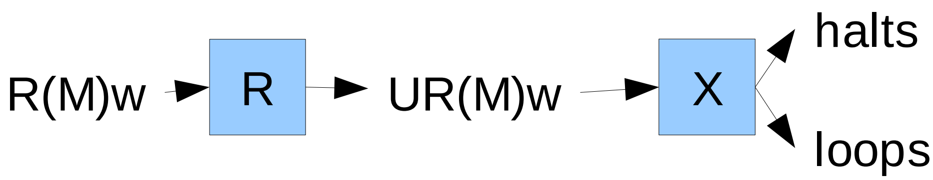
Chapter 12: problem 1, 4, 6, 10, 12, 13, 15, 16cd, 17cd, 19

1.

Assume there is a machine X that will decide whether the universal machine with input string w will halt. In other words:



Let us construct a reduction R that takes an input R(M)w and puts the universal machine U in front of it. As stated above, X runs the universal machine U on input w and halts if U halts and loops if U loops. In other words:



This is the form of the Halting Problem, and we have found our contradiction.

4.

We can reduce the Halting Problem to this problem. Let us assume we have an arbitrary machine A that solves the decidability problem in question. The input to our reduction R is the representation of a Turing Machine M followed by an input string w. The result of R is the representation of a machine M' that
1. Starts with the input 101 on the tape, and otherwise fails
2. Erases 101 from the tape and writes w
3. Runs M on w.
We thus have a computation with input R(M)w and is decidable. This is a contradiction of the proof of the halting problem

6.

Assume we have a machine X that is decidable on input machine R(M)qiw, which halts if M enters state qi on computation of w.

Let us define a reduction R that takes as input a machine of the form R(M)w. The reduction takes all the outgoing transitions of qi and removes them. It then takes all the undefined transitions and loops them for all the other states.

This way, transitions into qi will always halt, and transitions elsewhere will always loop. This is a reduction from the halting problem with input R(M)w to the state entry problem, which is our desired contradiction.

10.

Assuming that it is supposed to be {1}* and assuming that this is for an arbitrary turing machine, then we can assume that we have a machine X that takes as input R(M)1* and decides whether R(M) halts with input 1*. Let us define a reduction R that takes as input R(M) and produces a machine M' that erases its input and runs M on the blank tape. I think this only works if we're talking about arbitrary turing machines rather than a specific machine. We *can* say that a specific machine decides whether a string 1* is even, but not an arbitrary machine by the above proof.

12.

(a) To show that P is non-trivial, we simply need to show that P is satisfiable by some, but not all and not zero, languages. In other words, we need to show that one language satisfies P and one does not. The language L={w} satisfies P and the language L'={} does not.
(b) The language L={x} satisfies P while L={x}* does not.
(c) L = {x} satisfies P while L = {$a^ib^ic^i$} does not.
(d) The language L={0,1}* satisfies P while L ={a} does not.

13.

(a) If L is recursive, then there is a turing machine X that halts for all inputs from L. Let us design X so that it takes as input R(M), and runs M on R(M). This guarantees that X will halt for every input R(M) that is in the language. It will also halt on any malformed input. However, if R(M) is not in the language, then the machine will run M on input R(M) and not halt, which means X will also not halt.
(b) I have already described L's recursive enumerability above. If L is recursively enumerable, then it will halt on success and loop on failure. See (a) for details.

15.

(a) This Semi-Thue system will look exactly like the one from example 12.5.1, except that the 1's are 0's and the 0's are 1's.
(b) This computation assumes that there are loops on the final state for 0 and 1 as in the example 12.5.1
q0B01B
|- Bq101B
|- B0q21B
|- B01q2B
|- accept
[q0B01B]
=>[Bq001B]
=> [B0q11B]
=> [B01q1B]
=> [B01qR]
=> [B01qL]
=> [B0qL]
=> [BqL]
=> [qL]
=> [qf]

16
c.
aabbaaabb
aabbaaabb

d.

I cannot find a solution for this one that uses all the dominos. The following solution uses only 3 of them:

a	bba	ba
ab	b	aba

I'll show that this seems to be unsolvable using a tabbed decision tree.

```
a
ab
    aba
    ababa
        ababa
        ababaaba
            ababaa
            ababaabaab
                ababaaba
                ababaabaababa [endless loop]
                or
                ababaab
                ababaabaababa  [endless loop]
        or
        abab
        ababaaba
            ababa
            ababaabaab
                ababaa
                ababaabaabab
                    ababaaba
                    ababaabaabababa [endless loop]
                    or
                    ababaab
                    ababaabaababab [endless loop]
    or
    ab
    ababa
        aba
        ababaab
            ababa
            ababaababa [endless loop]
            or
            abab
```

```
                ababaababa [endless loop]
        or
        abba
        abb
            abbaba
            abbaba [solution without using all dominos]
            or
            abbab
            abbaba
                abbaba
                abbabaab
                    abbabaa
                    abbabaabab
                        abbabaaba
                        abbabaababaaba [endless loop]
    or
    bba
    b
        bba
        b
            bbabba
            bb
                bbabbaa
                bbab
                    bbabbaabba
                    bbabb [no more options]
```

17.

c.

Again, I'll use a tabbed decision tree where each leaf fails. I'm sorry if this isn't very readable, but it's fast to write out.

```
    ab
    aba
        ababa
        ababaa
            ababaaba
            ababaabaa [endless loop]
            or
            ababaab
            ababaaaba
                ababaabbaa
                ababaaabaaa [endless loop]
        or
        abab
        abaaba
            ababbaa
```

abaabaaa [endless loop]

d.
ab
abb
abbb
abbbab
abbbab
abbbababb
abbbabab
abbbababbabb [no options]
or
abbbabab
abbbababbbb
abbbababbb
abbbababbbbbab
abbbababbbbb
abbbababbbbbabbab
abbbababbbbbab
abbbababbbbbabbababb [no options]
or
abbbababbbbbab
abbbababbbbbabbabbb [no options]
or
abbbab
abbbabbb
abbbabbb
abbbabbbbab [no options]

19.
(a)

b	babbb	bab	ba
bbb	ba	aab	a

solution:

babbb	b	b	ba
ba	bbb	bbb	a

Like 17 d, I'm assuming we're not required to use all the dominos; otherwise, there is no solution.
(b)
G_U:
Su → bSu1 | b1
→ babbbSu2 | babbb2
→ babSu3 | bab3
→ baSu4 | ba4
Sl → bbbSl1 | bbb1

→ baSl2 | ba2
→ aabSl3 | aab3
→ aSl4 | a4

©

Su
=> babbbSu2
=> babbbbSu12
=> babbbbbSu112
=> babbbbbba4112

Sl
=> baSl2
=> babbbSl12
=> babbbbbbSl112
=> babbbbbba4112