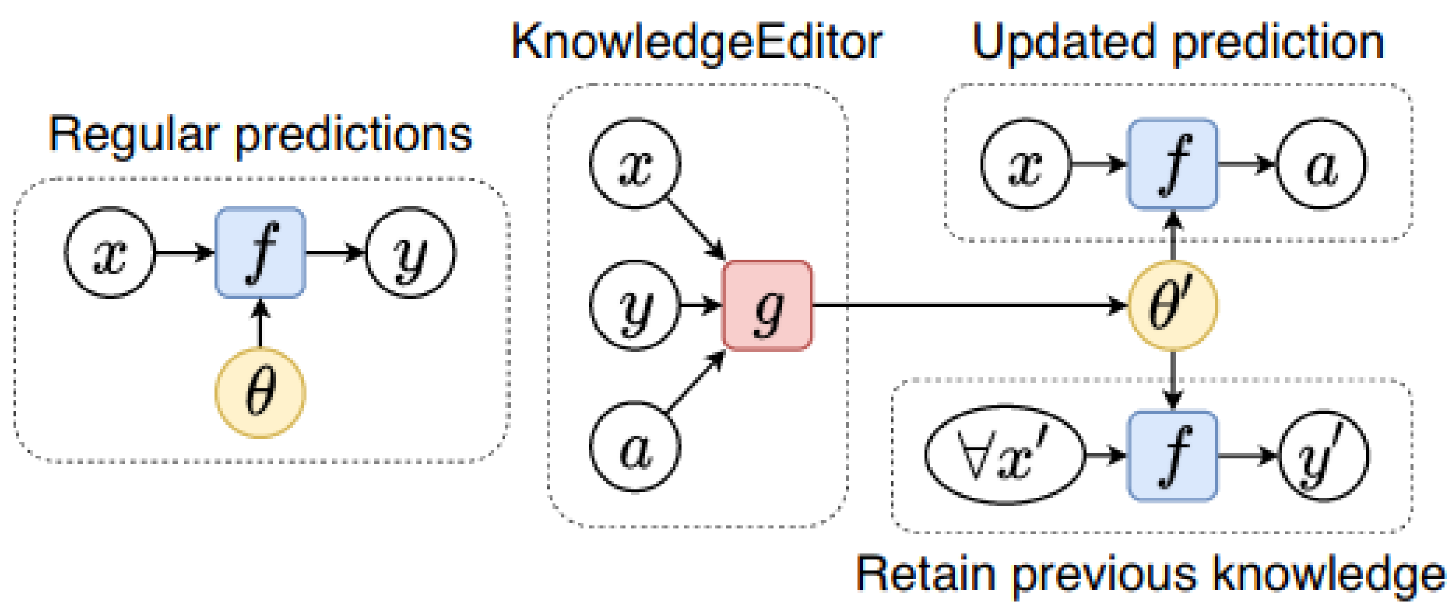
Editing Factual Knowledge in Language Models

Nicola De Cao, Wilker Aziz, Ivan Titov

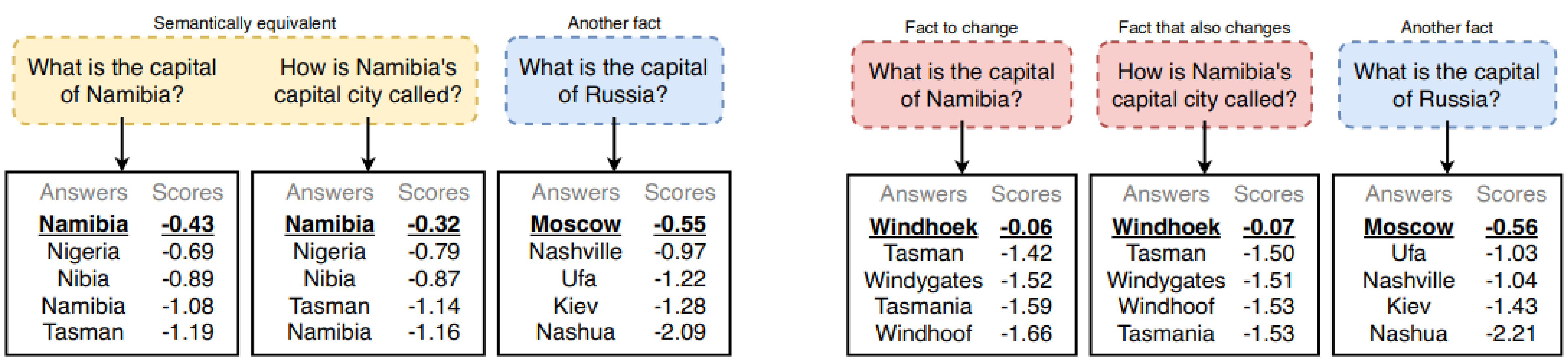
https://aclanthology.org/2021.emnlp-main.522/

Goal: Edit factual knowledge in a language model

In practice: try to change the output y corresponding to only one input x to a, by changing the parameter of the model



Evaluation: based on groups of semantically equivalent inputs P^x that should change too, and other inputs O^x that should remain unchanged



(a) Model predictions before the update.

(b) Model predictions with edited parameters.

Method: Use an *hyper-network* g to generate parameters θ'

Objective: finding the parameters minimizing $\mathcal{L}(\theta; x, a)$!

Optimization:

$$\begin{aligned} \min_{\phi} \quad & \sum_{\hat{x} \in \mathcal{P}^x} \mathcal{L}(\theta'; \hat{x}, a) \\ \text{s.t.} \quad & \mathcal{C}(\theta, \theta', f; \mathcal{O}^x) \leq m \end{aligned}$$

The prediction is of the form: $y = argmax_{c \in \mathcal{Y}} p_{Y|X}(c|x, \theta)$

Hence, the constraint is written: $C_{KL}(\theta, \theta', f; O^x) =$

$$\sum_{x' \in \mathcal{O}^x} \sum_{c \in \mathcal{Y}} p_{Y|X}(c|x', \theta) \log \frac{p_{Y|X}(c|x', \theta)}{p_{Y|X}(c|x', \theta')}$$

Re-written by using *Lagrangian relaxation*; then approximately evaluate the constraint via *Monte Carlo sampling* (and *beam search* when a sequence is to be generated)

Now, in practice: Try to generate the shift from θ, $\Delta\theta$!

- $< x, y, a >$ (as text with separators) is fed to a Bi-LSTM, which outputs h
- h is an input to 5 FFNNs by weight matrix $W^{n \times m} \in \theta$ of the original model, which each produce $\alpha, \beta \in \mathbb{R}^m, \gamma, \delta, \mathbb{R}^n$ and a scalar $\eta \in \mathbb{R}$. Then, the shift ΔW is seen a gated sum of **a scaled gradient of the objective** and a bias term.

  $$\Delta W = \sigma(\eta) \cdot \left(\hat{\alpha} \odot \nabla_W \mathcal{L}(W; x, a) + \hat{\beta}\right)$$
  $$\text{with} \quad \hat{\alpha} = \hat{\sigma}(\alpha)\gamma^\top \quad \text{and} \quad \hat{\beta} = \hat{\sigma}(\beta)\delta^\top$$

  This allows to efficiently parametrise a matrix with a reduced number of vectors.
- Annealing is used to find the hyperpameter m.

Evaluation: 4 measures -

- *Sucess rate*: Accuracy of revised predictions - shows how well g changes the parameters to the right θ'
- *Retain accuracy*: How well original predictions are retained, measured as accuracy on O^x
- *Equivalence accuracy*: Consistency of revised model, measured as accuracy on P^x
- *Performance deterioration*: How much test performance of the revised model deteriorates

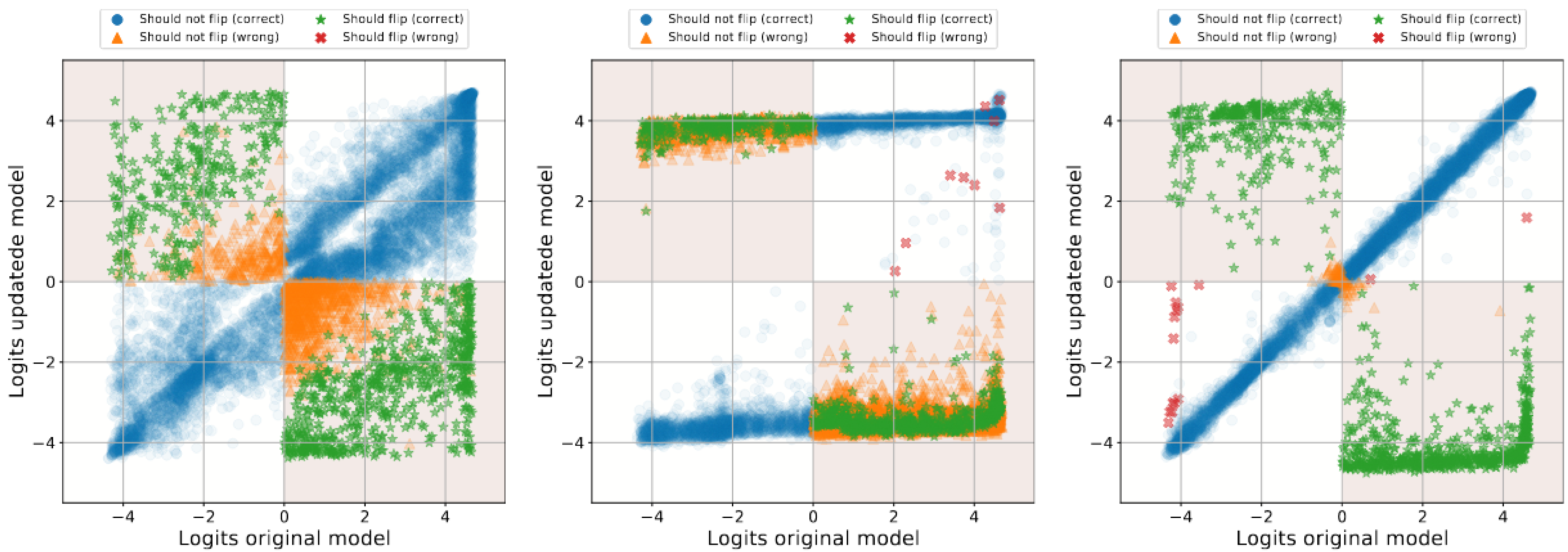
Retain and *equivalence* accuracy are the main innovations in evaluating compared to the related works:

- **Modifying Memories in Transformer Models** (Zhu et al, 2020): based on meta-learning, but costly (regularized updates on the full network)
- **Editable Neural Networks** (Sinitsin et al, 2020): fine-tuning with a norm-based contraint on parameters

Tasks:

- *Fact-checking*: Binary prediction from text - using a BERT model on FEVER dataset.
- *Closed-book Question answering*: Generating a sequence of text (response) to a question, using a fine-tuned BART model on the Zero-Shot Relation Extraction dataset.
- Alternative predictions generated by changing labels/non-best beam search results; semantically equivalent intputs generated via back-translations.

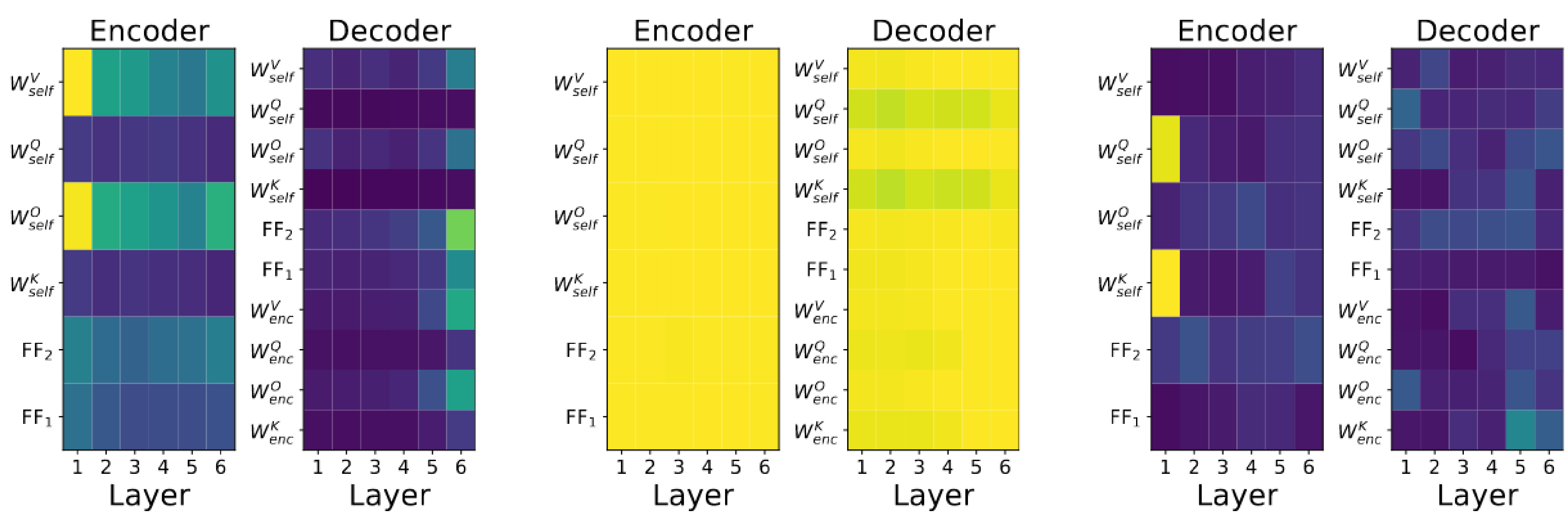
	Fact-Checking				Question Answering			
Method	Success rate ↑	Retain acc ↑	Equiv. acc ↑	Perform. det ↓	Success rate ↑	Retain acc ↑	Equiv. acc ↑*	Perform. det ↓
Fine-tune (1st layer)	100.0	99.44	42.24	0.00	98.68	91.43	89.86 / 93.59	0.41
Fine-tune (all layers)	100.0	86.95	95.58	2.25	100.0	67.55	97.77 / 98.84	4.50
Zhu et al. (1st layer)	100.0	99.44	40.30	0.00	81.44	92.86	72.63 / 78.21	0.32
Zhu et al. (all layers)	100.0	94.07	83.30	0.10	80.65	95.56	76.41 / 79.38	0.35
Ours $\mathcal{C}_{L_2}$	99.10	45.10	99.01	35.29	99.10	46.66	97.16 / 99.24	9.22
KnowledgeEditor	98.80	98.14	82.69	0.10	94.65	98.73	86.50 / 92.06	0.11
+ loop†	100.0	97.78	81.57	0.59	99.23	97.79	89.51 / 96.81	0.50
+ $\mathcal{P}^{x\ \ddagger}$	98.50	98.55	95.25	0.24	94.12	98.56	91.20 / 94.53	0.17
+ $\mathcal{P}^{x}$ + loop‡	100.0	98.46	94.65	0.47	99.55	97.68	93.46 / 97.10	0.95



(a) Fine-tune (all layers).

(b) $\mathcal{C}_{L_2}$.

(c) Ours $\mathcal{C}_{KL}$ with $\mathcal{P}^x$.



(a) Gradients.

(b) Fine-tune (all layers).

(c) KNOWLEDGEEDITOR + $\mathcal{P}^x$.