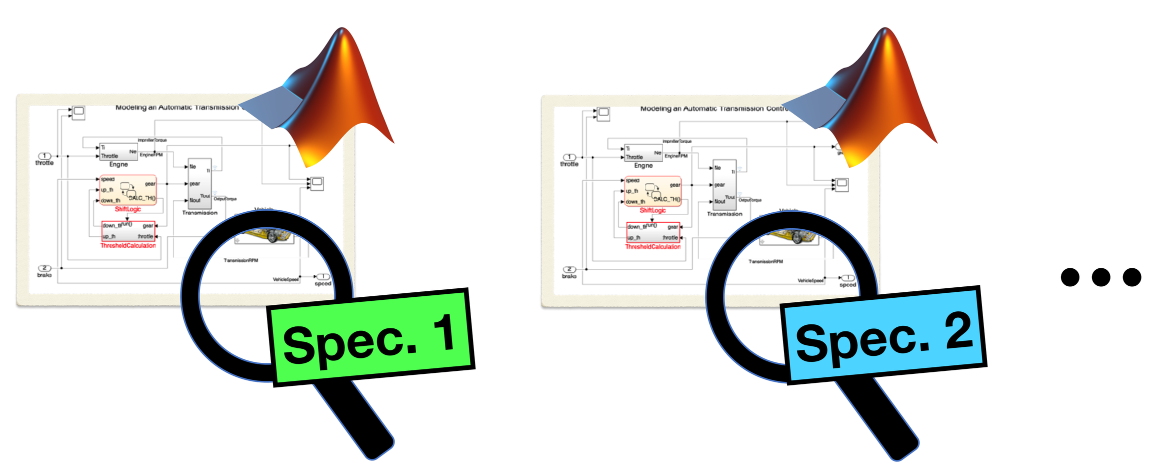
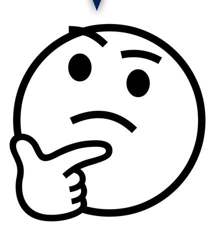
FalCAuN – CPS Testing with Automata Learning

Masaki Waga ・ Kyoto University

Q. How to enhance system testing?
e.g. Reusability, Explanation, Theoretical gurantee, …

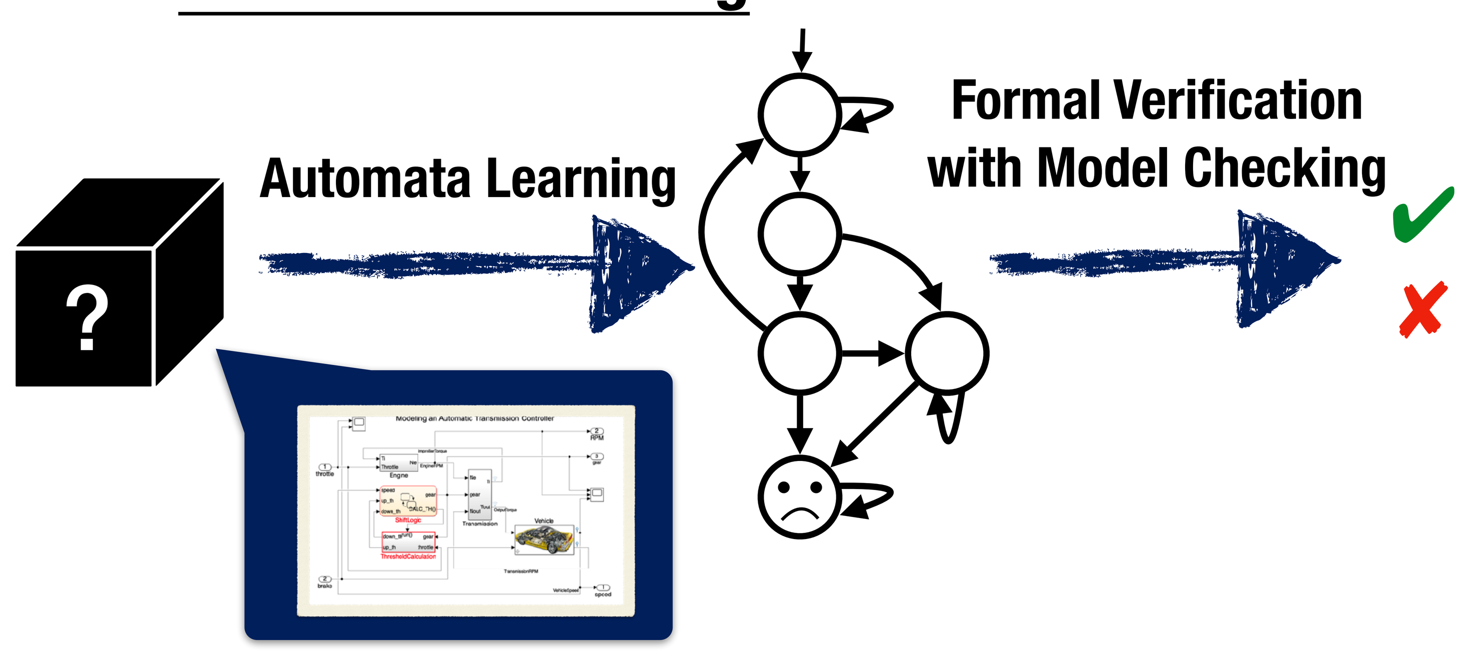


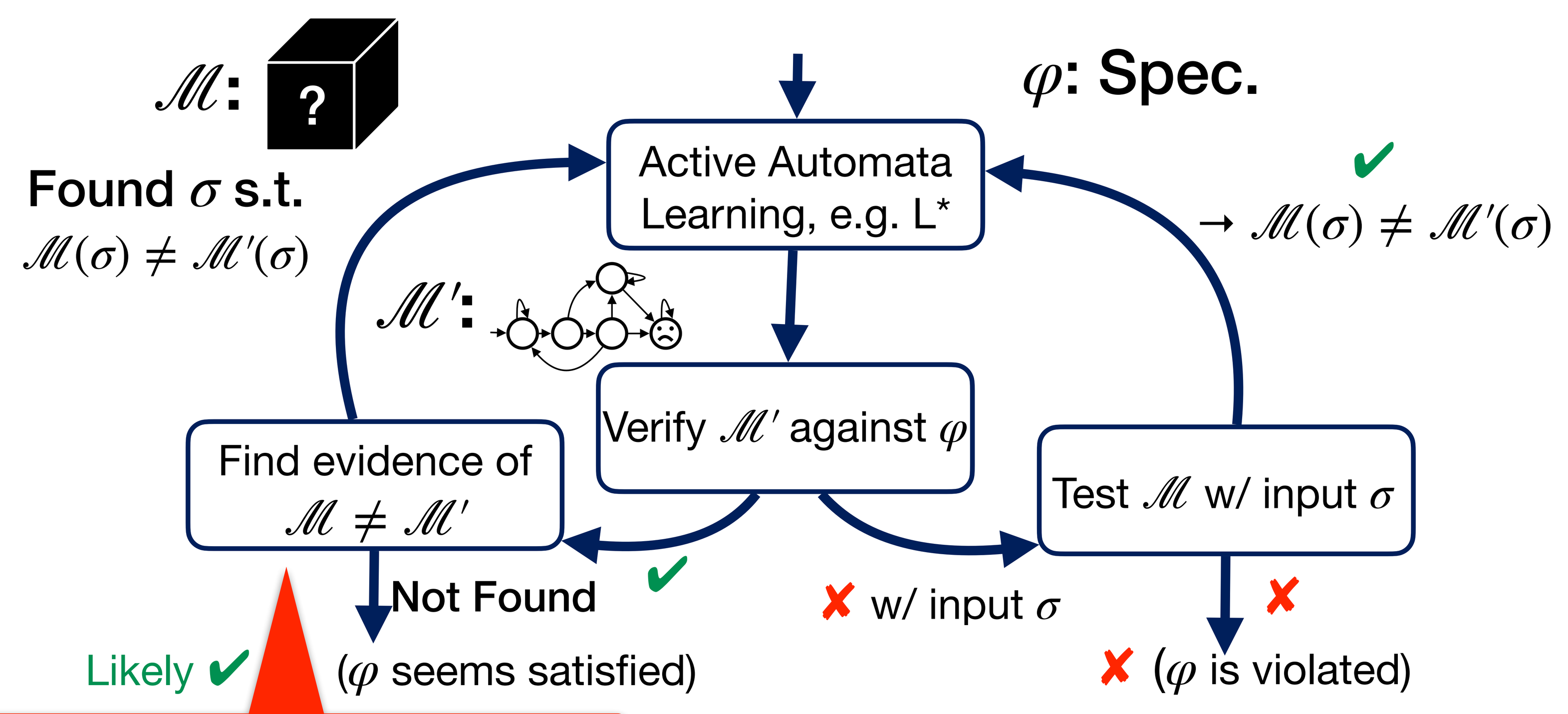
Can testing be faster?
Why tests passed?
Is it reliable?



Black-Box Checking for CPS

Black-Box Checking [Peled et al., PSTV & FORTE'99]

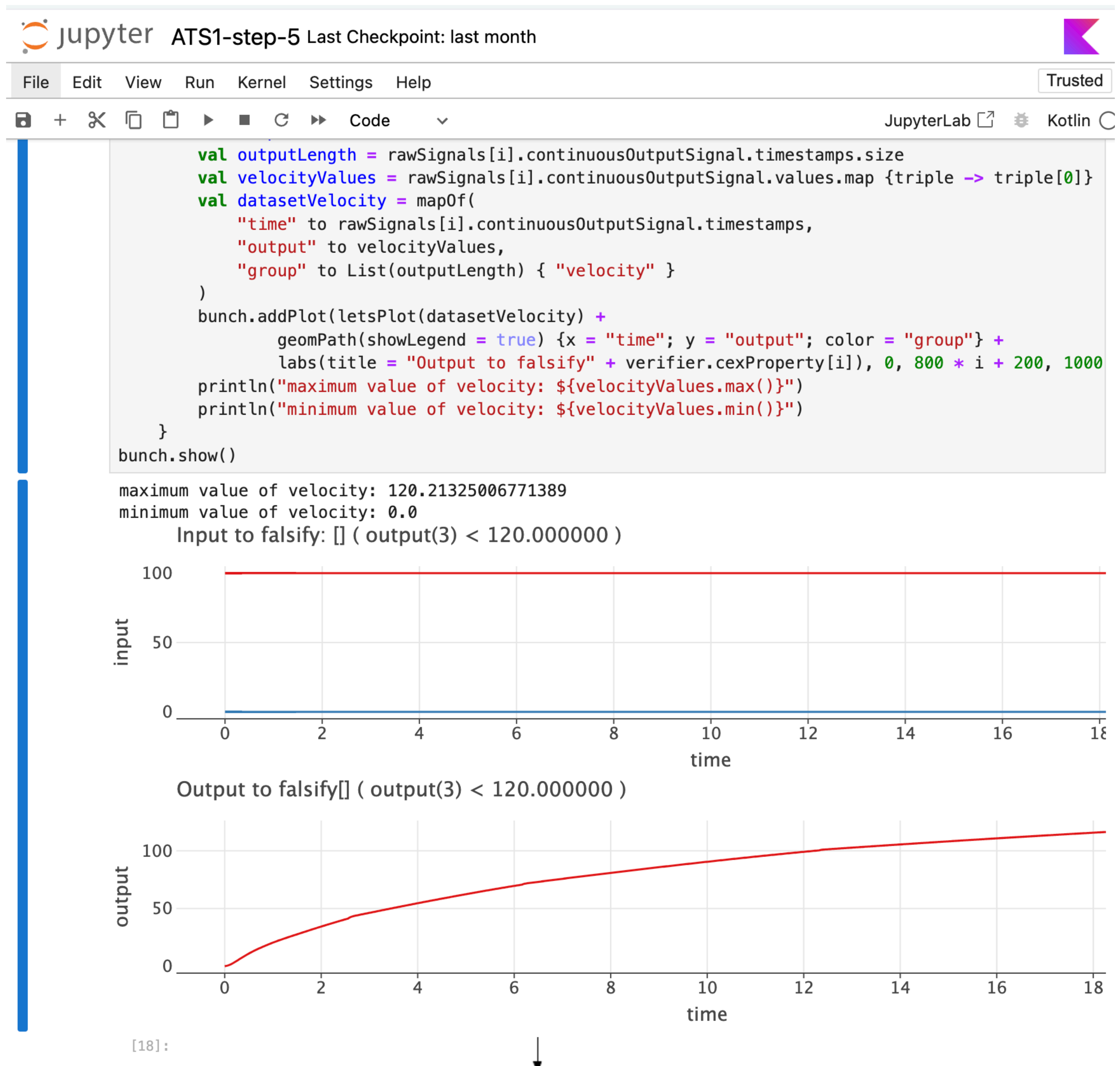


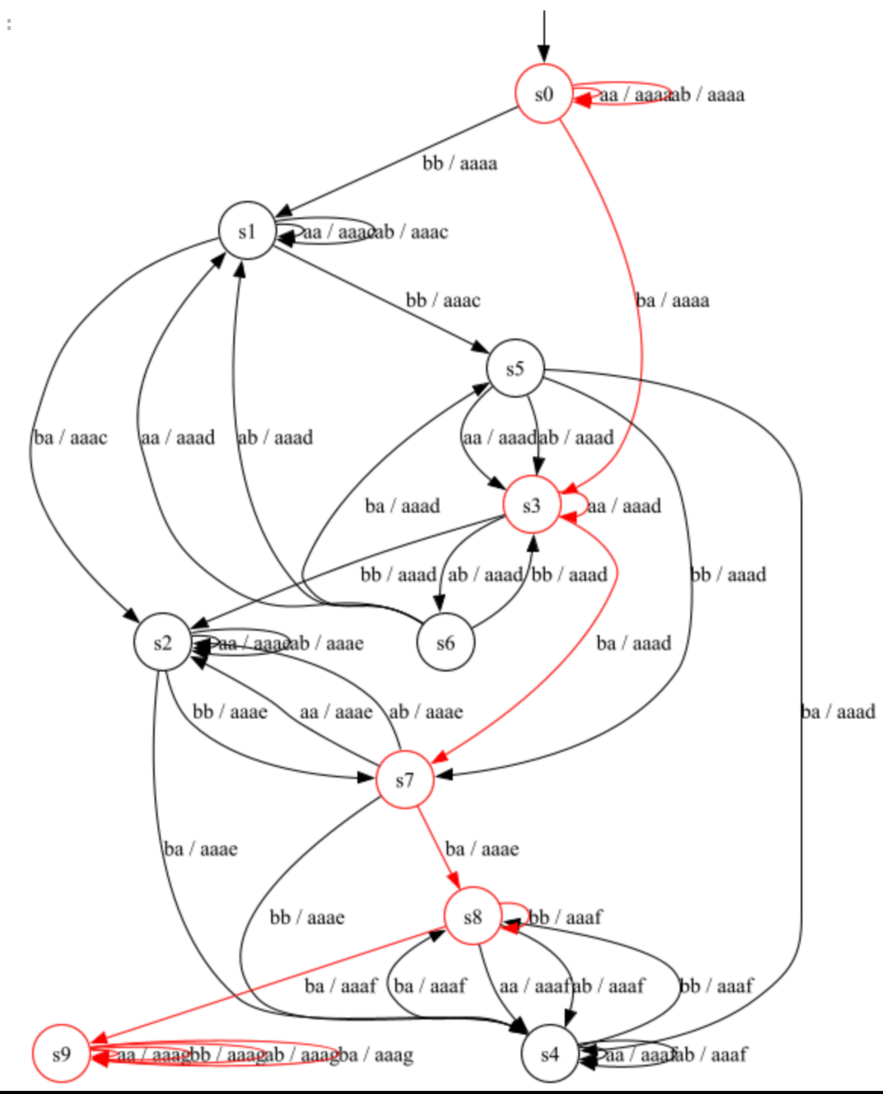


Difficult Part!!
- Typically by random test
- Hard to find corner cases or something useful to falsify φ

Our Approach Testing black-box CPS with Learning of formal model + Verification

Our Toolkit: FalCAuN (on Jupyter with Kotlin Kernel)





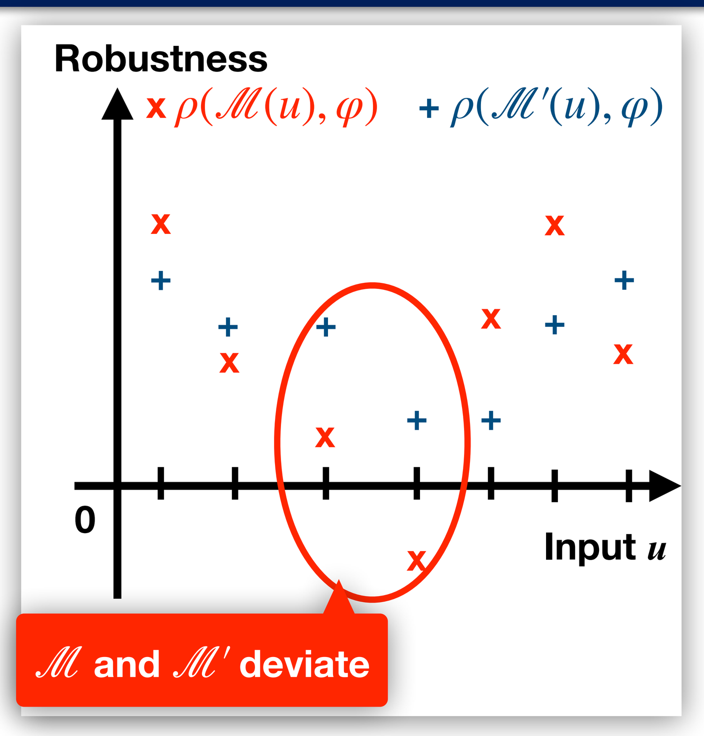
Robustness-guided equivalence test [Waga, HSCC'20]

Idea: Find evidence of $\mathcal{M} \neq \mathcal{M}'$ using inputs w/ low robustness i.e. use inputs leading "near dangerous" status

Assumption: $\mathcal{M} \nvDash \varphi$
Robustness of $\mathcal{M}$ can get negative for some inputs

Fact: $\mathcal{M}' \vDash \varphi$
Robustness of $\mathcal{M}'$ is always positive (Guaranteed by model checking)

Heuristic: Find u s.t. $\mathcal{M}(u) \neq \mathcal{M}(u)$ focusing on u making $\mathcal{M}$ less robust



Counterexample synthesis via Model Checking of strengthened formulas [Shijubo, Waga, Suenaga, RV'21]

Idea: Model checking of "related" specification can find useful evidence of $\mathcal{M} \neq \mathcal{M}'$

Fact: Counterexample σ of model checking progresses learning if $\mathcal{M}$ does not violate φ with σ

Observations:
- Model checking is typically faster than equivalence testing
- σ obtained by model checking is often useful for learning since it is related to φ
- Also the case for the formulas "related" to φ

Approach: Syntactically strengthen LTL formulas and conduct model checking with it

Eventually μ → … → Always μ

1. *For any $\mu, \nu \in$ **LTL**, we have $(\mu \vee \nu) \rightarrowtail (\mu \wedge \nu)$.*
2. *For any $\mu \in$ **LTL**, we have $\Diamond\mu \rightarrowtail \Box\Diamond\mu$.*
3. *For any $\mu \in$ **LTL**, we have $\Box\Diamond\mu \rightarrowtail \Diamond\Box\mu$.*
4. *For any $\mu \in$ **LTL**, we have $\Diamond\Box\mu \rightarrowtail \Box\mu$.*
5. *For any $\mu \in$ **LTL** and for any indices $i, j \in \mathbb{N} \cup$ have $\Diamond_{[i,j)}\mu \rightarrowtail \Box_{[i,j)}\mu$.*
6. *For any $\mu, \nu \in$ **LTL**, we have $(\mu\, \mathcal{U}\, \nu) \rightarrowtail (\Box\mu \wedge \Box$*

⋮

Notes on formal guarantee

Assumption: Target system can be modeled with a Mealy machine $\mathcal{M}$

- If ∀ input, eq. test eventually try it, then we can find any counterexample in the limit
- If we know the number of states of $\mathcal{M}$, we can stop eq. test with correctness guarantee (based on conformance testing, such as W-method [Chow, TSE' 78])
  - Alternatively, we can stop with probably approximately correct (PAC) guarantee [Angluin, '87]

Future directions

- Testing of Python classes, particularly (R)NNs
- Better illustration of falsifying executions
- Support of hyperproperties, e.g., to test robustness and fairness
- Support numeric inputs (w/ symbolic automata)
- Extension for stochastic systems

This work is partially supported by JST ACT-X Grant No. JPMJAX200U, JST PRESTO Grant No. JPMJPR22CA, JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST CREST Grant Number JPMJCR2012, and JSPS KAKENHI Grant Number 15KT0012, 18J22498, 19H04084, and 22K17873.