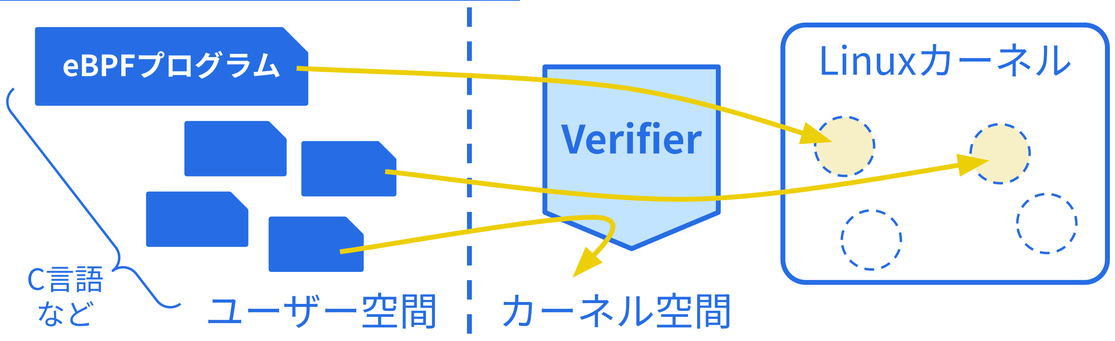
eBPFプログラムの機能正当性検証のための モデル検査と実行時検証の統合

神拓己　和賀正樹　五十嵐淳　末永幸平　　京都大学情報学研究科

①背景：eBPF

Linuxカーネルの機能を**安全に**拡張するための技術

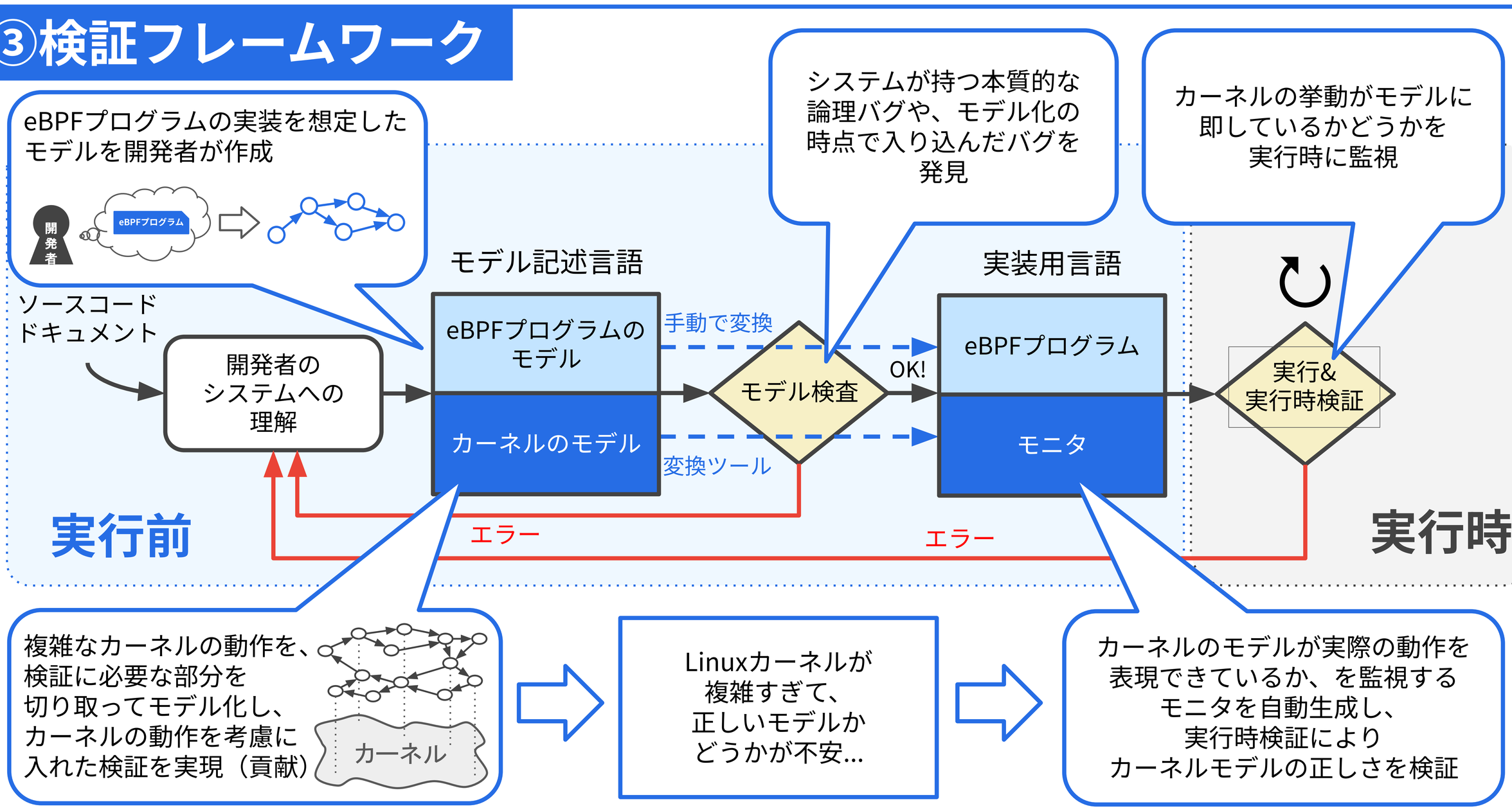


- 複数の小さなeBPFプログラムとカーネルが**協調して動作**
- メモリ安全性や停止性などの**”安全性”**をVerifierが検証
  → **意図した機能が実現されているか、といった機能正当性は検証しない**

②目標

Linuxカーネルと密に連携して動作するeBPFプログラムの機能正当性検証

③検証フレームワーク



④pml2bpf

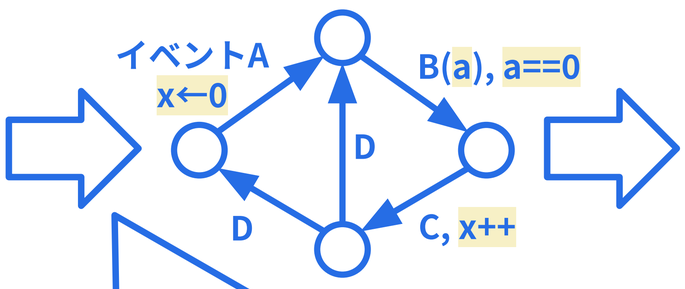
モデル記述言語Promelaで書かれたカーネルモデルからモニタ実装への変換ツール

- カーネルイベントの発生順序を手続き型言語のように記述
- 実行時検証用の**変数**や**遷移条件**を記述可能（貢献）
- カーネルの**非決定的な制御フロー**を表現可能（貢献）

Promela

```
inline __schedule() {
  if
  :: INVOKE_EBPF(ops_dispatch);
  :: skip
  fi
  pick_next_task(next_pid);
  if
  :: RV_CTX(curr_pid) != next_pid
     -> rv_sched_switch(next_pid);
  :: else
  fi
}
```

実行時検証用のアノテーションをつける



1つのカーネルイベントの発生と1つの状態遷移が対応するような拡張有限状態機械に変形

eBPF

```
void transition(...){...}
void probe_A(...){
  transition(EVENT_A);
  ctx->x = 0;
}
void probe_B(...){...}
void probe_C(...){...}
```

カーネルイベントのプローブ関数で遷移が妥当かを検証

⑤ケーススタディ

- モデル検査に**SPIN**を採用
- eBPFで実装されたタスクスケジューラに対して本手法を適用
  → ワークコンサービング性に違反する
  既知のバグを発見することに成功
- カーネルへの誤った理解によって発生した**カーネルモデルの欠陥を実行時検証によって発見できた**

モデル検査の課題

状態空間爆発が起きないように**制限**を課せられたモデルに対しての検証しかできず...
例）CPUの数は2つまで、データ構造の大胆な抽象化、など

⑥今後の展望

- よりスケールする形式手法と本手法を統合
- eBPFプログラムのモデルから実装への変換を自動化