Turbo Codes : Decoder

Block diagram : Decoder With Puncturing

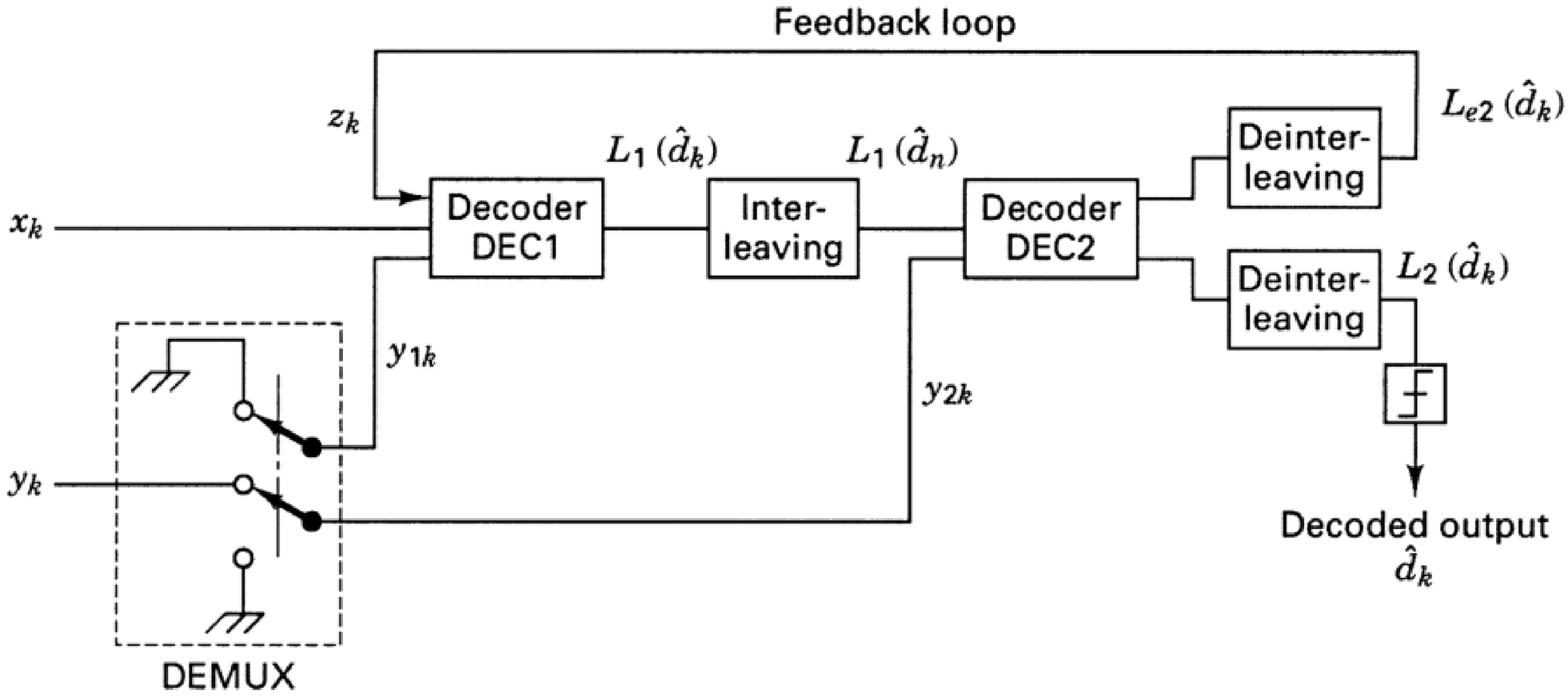
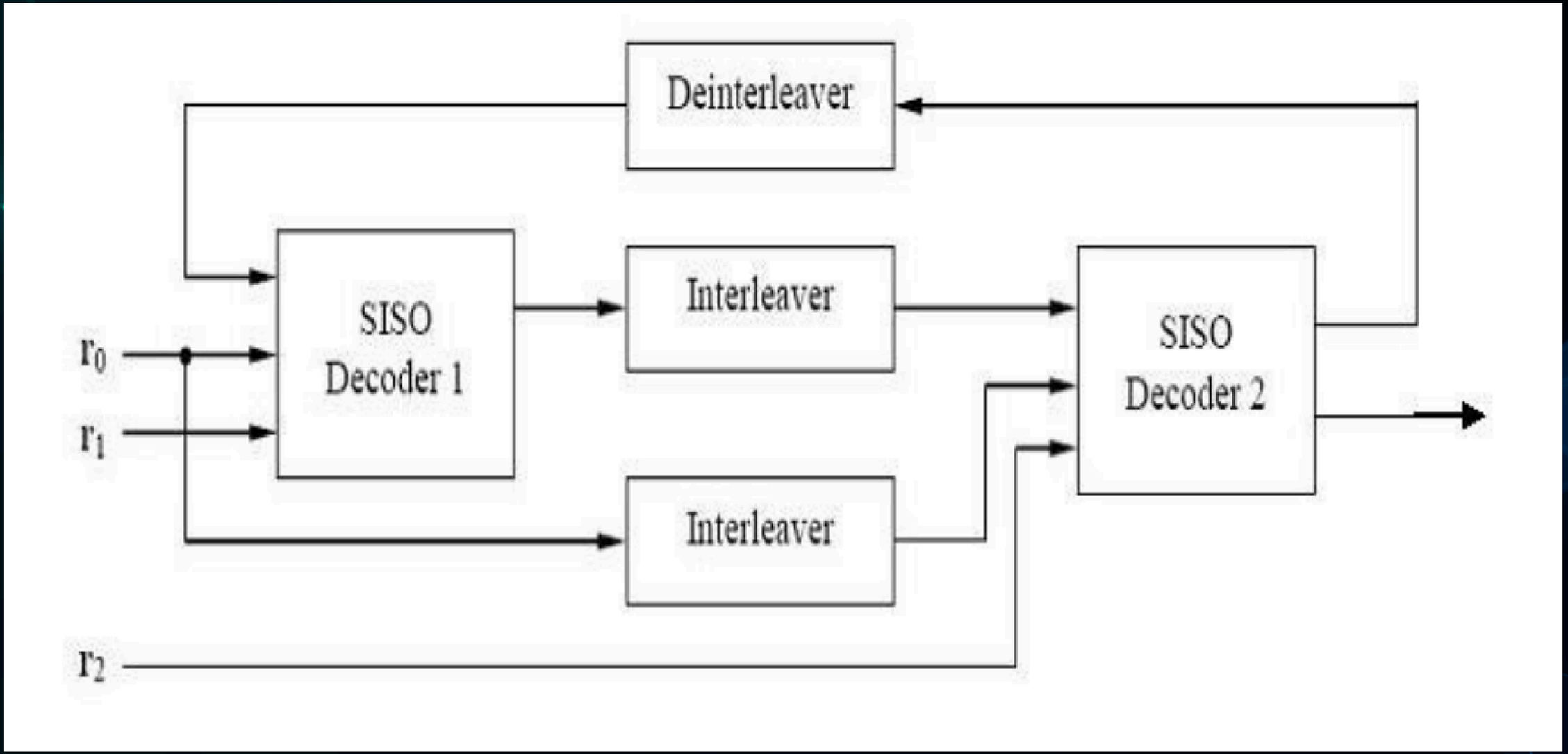


Figure 8

Feedback decoder.

Decoder block diagram without puncturing



REASONS FOR TURBO CODE'S AWESOME PERFORMANCE

- weight : no of 1's in a message
- low weight messages doesn't usually give low weight codewords
- size of codewords are pretty large gt 500
- turbo decoder has a feedback loop, similar to turbo charger in automobiles, info gets passed from one decoder to the other as soft outputs. i,e, llr's are sent.
- finally, after some iterations/ convergence criterion, we end the loop and take the output

BCJR ALGORITHM

Maximum A Posteriori(MAP) Decoding Algorithm

BCJR algorithm rate-$\frac{1}{2}$, zero-terminated convolutional encoder memory, $\mu = 2$

$\underline{u} = [u_0 \; u_1 \; \ldots \; u_{k-1}]$

rate - $\frac{1}{2}$

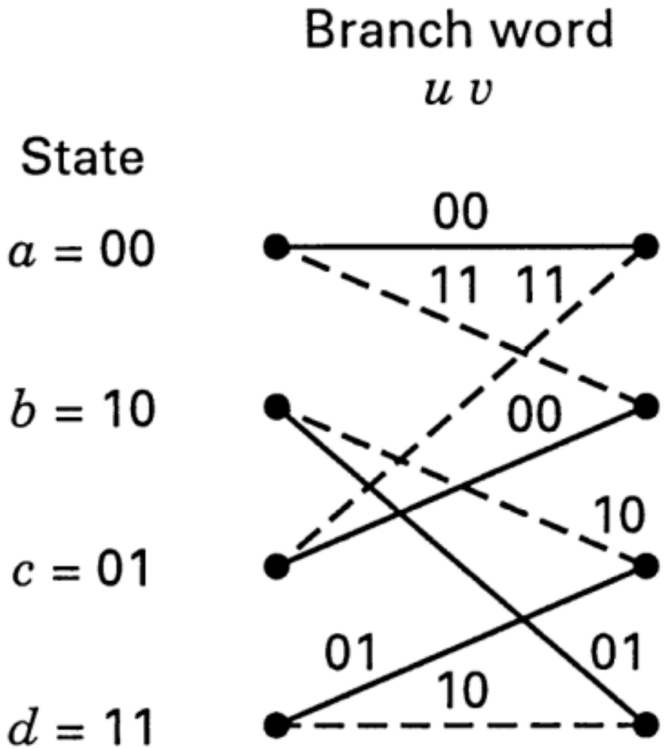
$\underline{v} = [v_0^{(1)} \; v_0^{(2)} \; v_1^{(1)} \; v_1^{(2)} \; \ldots \; v_{k+1}^{(1)} \; v_{k+1}^{(2)}]$

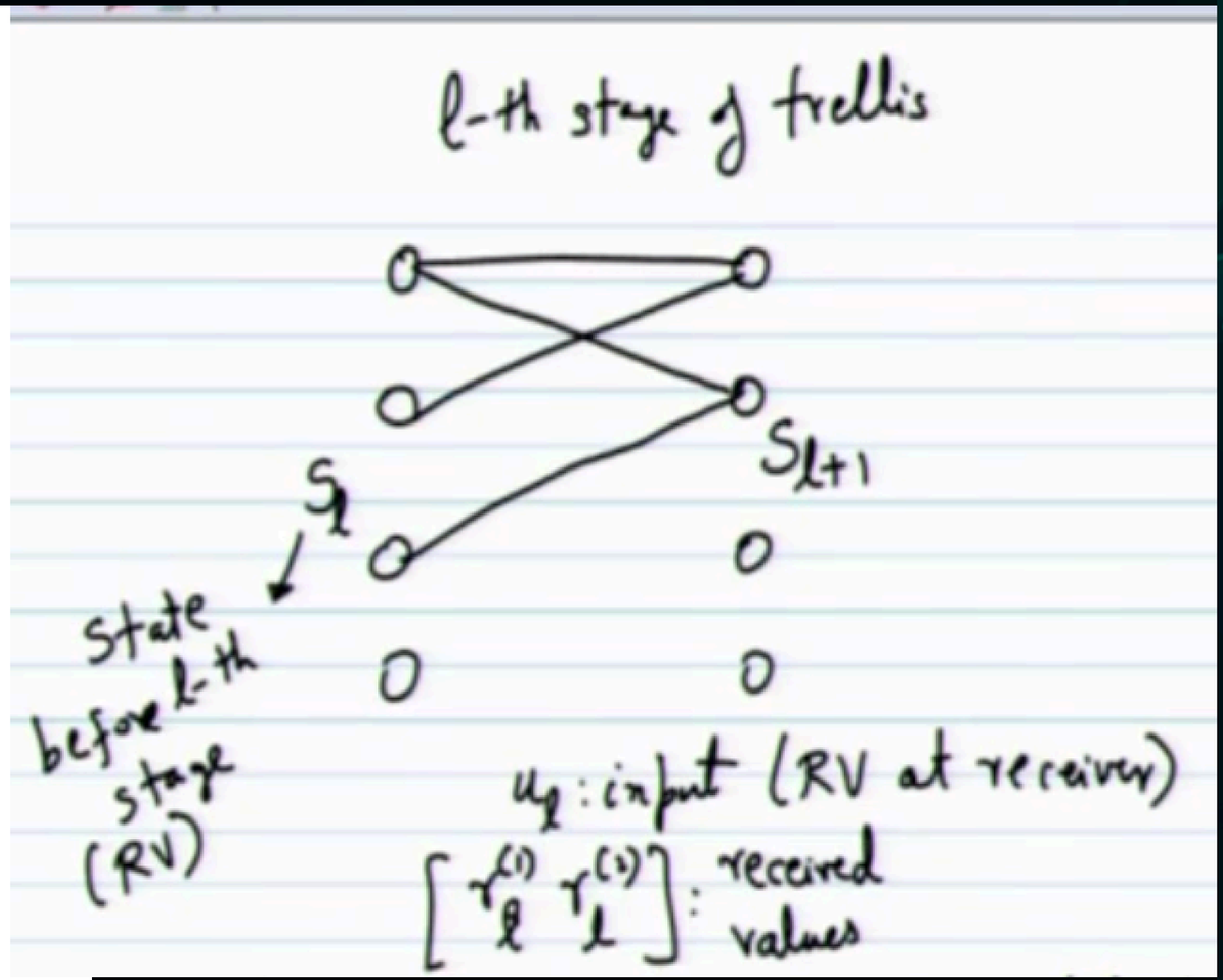
BPSK

$\underline{r} = [r_0^{(1)} \; r_0^{(2)} \; \ldots \; r_{k+1}^{(1)} \; r_{k+1}^{(2)}]$

$\underline{y} = [y_0^{(1)} \; y_0^{(2)} \; \ldots \; y_{k+1}^{(1)} \; y_{k+1}^{(2)}]$

SOME MORE NOTATION



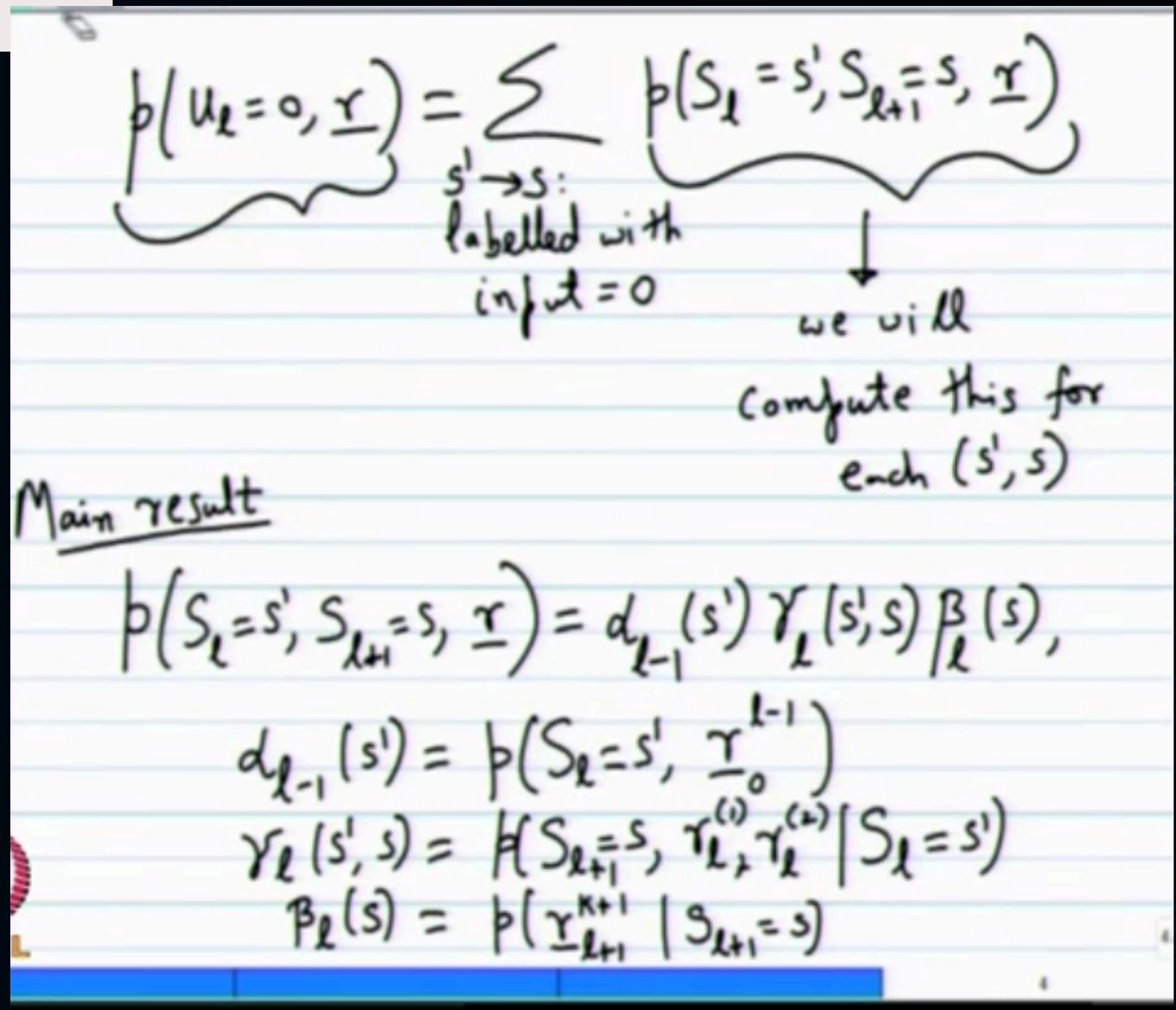


THE RESULT

$$P(u_l = +1|\mathbf{r}) = \frac{p(u_l = +1, \mathbf{r})}{P(\mathbf{r})}$$

HERE THE DENOMINATOR HAS NOT BEEN WRITTEN AS WE ARE ANY WAY GOING TO COMPUTE LLR WHERE THE DENOMINATOR GETS CANCELLED

ALPHA'S AND BETA'S GET CALCULATED BY RECURSION GAMMA'S COULD BE DIRECTLY CALCULATED



DERIVATION

Pf:

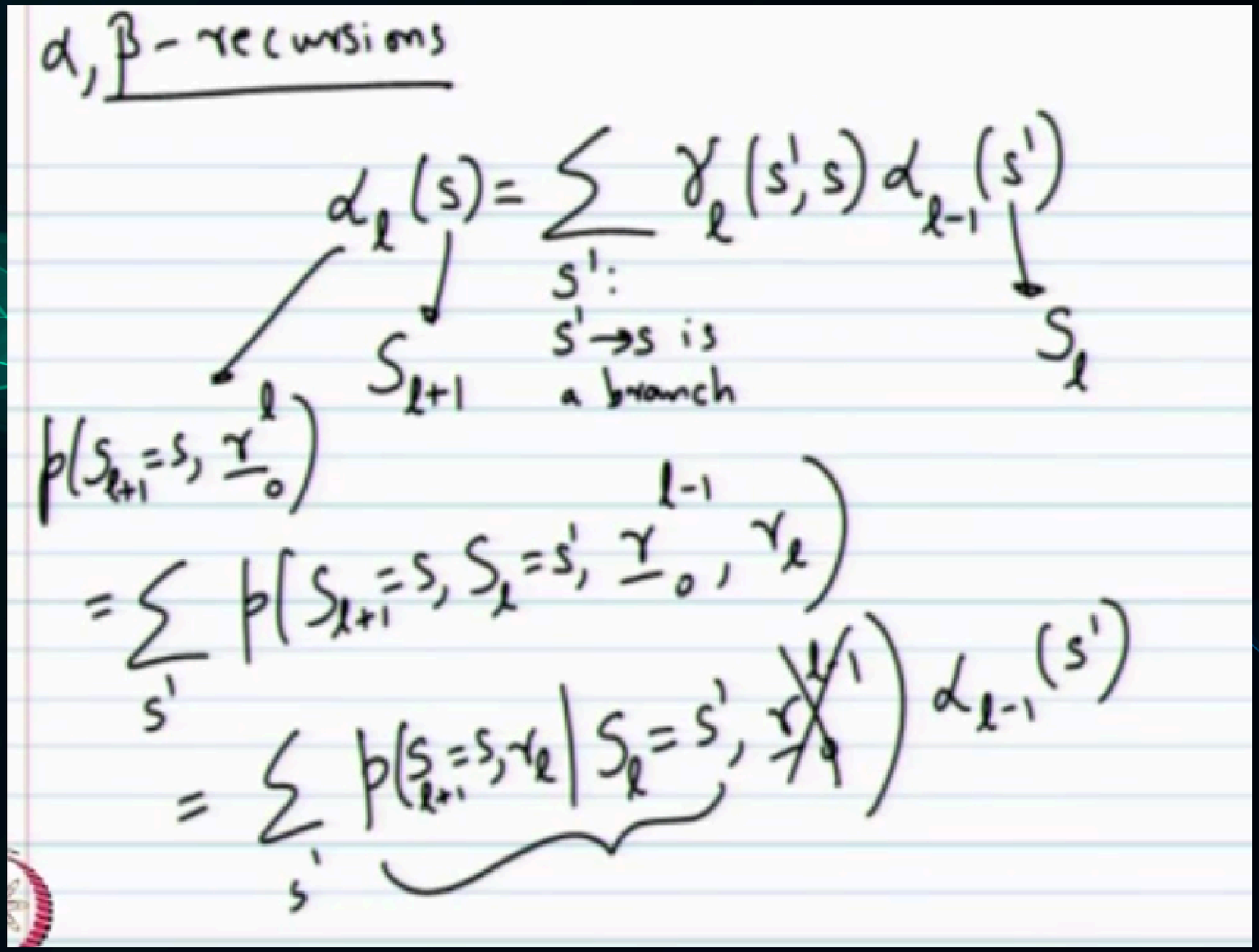
$$p(S_l = s', S_{l+1} = s, \underline{r}) = p(s', s, \underline{r}_0^{l-1}, r_l, \underline{r}_{l+1}^{k+1})$$

$$= p(\underline{r}_{l+1}^{k+1} \mid s', s, r_l, \underline{r}_0^{l-1}) \cdot p(s', s, r_l, \underline{r}_0^{l-1})$$

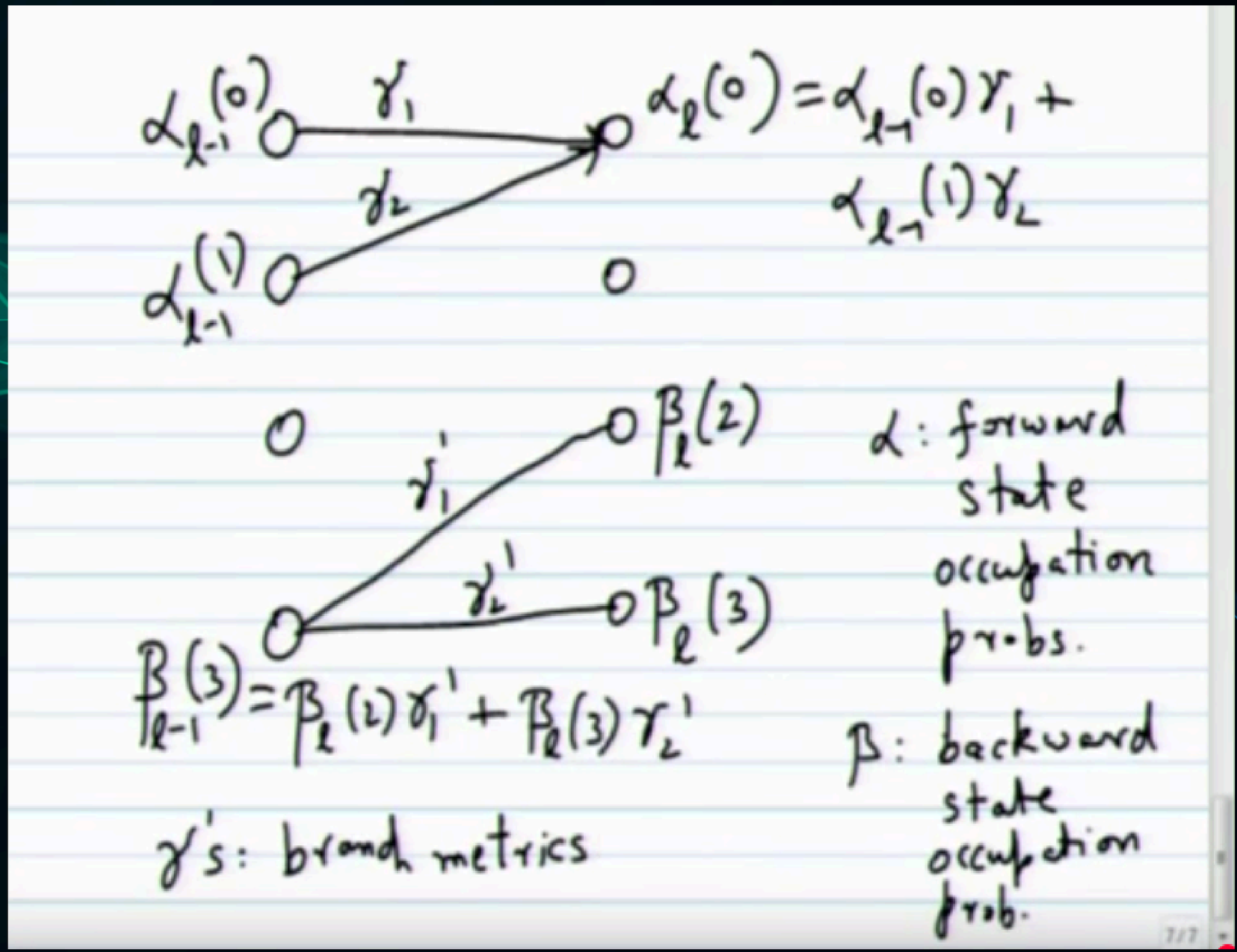
$$= p(\underline{r}_{l+1}^{k+1} \mid s) \cdot p(s, r_l \mid s', \underline{r}_0^{l-1}) \cdot p(s', \underline{r}_0^{l-1})$$

$$= p(S_l = s', \underline{r}_0^{l-1}) \cdot p(S_{l+1} = s, r_l \mid S_l = s') \cdot p(\underline{r}_{l+1}^{k+1} \mid S_{l+1} = s)$$

RECURSION


α, β - recursions
$\alpha_l(s) = \sum_{s'} \gamma_l(s', s)\, \alpha_{l-1}(s')$
s': $s' \to s$ is a branch
s_{l+1}
s_l
$p(s_{l+1}=s, \underline{r}_0^l)$
$= \sum_{s'} p(s_{l+1}=s, s_l=s', \underline{r}_0^{l-1}, r_l)$
$= \sum_{s'} p(s_{l+1}=s, r_l \mid s_l=s', \underline{r}_0^{l-1})\, \alpha_{l-1}(s')$

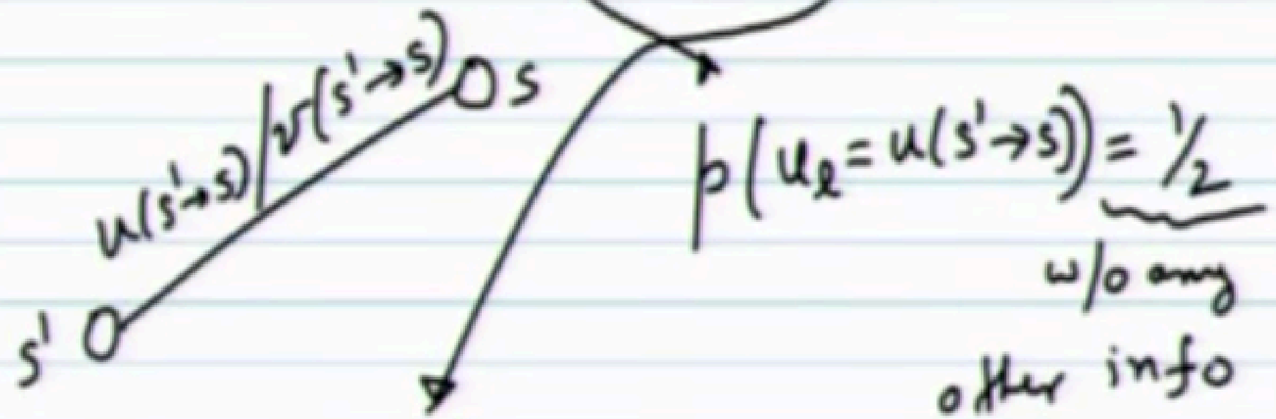
SEEING IN THE TRELLIS DIAGRAM



GAMMA CALCULATION

Branch metric: $\gamma_l(s', s) = p(S_{l+1} = s, r_l \mid S_l = s')$

$= p(S_{l+1} = s \mid S_l = s') \cdot p(r_l \mid S_l = s', S_{l+1} = s)$



$p(u_l = u(s' \to s)) = \frac{1}{2}$

w/o any other info

$p(r_l^{(1)}, r_l^{(2)} \mid v^{(1)}(s' \to s), v^{(2)}(s' \to s))$

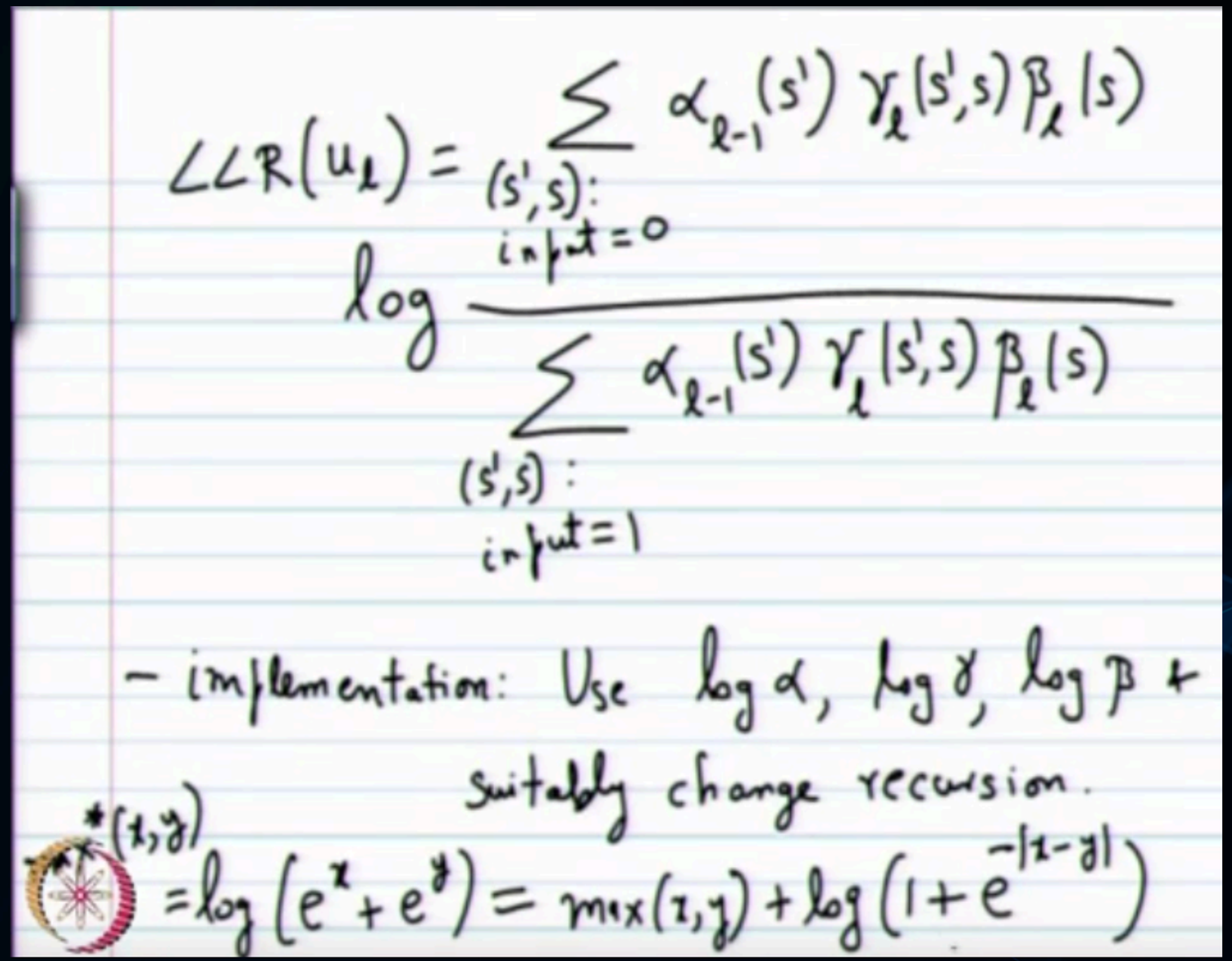
↓ BPSK

$y^{(1)}(s' \to s),\ y^{(2)}(s' \to s)$

IN IMPLEMENTATION, WE STORE THE LOG VALUES OF ALPHA, BETA, GAMMA:

ALPHA'S AND BETA'S GET CALCULATED BY RECURSION, THERE VALUES BECOME VERY SMALL, COMPUTERS CAN'T STORE THEM.

WE ALSO USE WINDOWING AS A LARGE NO OF VALUES HAVE TO BE STORED.



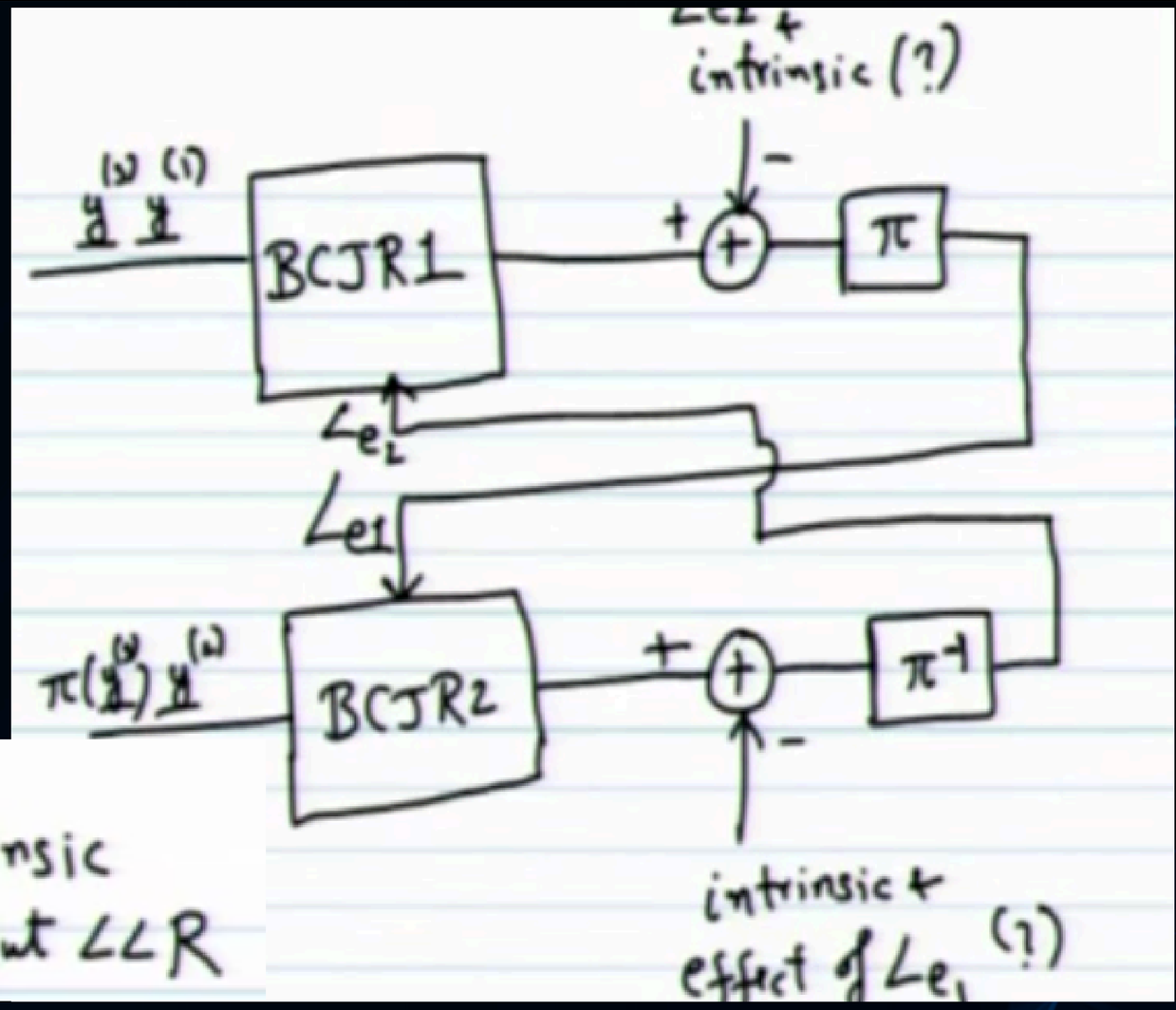
INTERLEAVER:
SHOULD BE ONE-ONE FOR INVERTIBLITY BUT BE CLOSE TO RANDOM.

IN THE INFO EXCHANGE, WHATEVER IT KNOWS ALREADY SHOULD NOT BE EXCAHNGED.
I.E, INTRINSIC LLR'S AND EFFECTS OF INFO COMING FROM THE PREVIOUS DECODER HAVE TO BE SUBTRACTED.,

Quadratic Permutation Polynomial (QPP) interleaver:

$$\pi(i) = f_1 i + f_2 i^2 \bmod K$$

$$i = 0, 1, \ldots, K-1$$



$$\text{Output LLR} = \delta_L + \frac{2\, y_l^{(s)}}{\sigma^2} + \sum_{\text{extrinsic output LLR}}$$

THANK YOU!

FOR YOUR ATTENTION/PATIENCE