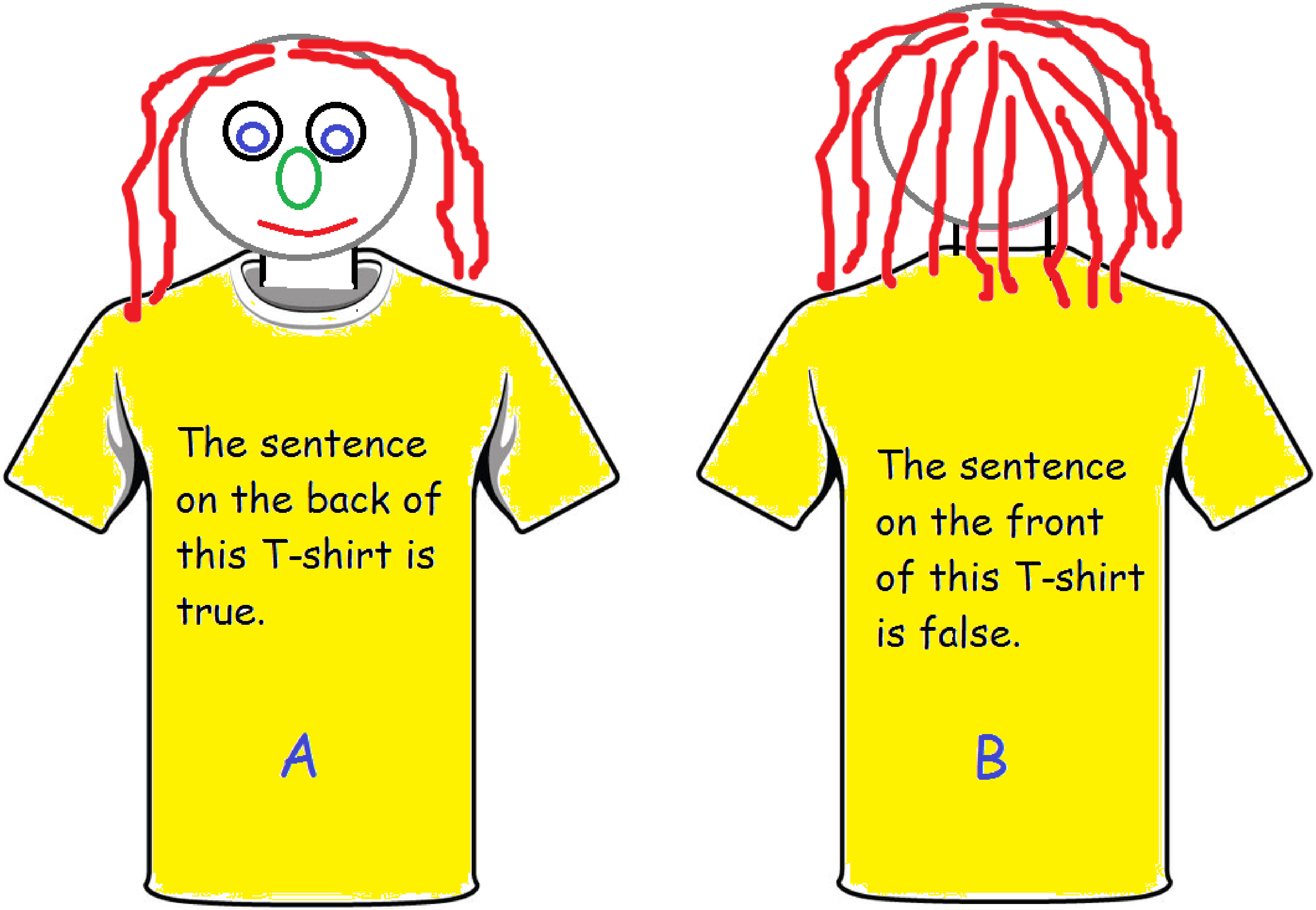
Argue that the following languages are Turing-decidable:

(a) {"M" "w" : M halts on w after at most 700 steps}

(b) {"M" "w": M halts on w without using more than the first 100 tape squares}

For part (b): we only have to simulate M for a finite number of steps and in that time frame it will either halt, hang, use more than the first 100 tape squares, or it will never halt. How many steps must we run M for?


The sentence on the back of this T-shirt is true.
A
The sentence on the front of this T-shirt is false.
B

Proving Problems Undecidable

To prove problem Q is not decidable:

1. Select a problem P known to not be decidable.

2. Prove that if you can solve Q you can solve P.

3. This is a contradiction since P is not decidable and therefore Q is not decidable.

Let H_1 = {"M": M halts on input "M"}.

A string "M":

(q01,a0010,q01,a0000),(q01,a0100,q10,a0000),
(q10,a0101,q00,a0001),(q10,a0110,q10,a0110) $\in H_1$

State	Sym.	Next State	Head
s	#	s	L
s	)	t	L
t	0	h	R
t	1	t	1

Let H_1 = {"M": M halts on input "M"}.

Change one transition so it does not halt:

(q01,a0010,q01,a0000),(q01,a0100,q10,a0000),
(q10,a0101,q10,a0101),(q10,a0110,q10,a0110) $\notin H_1$

State	Sym.	Next State	Head
s	#	s	L
s	)	t	L
t	0	t	0
t	1	t	1

q00a())01aaq01 $\notin H_1$

$\varepsilon \notin H_1$

H_1 = {"M": M halts on input "M"}

The complement of $H_1 = \overline{H_1}$

= {"M" : M does not halt on input "M"}

U { w: w ≠ "M" for any TM M}

Theorem: $\overline{H_1}$ is not Turing-acceptable.

Proof: Assume that TM M' accepts L.

What does TM M' do on input "M'"?

We are assuming M' accepts $\overline{H_1}$.

$\overline{H_1}$ = {"M" : M does not halt on input "M"}

U { w: w ≠ "M" for any TM M}

Case 1: M' halts on "M'".

But then "M'" is not in $\overline{H_1}$ so M' should not halt- contradiction.

Case 2: M' does not halt on "M'".
But then "M'" is in $\overline{H_1}$ so M' should halt- contradiction.

H_1 = {“M“: M halts on input “M”}

$\overline{H_1}$ = {“M“ : M does not halt on input “M”}

∪ { w: w ≠ “M“ for any TM M}

Implications:

1. $\overline{H_1}$ is not Turing-acceptable.

2. $\overline{H_1}$ is not Turing-decidable.

3. H_1 is not Turing-decidable.

Known: H_1 = {"M": M halts on input "M"} is not Turing decidable.

Theorem 5.4.2 (p. 255): The following problems are not Turing-decidable:

(a) Given M_a, w: Does M_a halt on input w?

(b) Given M_b: Does M_b halt on input ε?

(c) Given M_c: Is there any string on which M_c halts?

(d) Given M_d: Does M_d halt on every string?

(e) Given two TM's M_1 and M_2: Do they halt on the same input strings?

Halting problem reductions:

Known: Problem P is not decidable. (HARD)

This means that it is impossible to design a TM or a program (e.g. in C or Java) that can solve the problem P.

New problem Q: (EASY?)

To show there is no algorithm for Q, show that if there is one, it could be used to solve P. [Proof by contradiction].

Each question has an equivalent language membership formulation:

(a) Given a TM M_a and string w, does M_a halt when started on input w?

L_a = {"M_a" "w" : M_a halts on input w}

(b) Given a TM M_b, does M_b halt when started on a blank tape?

L_b = {"M_b" : M_b halts on input ε}

Given that (a) has no algorithm:

(a) Given a TM M_a and string w, does M_a halt when started on input w?

prove that (b) has no algorithm:

(b) Given a TM M_b, does M_b halt when started on a blank tape?

The proof is a proof by contradiction. Assume that there is an algorithm for problem (b). We show that it then would be possible to design an algorithm for problem (a). This is a contradiction since it is not possible to design an algorithm for (a).

Summary: Closure

Operation	Turing-decidable	Turing-acceptable
Union	yes	yes
Concatenation	yes	yes
Kleene star	yes	yes
Complement	yes	no
Intersection	yes	yes

L= { "M" : there is some string w in {a,b}* such that M halts on input w}

Theorem: L is Turing-acceptable.

Proof: Example of dovetailing.

Suppose I construct a TM M_f which:

1. Preserves its input u.

2. Simulates a machine M_b on input ε.

3. If M_b hangs on input ε, force an infinite loop.

4. If M_b halts on input ε, then run a UTM on the original input u which forces an infinite loop if u is not "M" for any TM M, and otherwise it runs M (u= "M") on input ε. If M hangs on input ε, the simulating UTM also hangs.

What language does this machine M_f accept in each of two cases:

Case 1: When machine M_b does not halt on input ε.

Case 2: When machine M_b halts on input ε.