IBM **Semiconductors**

Cross-Process Defect Attribution using Potential Loss Analysis

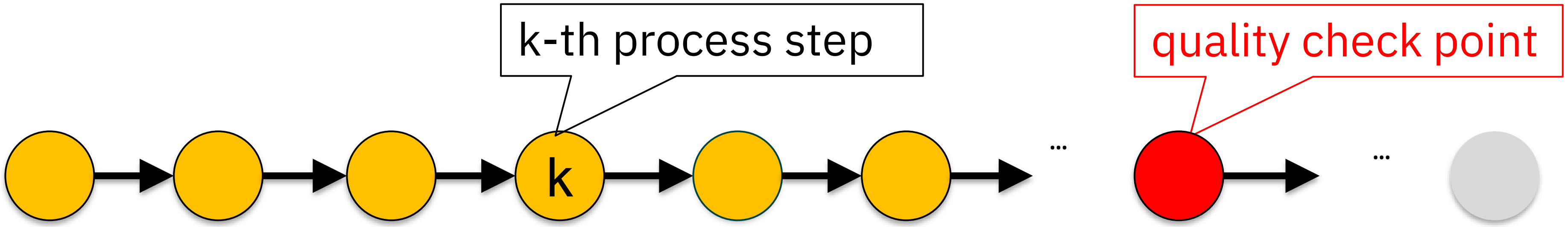
Tsuyoshi ("Ide-san") Ide and Kohei Miyaguchi*
IBM Semiconductors, T.J. Watson Research Center
*IBM Research – Tokyo (currently with LY Corporation)

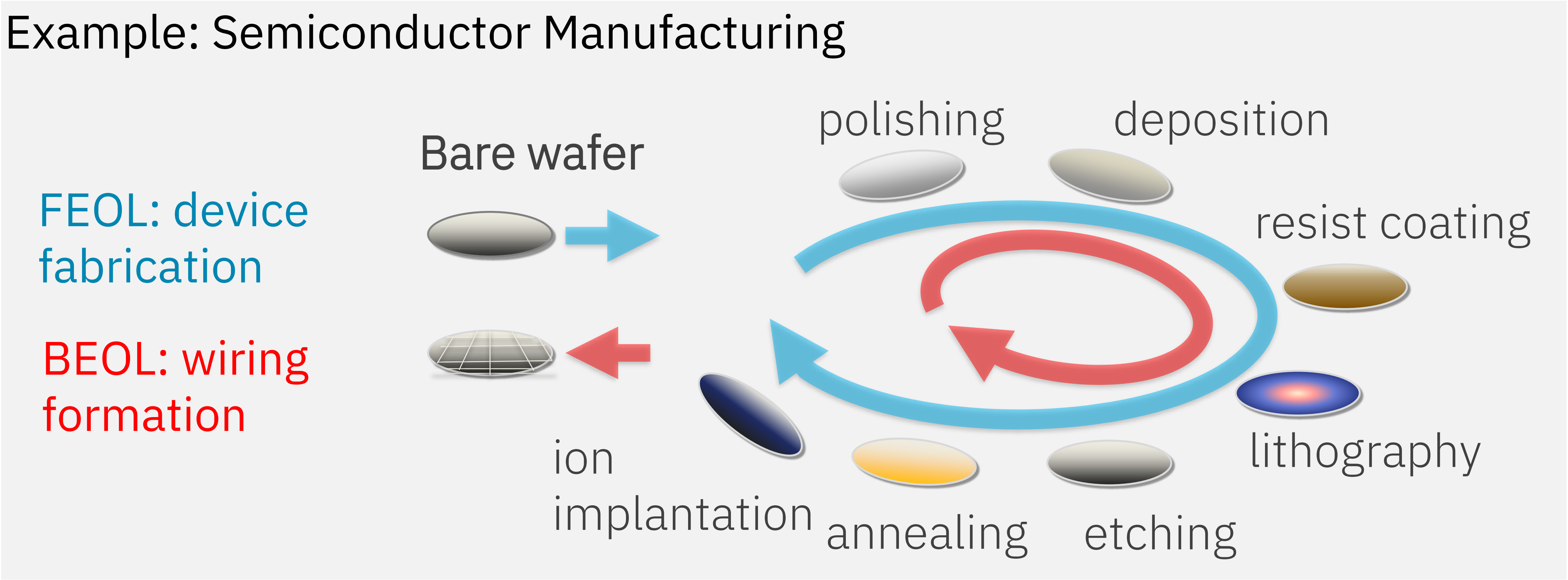
T. Ide and K. Miyaguchi, "Cross-process defect attribution suing potential loss analysis," Proceedings of the 2025 Winter Simulation Conference (WSC 25), to appear; https://arxiv.org/abs/2508.00895

Agenda

- Problem description and background
- Interventional causal attribution and its challenges
- Cross-process attribution with potential loss analysis (PLA)

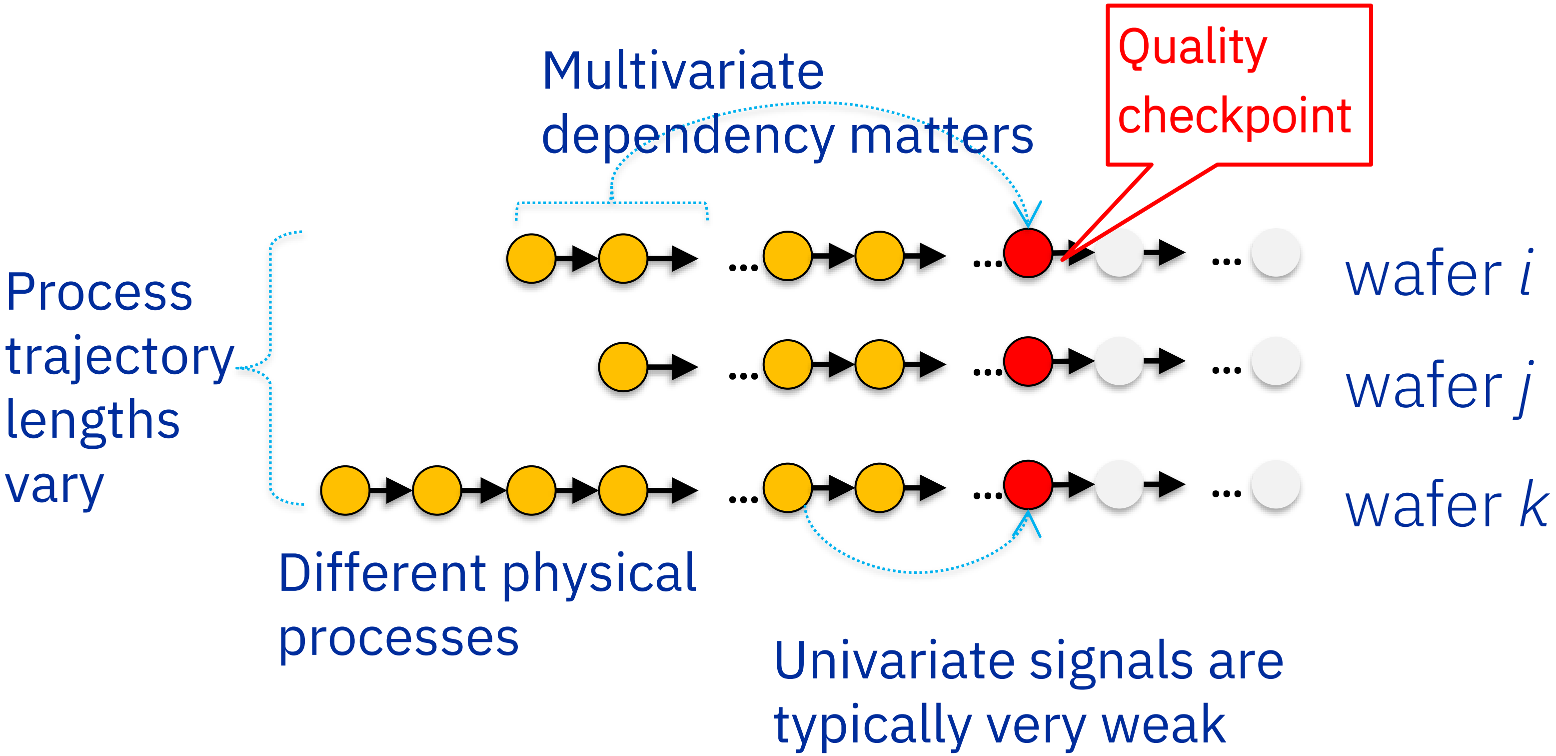
Target task: cross-process defect attribution





Target task: cross-process defect attribution

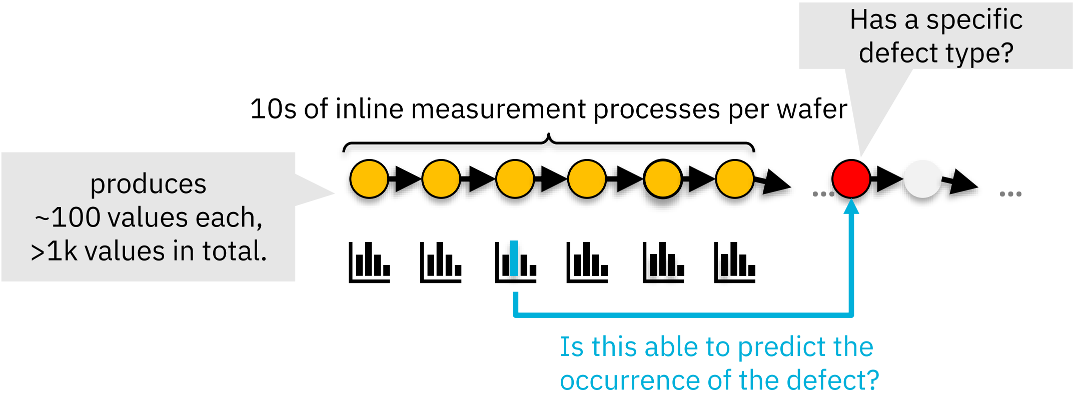
<u>Problem</u>: Given a wafer quality metric value, compute the **attribution score** for each of the upstream process steps.



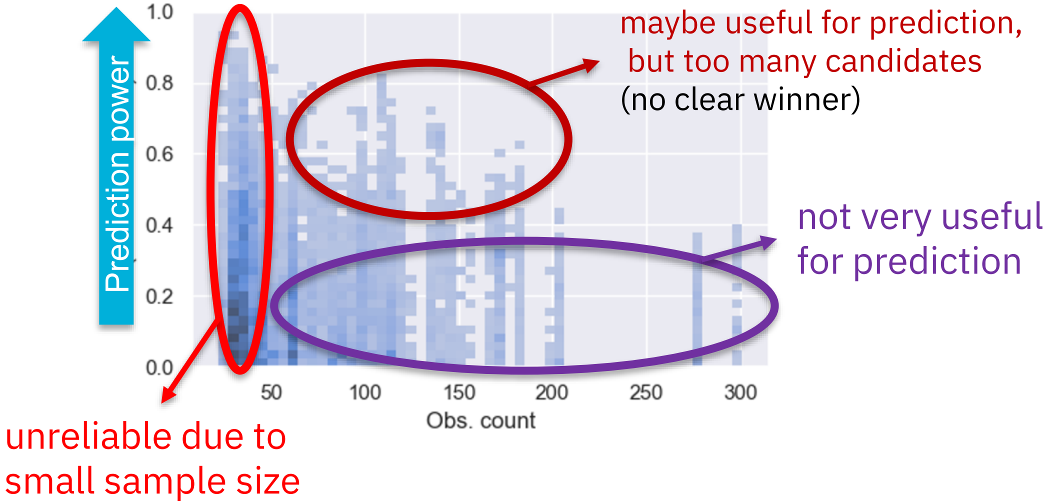
- In current practice…
  - The only viable approach is to run as many wafers as possible under varying conditions.
  - Then, relatively simple statistical analysis is applied.
  - This approach requires significant domain expertise to decide on parameter choices.
  - This semi-manual approach is reaching its limits as technology nodes advance.

Univeriate correlation analysis is often not informative

- Univariate analysis typically yields weak predictive signals.
  - Example: measurement-based univariate defect prediction
    - ✓ Trained a univariate binary classifier to distinguish between good and bad wafers.
    - ✓ Accuracy tends to decrease as the observation count increases.
- Traditional correlation metrics apply to populations, not to individual wafers.
  - We need a wafer-wise diagnostic algorithm.



Defect occurrence prediction with single measurement values



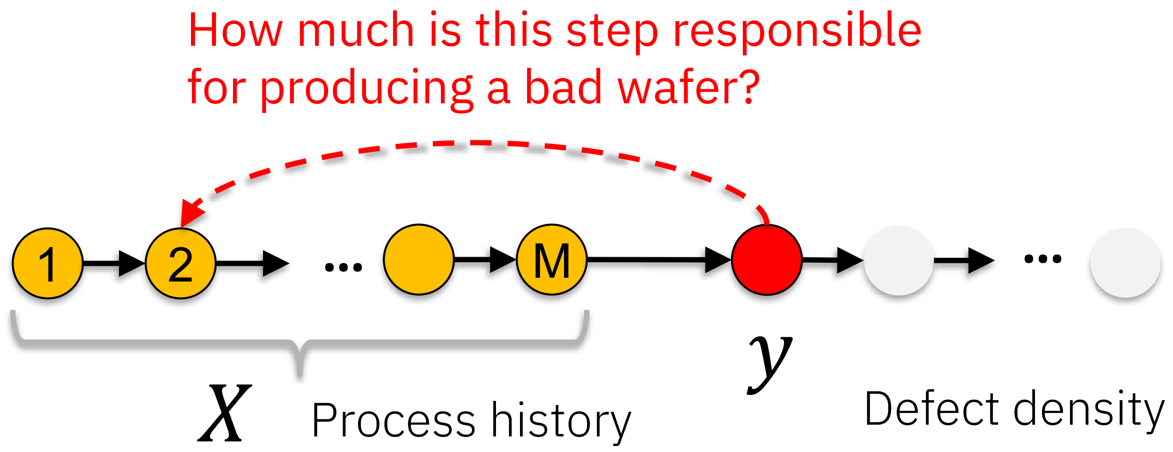
Agenda

- Problem description and background
- Interventional causal attribution and its challenges
- Cross-process attribution with potential loss analysis (PLA)

Defect attribution: Data assumptions and problem setting

- Data $D = \{(X^{(n)}, y^{(n)}) \mid n = 1, \dots, N\}$
  - $X^{(n)}$: process trajectory $(\boldsymbol{x}_1^{(n)}, \dots, \boldsymbol{x}_{L^{(n)}}^{(n)})$. Each process is assumed to have a vector representation (→ discussed later).
  - $y^{(n)}$: Product badness such as defect density (a real number).

- Task: wafer-wise defect attribution
  - Given a wafer quality metric value, compute the <u>attribution score</u> for each of the process steps.

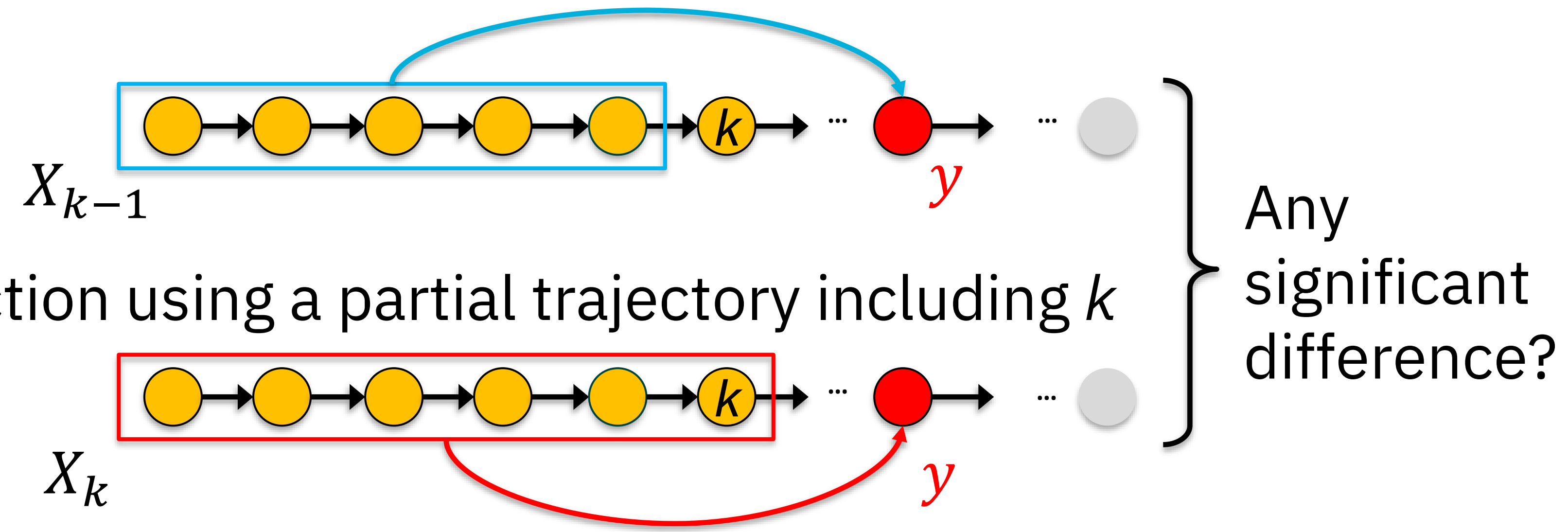


Interventional Causal Attribution with Partial trajectory regression (PTR)

- PTR computes the attribution score of the *k*-th process by evaluating the impact of *k*'s "participation" in the process trajectory.

A: Prediction using a partial trajectory **not** including *k*

B: Prediction using a partial trajectory including *k*



Miyaguchi et al., "Wafer Defect Root Cause Analysis with Partial Trajectory Regression," ASMC 2025

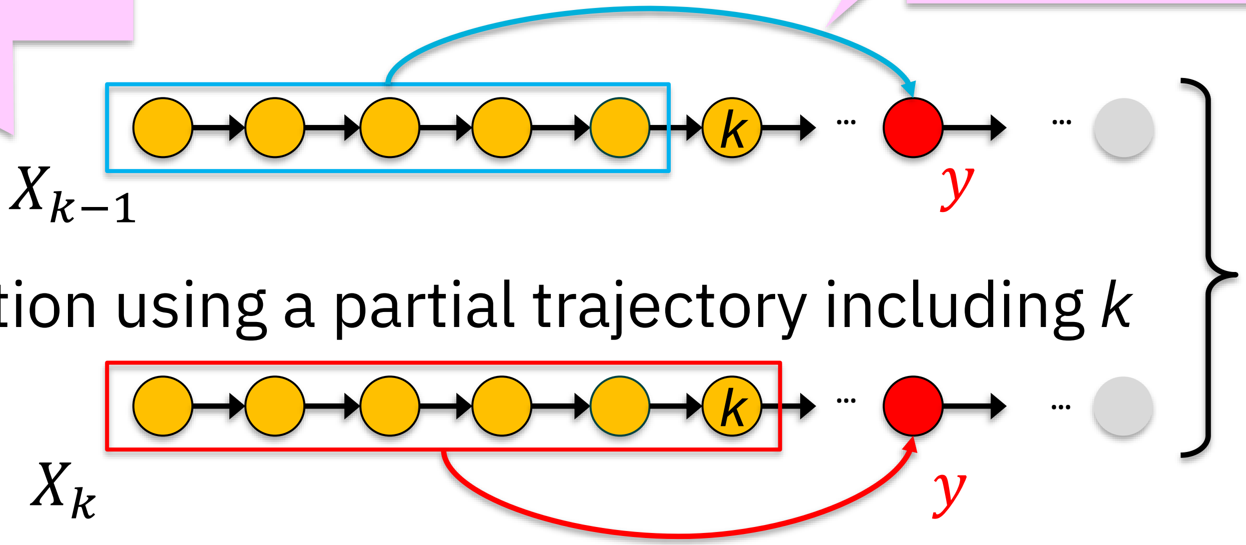
Interventional Causal Attribution with Partial trajectory regression (PTR)

- PTR computes the attribution score of the *k*-th process by evaluating the impact of *k*'s “participation” in the proce

How do we represent the process trajectories mathematically?

using a partial trajectory **n**

How do we predict *y* from a **partial** process trajectory ?

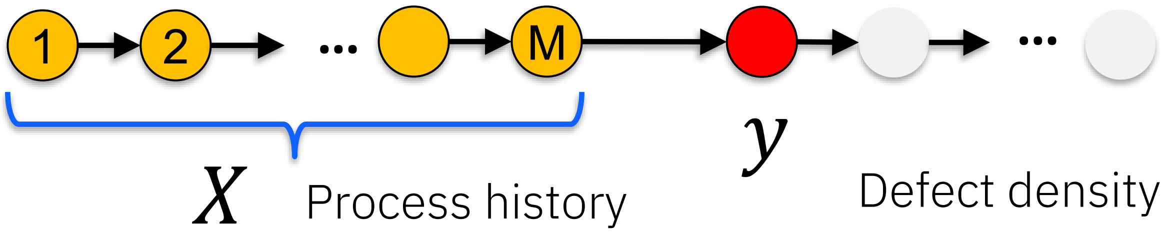


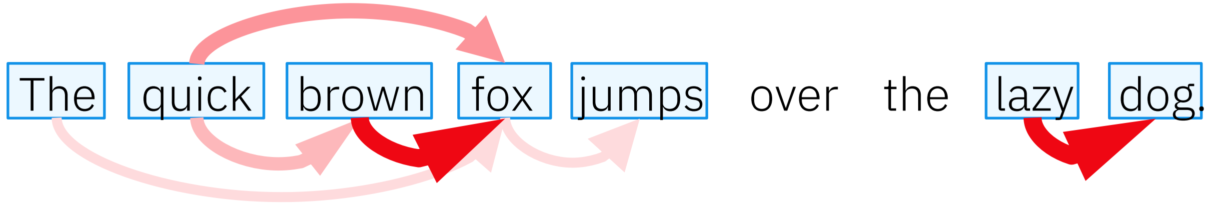
Any significant difference?

Miyaguchi et al., “Wafer Defect Root Cause Analysis with Partial Trajectory Regression,” ASMC 2025

Representing process trajectories as vector sequences: Process embedding ("proc2vec")

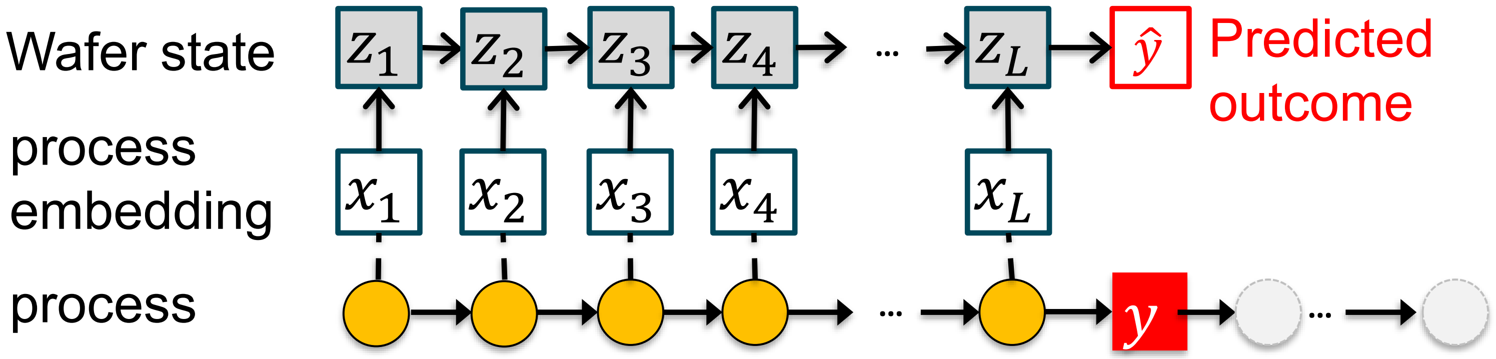
- Approach 1: Use measurement data (e.g., CDs) as a surrogate for process steps.
  - Straightforward, but data may have many missing entries due to wafer sampling.
- Approach 2: Use process data and apply (deep) embedding.
  - $\boldsymbol{x} = \mathrm{ReLU}\,(\mathrm{W}\boldsymbol{u} + \boldsymbol{b})$,
    - ✓ $\boldsymbol{u}$: some process data; W, $\boldsymbol{b}$: trainable parameters
  - Data hungry, prone to overfit.
- Approach 3: Symbolic embedding
  - Create a synthetic process token
    - ✓ "Process token"=eqp_id$\oplus$ recipe_id$\oplus \cdots \oplus$tool_trace
  - Employ multidimensional scaling based on a similarity matrix among the tokens





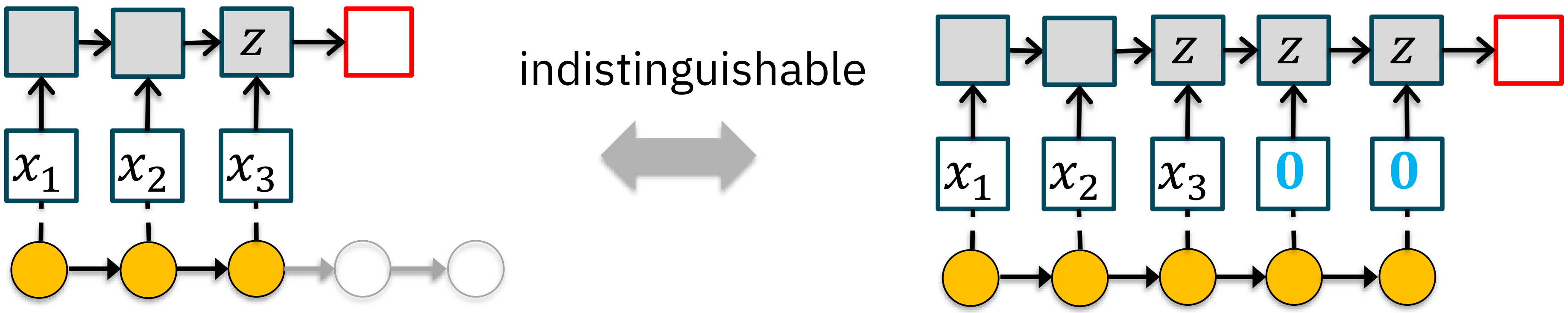
Learning partial trajectory regression model

- Typically, a prediction function has a fixed number of input slots.
  - For 3 processes, it would be like $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$, a 3-slot function.
  - Hence, cannot handle process trajectories with different lengths.
- State-space model (or RNN) eliminates this limitation
  - Partial prediction by a length-k trajectory: $\hat{y} = f(\boldsymbol{z}_k = \mathrm{RNN}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_k))$
    - ✓ f: A parametric function with trainable parameters
    - ✓ $\boldsymbol{z}_k$: Latent state vector after observing $\boldsymbol{x}_k$
- Off-the-shelf RNN (e.g., LSTM, GRU) models need adjustments.
  - → next page



Catch: You can't simply zero off downstream processes

- RNNs are generally data hungry. Careful model customization is needed.
  - Process timestamp needs special treatment (→ paper)
- RNNs may introduces a significant bias.
  - Example: k=3 partial prediction is indistinguishable with k=5 partial prediction with zeroed-off input.
    - ✓ i.e., RNN's partial trajectory prediction = full trajectory prediction with a zeroed-off process sequence



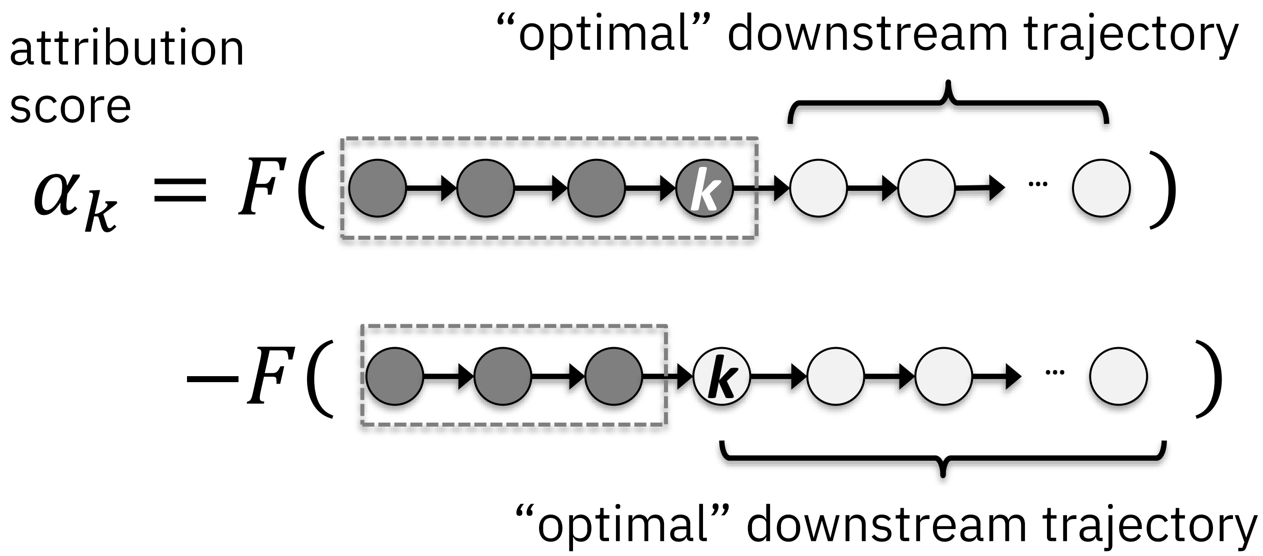
Agenda

- Problem description and background
- Interventional causal attribution and its challenges
- Cross-process attribution with potential loss analysis (PLA)

Eliminating the bias of partial trajectory analysis: Potential Loss Analysis (PLA)

- PLA: The attribution score for k is evaluated, given an optimal downstream trajectory:
  - $\min_{\boldsymbol{x}_{k+1},\ldots,\boldsymbol{x}_L} F(\boldsymbol{z}_k, \boldsymbol{x}_{k+1}, \ldots, \boldsymbol{x}_L)$
    - ✓ $\boldsymbol{z}_k$: latent wafer state at process k
    - ✓ We don't use $F(\boldsymbol{z}_k, \boldsymbol{0}, \ldots, \boldsymbol{0})$.

- This trajectory optimization problem can be solved as a Bellman equation.
  - → Next page



Formalizing PLA as a reinforcement learning problem (1/2)

- $\min_{\boldsymbol{x}_{k+1},\ldots.} F(\boldsymbol{z}_k, \boldsymbol{x}_{k+1}, \ldots.) = \min_{\boldsymbol{x}_{k+1},\ldots.} \mathbb{E}[\sum_{t=1}^{\infty} C(\boldsymbol{z}_{k+t}) \mid \boldsymbol{z}_k]$
  - terminal reward model: $C(\boldsymbol{z}) = \begin{cases} y(\boldsymbol{z}), & z \in (\text{terminal state}) \\ 0, & \text{otherwise} \end{cases}$
- Bellman equation
  - $F^*(\boldsymbol{z}) \equiv \min_{\boldsymbol{x}_1,\ldots.} F(\boldsymbol{z}, \boldsymbol{x}_1, \ldots) = \min_{\boldsymbol{x}_1} \{C(\boldsymbol{z}) + \sum_{\boldsymbol{z}_2} \underbrace{p(\boldsymbol{z}_2 \mid \boldsymbol{x}_1, \boldsymbol{z}_1)}_{\text{transition model (assumed deterministic)}} F^*(\boldsymbol{z}_2)\}$
- The optimization problem we solve (with $F^\theta(\boldsymbol{z})$ approximating $F^*(\boldsymbol{z})$):
  - $\max_\theta \sum_{\boldsymbol{z}} \underbrace{\rho(\boldsymbol{z})}_{\text{empirical density of } \mathbf{z} \text{ (under deterministic assumption)}} F^\theta(\boldsymbol{z}) \quad \text{s.t.} \quad F^\theta(\boldsymbol{z}) \le C(\boldsymbol{z}) + F^*(\boldsymbol{z}'),\ \forall(\boldsymbol{z} \to \boldsymbol{z}'),$

K. Miyaghichi, "Path Learning with Trajectory Advantage Regression," 2025, https://arxiv.org/abs/2506.19375

Formalizing PLA as a reinforcement learning problem (2/2)

- Final objective function to be maximized
  - $$R(\theta \mid \mu) = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{\mu}{L^{(n)}}\sum_{t=1}^{L^{(n)}} F^{\theta}\left(z_t^{(n)}\right) - \underbrace{\frac{1}{2}\left\{y^{(n)} - F^{\theta}\left(z_{L^{(n)}}^{(n)}\right)\right\}^2}_{\text{squared loss}} - \frac{1}{2}\sum_{t=1}^{L^{(n)}-1}\left\{F^{\theta}\left(z_{t+1}^{(n)}\right) - F^{\theta}\left(z_t^{(n)}\right)\right\}^2\right]$$
  - This provides the partial prediction function and attribution model simultaneously.

- The time difference of the F function can be directly parameterized:
  - $G^{\theta}(\boldsymbol{z}_t, \boldsymbol{z}_{t-1}) \equiv F^{\theta}(\boldsymbol{z}_t) - F^{\theta}(\boldsymbol{z}_{t-1}) = \mathrm{ReLU}_{\theta}(\boldsymbol{z}_t \oplus \boldsymbol{z}_{t-1})$
  - This function provides the attribution score for the k-th process.
    - ✓ Yields a positive number

K. Miyaghichi, "Path Learning with Trajectory Advantage Regression," 2025, https://arxiv.org/abs/2506.19375

2nm process defect diagnosis example

- The graphs plot the cumulative attribution score.
  - Big jump = likely root cause
  - The model was trained with 727 wafers.
- PTR approach does not provide meaningful signals.
- PLA successfully detects likely root cause
  - In this case, they correspond to too long waiting hours at a certain piece of equipment.