Lecture 5

Frank

February 15, 2022

1 More Examples of Using Reductions to Prove Undecidability/Unrecognizability

1.1 Example 1

Consider language

$$HET = \{\langle M \rangle : M \text{ halts on empty tape } \}$$

Thm 5.1: HET is not decidable

Proof: Suffices to show that $U \leq_m HET$

Given $\langle M, x \rangle$, construct $\langle M' \rangle$ s.t. M accepts $x \iff M'$ halts on empty tape.

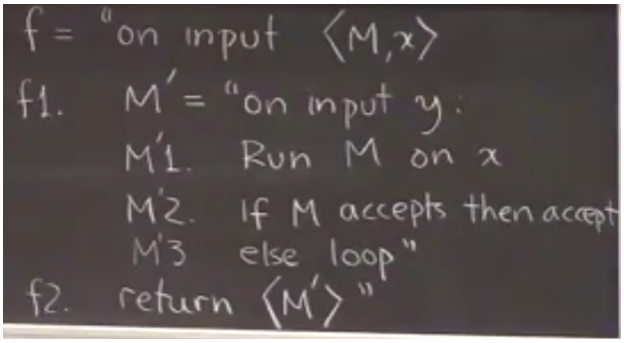

Figure 1: l51

M accepts x
$\implies$ M' accepts $\Sigma^\star$
$\implies$ M' accepts ϵ
$\implies$ M' halts on empty tape
$\implies$ $\langle M' \rangle \in HET$

M' doesn't accept x
$\implies$ M' always loops
$\implies$ M' does not halt on empty tape
$\implies$ $\langle M' \rangle \notin HET$

Exercises: HET is recognizable and determine if $H\bar{E}T$ i s recognizable/decidable

1.2 Properties of Mapping Reduction

Consider

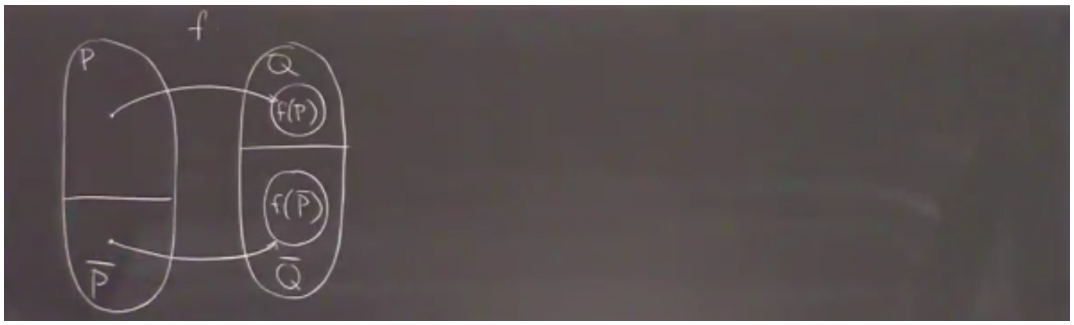

Figure 2: l52

Note that the mapping does not have to be (and usually isn't) onto.

For our previous example, this means we did not construct **all** TMs that halts on empty tape, only a particular set of TMs with that property, namely the TMs that accepts everything or never halts.

This means our mapping does not have to create a solution for the entire problem Q, but some special case of the problem (to our convenience)

1.3 Example 2

Consider language $ODD = \{\langle M \rangle : L(M) \text{ is finite and } |L(M)| \text{ is odd}\}$

Thm 5.2: ODD is not recognizable

Proof: Suffices to show $\bar{U} \leq_m ODD$

Given $\langle M, x \rangle$ [input to $\bar{U}$], construct $\langle M' \rangle$ [input to ODD] s.t.

M does not accept $x \implies L(M')$ is finite and $|L(M')|$ is odd
M accepts $x \implies L(M')$ is infinite or $|L(M')|$ is even

Note that there are many ways to construct M'; make M' only accept an odd number of strings if M does not accept x, and make M' accept an even number or infinite number of strings if M accepts x

Here is one solution

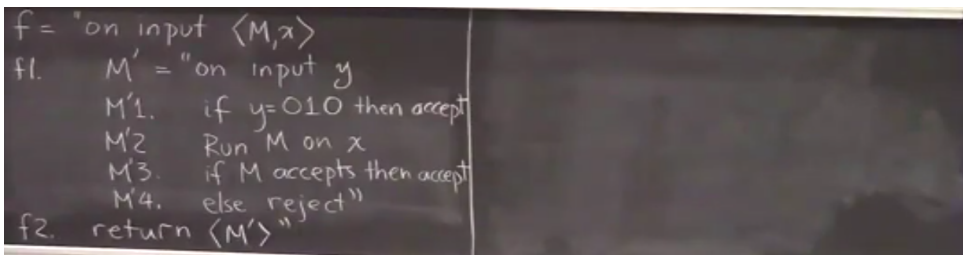

Figure 3: l53

Verify that:

If M does not accepts x then $L(M') = \{010\} \in ODD$

If M accepts x then $L(M') = \Sigma^\star \notin ODD$

Exercise: Prove that $U \leq_m ODD$ then place $ODD \& O\bar{D}D$ on map.

1.4 Example 3

Let language $FIN = \{\langle M \rangle : L(M) \text{ is finite } \}$ and $INF = F\bar{I}N$

Thm 5.3: FIN and INF are not recognizable

Proof: Suffices to show: a) $U \leq_m FIN$ and b) $U \leq_m INF$
(this means that $\bar{U} \leq_m FIN$ and $\bar{U} \leq_m INF$ [Thm 4.5])

a)

Given $\langle M, x \rangle$ [input to U] constuct $\langle M' \rangle$ [input to FIN] s.t.

If M accepts x then $L(M')$ is finite
If M does not accept x then $L(M')$ is infinite

Define f as

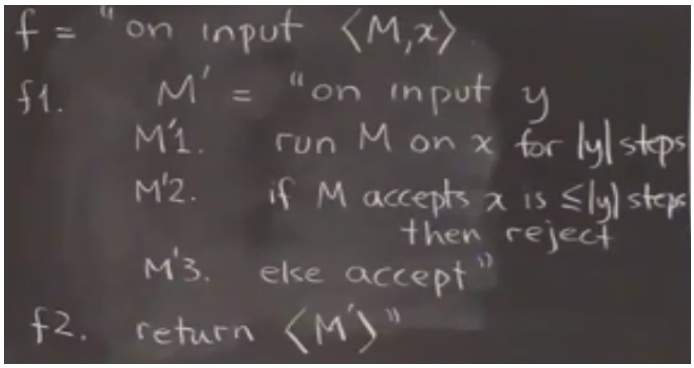

Figure 4: l54

Verify $\langle M, x \rangle \in U \iff \langle M' \rangle \in FIN$

$(\implies)$ $\langle M, x \rangle \in U$

$\implies$ M accepts x
$\implies$ $\exists k$ s.t. M accepts x after k steps
$\implies$ M' accepts y if $|y| < k$
M' rejects y if $|y| \geq k$
$\implies$ $L(M') = \{y : |y| < k\}$ (this is finite)
$\implies$ $\langle M' \rangle \in FIN$

$(\impliedby)$ $\langle M, x \rangle \notin U$

$\implies$ M does not accept x
$\implies$ $\forall k$, M does not accept x in k steps
$\implies$ $\forall y \in \Sigma^\star$, M' accepts y in line $M'3$
$\implies$ $L(M') = \Sigma^\star$ (this is infinite)
$\implies$ $\langle M' \rangle \notin FIN$

b)

Given $\langle M, x \rangle$ [input to U] construct $\langle M' \rangle$ [input to INF] s.t.

If M accepts x then M' accepts an infinite language [$\Sigma^\star$]
If M does not accept x then M' accepts a finite language [$\emptyset$]

Easily done.

1.4.1 Aside (Problem with reducing a))

Note that our M' typically has this form

1. (Maybe) accept certains y's
2. Run M on x
3. If M accepts then (maybe) accept more y's

Note: We cannot count on M rejecting because M might loop on x

This means

$$L(M') = \begin{cases} y's \text{ accepted in 1. or 3.} & \text{if } M \text{ accepts } x \\ y's \text{ accepted in 1.} & \text{if } M \text{ doesn't accept } x \end{cases}$$

Note the subset relationship between the $y's$ accepted if M accepts/doesn't accept x. Lower is a subset of the upper.

This applies well in b), but in a), there is no way an infinite language is a subset of a finite language

2 Post's correspondence problem

2.1 Problem

- Instance of the problem:

  Sequence $(x_1, y_1), \ldots, (x_k, y_k)$ where $x_i, y_i \in \Gamma^\star$ and $|\Gamma| \geq 2$

Figure 5: l61x

- Question: Does there exist a sequence, $1 \leq i_1, i_2, \ldots, i_m \leq k$
  s.t. $x_{i_1} x_{i_2} \ldots x_{i_m} = y_{i_1} y_{i_2} \ldots y_{i_m}$

  Solution to above instance:

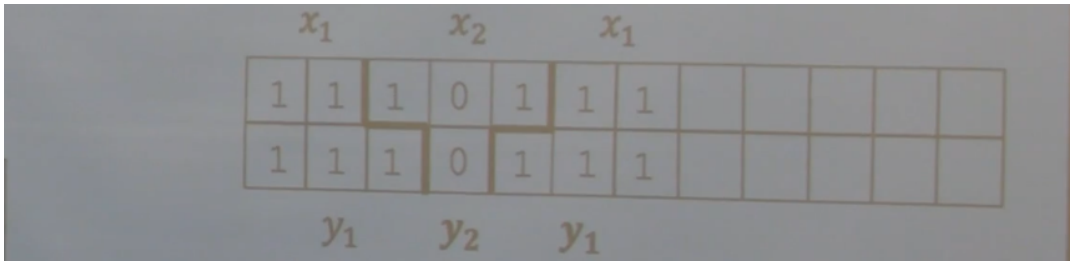

Figure 6: l62

- There is no TM that can take any instance of PCP and decide if it is solvable or not

- Another instance of PCP:

i	x_i	y_i
1	11	111
2	101	1
3	10	01
4	0	01

Figure 7: l63

Observations about this instance:

- Pair 2 cannot be used, because we would otherwise always have more 0's in the top string than in the bottom string.
- Given that pair 2 cannot be used, we cannot use pair 1 either, because pair 2 is the only pair that allows us to "replenish" the defecit of 1's present in pair 1 (1 less 1's in the top string)
- Pair 4 is not usable for the exact same reasons
- Pair 3 is the only string we can use, and clearly, there is no solution to this instance of the problem

2.2 Result to be proven

- **Thm 5.6**: PCP is undecidable (but recognizable)
  - Recognizability is not proven here
  - Guess solution and verify it works, or dovetail finite sequences and accept as long as one works

Proof: attempt to reduce U to PCP. Will use modified PCP (MPCP) - PCP with additional requirement $i_1 = 1$ (any solution to the PCP must have pair 1 as its first pair - solutions are not unique b/c we can repeat a solution infinite amount of times)

Note that the first instance of PCP shown above is an instance of MPCP.

Will show that 1) $U \leq_m MPCP$ and 2) $MPCP \leq_m PCP$

2.2.1 1)

- $U \leq_m MPCP$:

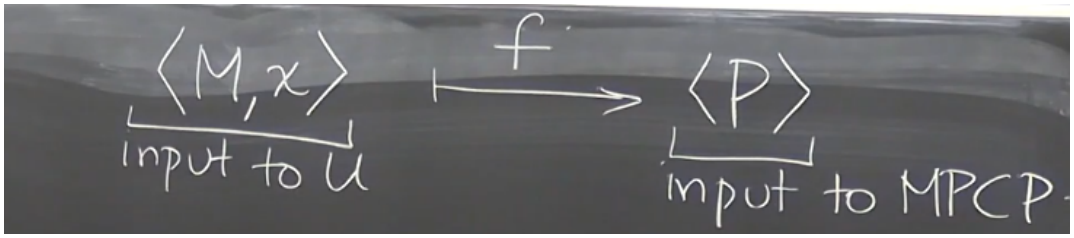

Figure 8: l64

s.t. M accepts $x \iff P$ has a solution

$(\implies)$

M accepts x means: There is a sequence of configurations $C_1 \vdash C_2 \vdash \ldots \vdash C_l$ where $C_1 = q_0x$ and $C_l = yq_Az$. And assume WLOG M never writes $\sqcup$ (assume that it writes another symbol in place of $\sqcup$)

- Partial solution to MPCP: $1 \leq i_1, i_2, \ldots, i_m \leq k$ s.t. $x_{i_1} \ldots x_{i_m}$ is prefix to $y_{i_1} \ldots y_{i_m}$ (top string is a prefix of the bottom string)
- **Intuition**:
  1. Top and bottom strings in any partial solution of P will be sequences of configs of M separated by #

     $\#C_1\#C_2\#\ldots$ where C_1 is the initial config q_0x and $C_i \vdash C_{i+1}$
  2. Top string is one configuration "behind" bottom

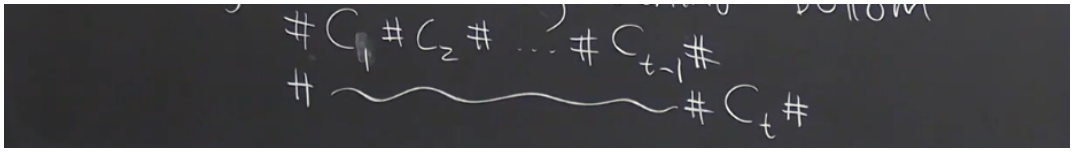

Figure 9: l65

     until C_t has the accept state.
  3. If that happens, P will contain pairs (x_i, y_i) to allow top string to "catch up" (Full solution emerges)

Note that we are picking the pairs to be in our P so that whenever M accepts x or that M has an accepting configuration on x, we have the pairs necessary to "fulfill" our MPCP problem.

We construct the solution to the MPCP as described below

- Grouping pairs of strings in P into categories

  (I) [First pair] is $(\#, \#q_0x\#)$

  (II) [Copy pairs] is (a, a) plus $(\#, \#)$ $\forall a \in \Gamma - \{\sqcup\}$ where Γ is the tape alphabet of M

  Consider the situation $C_i \vdash C_{i+1}$. These two configs differ very little except for the area where the state symbol is.

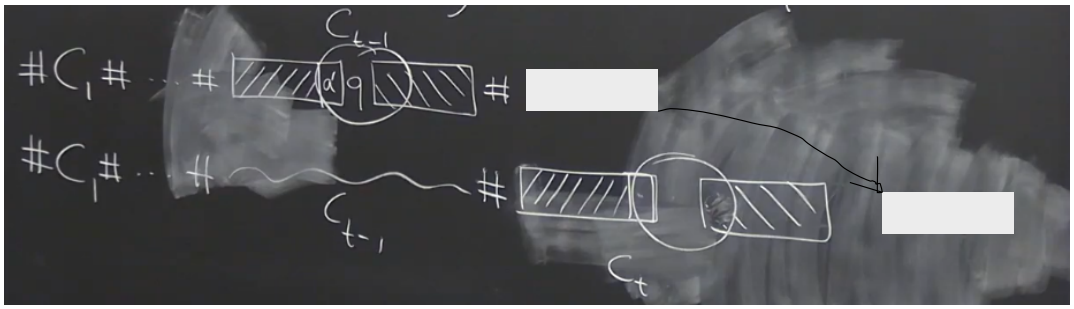

Figure 10: l66

Thus, the function of the copy pairs is to allow us to transfer the stuff that "doesn't change"

(III) [Modify pairs]

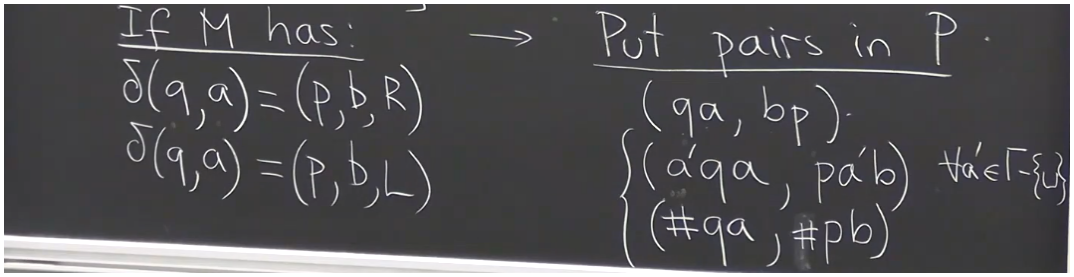

Figure 11: l67

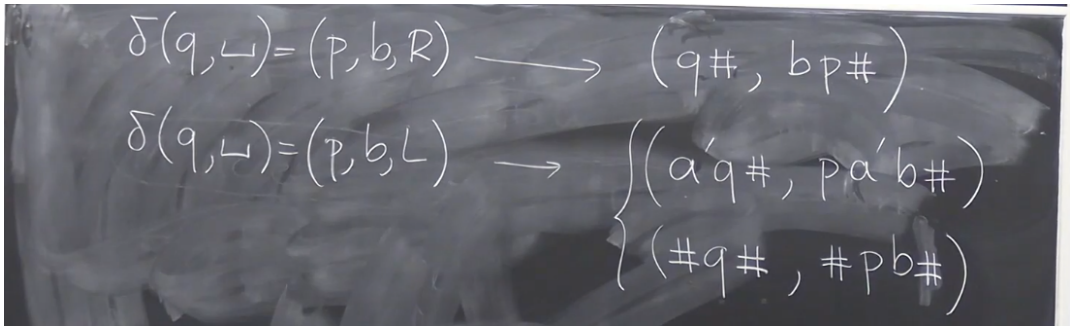

Figure 12: l68

Deals with the different cases of the transition function

(IV) [Catch-up pairs] is (aq_A, q_A) and (q_Aa, q_A)

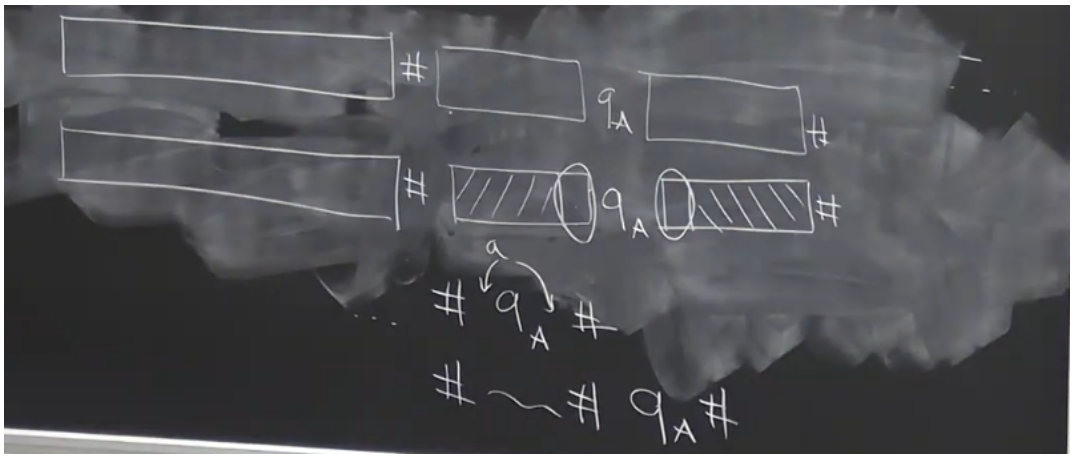

Figure 13: l69

The idea is to slowly eat away at either sides of the accepting state symbol q_A on the bottom string. If we have these pairs, we can eventually reach a state where both strings differ by exactly $q_A\#$ at the end.

(V) [Final Pair] is $(q_A\#\#, \#)$, which allows us to complete the construction of the top and bottom strings s.t. they are the same.

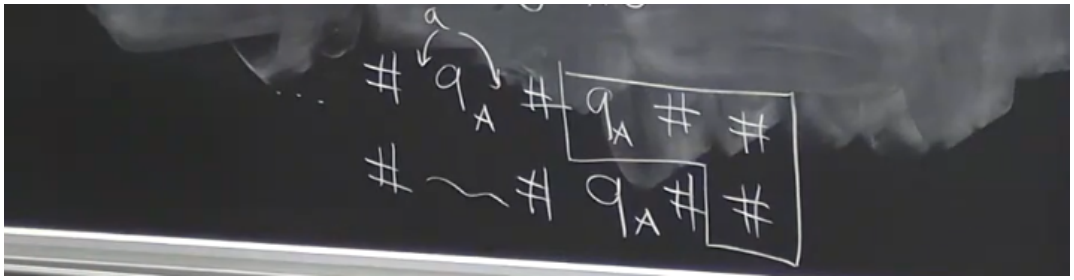

Figure 14: l610

Summary:

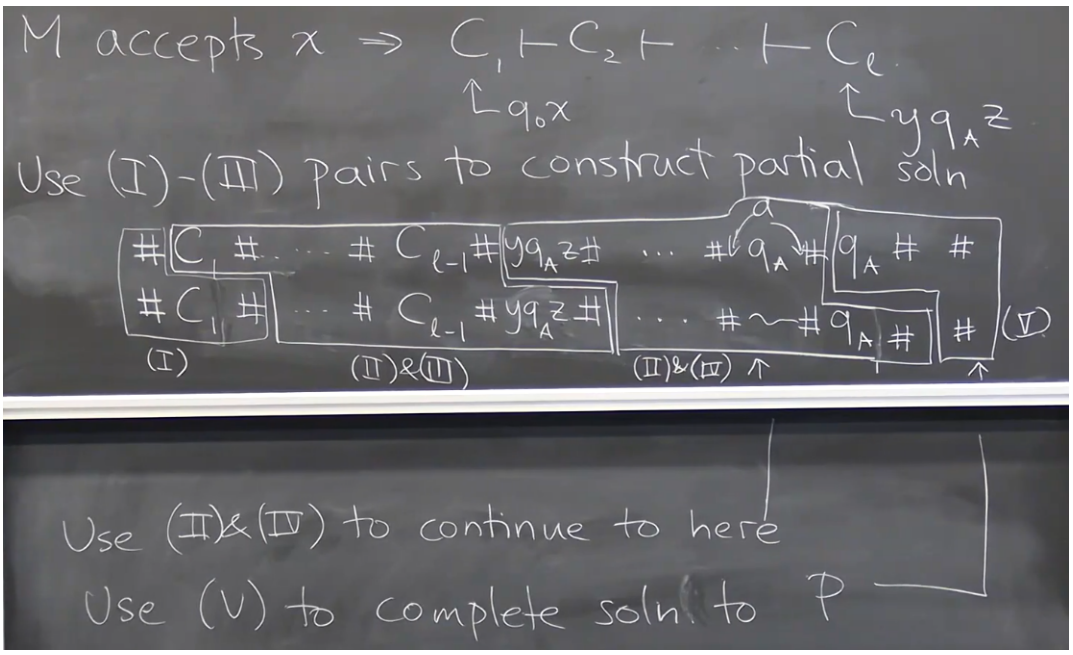

Figure 15: l611

Note that the pairs are always zig-zag'ed, so that when we use (III) to simulate changing symbols, we will not mess up the MPCP solution. We can then use "catch up" (IV) pairs to eventually "fill up" the top string and allow it to catch up to the bottom string. Then we will use (V) to close the gap completely

($\Longleftarrow$) Invariants of partial solutions to MPCP P

- Top and bottom strings of any proper partial solution are of the form:

Figure 16: l612

  where $C_1, \ldots, C_{t-1}$ are configs of M and C'_t is a prefix of a config of M
- if state of C_{t-1} is not q_A then C'_t is a prefix of some C_t s.t. $C_{t-1} \vdash C_t$
- if state of C_{t-1} is q_A then so is the state of C'_t (if there is one)

Proved by straightforward but tedious induction.

- Fastforward, the rest of the argument is:
  - Look at shortest prefix of solution that ends with ## (must exist because we must use final pair at some point)

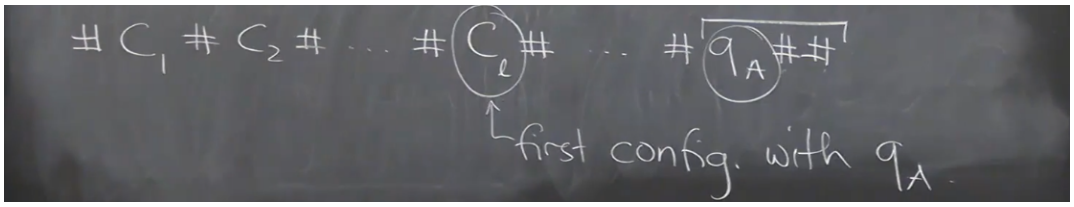

Figure 17: l613

Then look for the first config with q_A

Then by the invariants above, we must have an accepting configuration

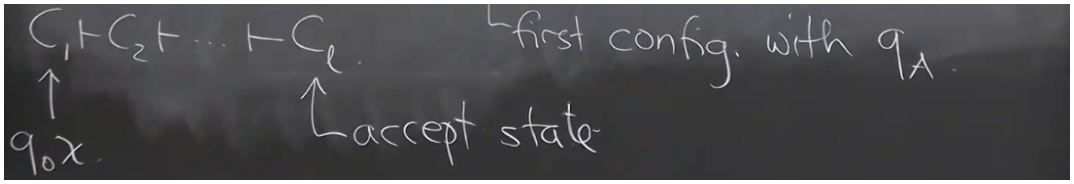

Figure 18: l614

Thus, M accepts x

2.2.2 2)

$MPCP \leq_m PCP$

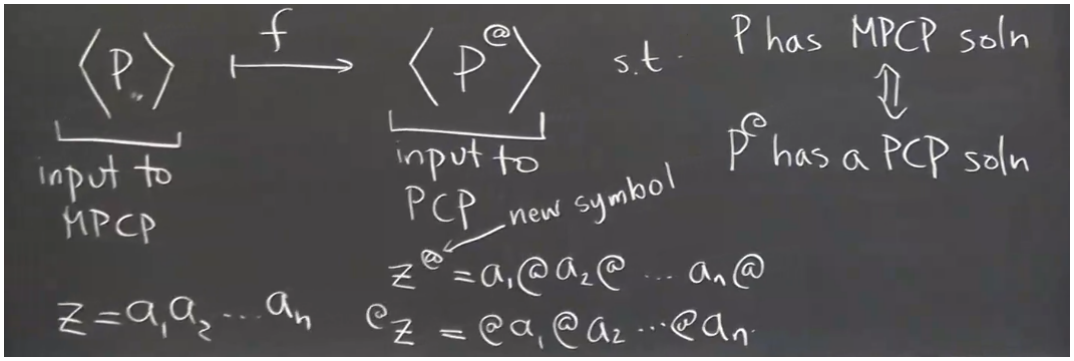

Figure 19: l615

$(\implies)$

Where $P^{@}$ has pairs $(v_1, w_1), \ldots, (v_k, w_k)$ and $v_i = x_i^{@}$, $w_i =^{@} y_i$ s.t. (x_i, y_i) are pairs in P

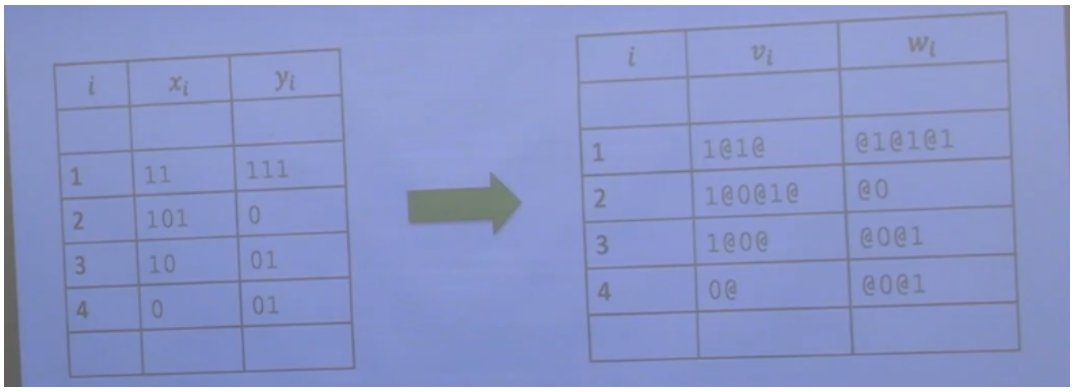

Figure 20: l616

Then we end up with a very perturbed matching

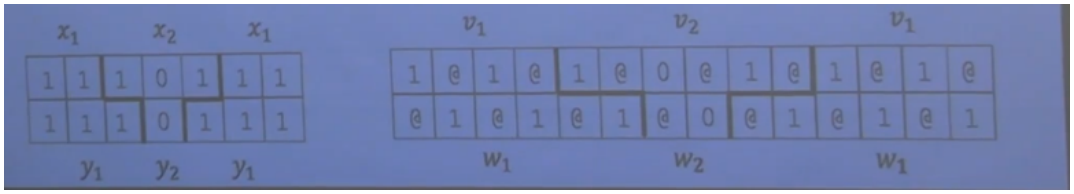

Figure 21: l617

However, looking carefully, we only have to shift the strings and fill in some gaps with 2 additional pairs at the start and finish

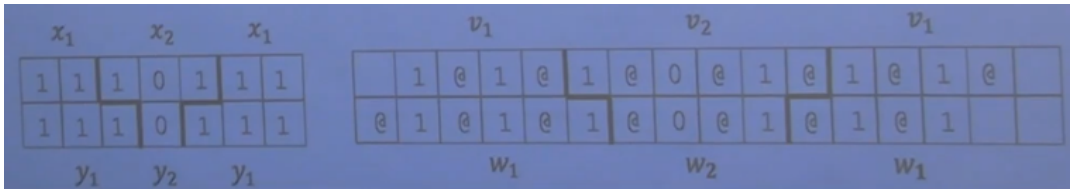

Figure 22: l618

$(v_0, w_0) = (@v_1, w_1)$ and $(v_{k+1}, w_{k+1}) = (\$, @\$)$

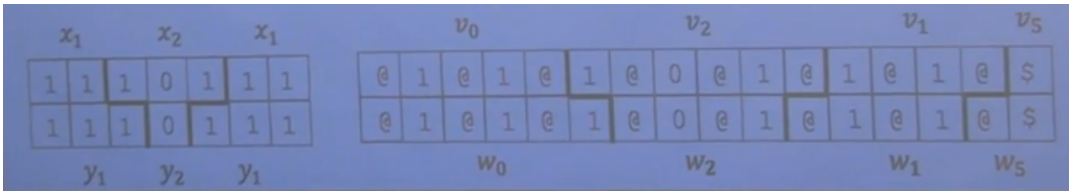

Figure 23: l619

The above diagram is actually a bit off, but you get the idea.

($\Longleftarrow$)

P has MPCP solution $x_1 x_{i_2} x_{i_3} \dots x_{i_m} = y_1 y_{i_2} y_{i_3} \dots y_{i_m}$

$\implies v_0 v_{i_2} \dots v_{i_m} v_{k+1} = w_0 w_{i_2} \dots w_{i_m} w_{k+1} \implies P^{@}$ has a solution

Using this, we can conclude that $P^{@}$ has PCP solution $v_{i_1} v_{i_2} \dots v_{i_m} = w_{i_1} w_{i_2} \dots w_{i_m}$ where $v_{i_1} = v_0$, $v_{i_m} = v_{k+1}$, $w_{i_1} = w_0$, $w_{i_m} = w_{k+1}$

$\implies x_1 x_{i_2} \dots x_{i_{m-1}} = y_1 y_{i_2} \dots y_{i_{m-1}}$ - essentially stripping away the @ symbols (reversing the process)

$\implies P$ has MPCP solution