Time to Clean Your Test Objectives

Michaël Marcozzi , Sébastien Bardin, Nikolai Kosmatov,
Mike Papadakis, Virgile Prevosto, Loïc Correnson

ICSE ’18, May 27-June 3, 2018, Gothenburg, Sweden

発表者　吉富　楓雅

目次

- 背景と目的
- LCleanによるアプローチ
- リサーチクエスチョンと結果
- まとめと所感

要約

テストでは効率的なテストスイートが求められるが，実行不可能・冗長な，所謂汚染されたテスト目標によって効率性が低下する．本研究では汚染されたテスト目標を除去する手法を提案し，その性能について既存のツールと比較を行う．

背景と目的

ソフトウェアテストとテスト目標

- ソフトウェアテストはソフトウェアの欠陥を発見するための主要な方法
- テスターは**効率的なテストスイート**を構築する必要がある
  - **テストスイート**：ソフトウェアテストの目的ごとに複数のテストケースをまとめたもの
  - **テスト目標(Test Objectives)**を定義する必要がある
    - テスト目標：ソースコード内の各分岐条件
- 現在は**カバレッジ基準**に頼っている

ホワイトボックスカバレッジ基準

- カバレッジ(網羅)基準：テストの入力によってカバーされるべき目標を提供する道標
  - ホワイトボックスの基準では，目標はテスト内の**プログラムの構文構造**である
    - 分岐網羅，機能網羅など
- **実行不可能・冗長な(汚染された)テスト目標によって，入力の選択を誤る可能性がある**

実行不可能・冗長なテスト目標

実行不可能

```
// Branch coverage criterion
int program (int input) {
  // obj1: input<0   obj2: input>=0
  if (input<0) return -1;
  // obj3: input==0   obj4: input!=0
  if (input==0) return 0;
  // obj5: input>0   obj6: input<=0
  if (input>0) { … } }
```

obj6: input <= 0
は実行不可能

冗長

```
// Branch coverage criterion
int program (int input) {
  // obj1: input<0   obj2: input>=0
  if (input<0) return -1;
  // obj3: input==0   obj4: input!=0
  if (input==0) return 0;
  // obj5: input>0   obj6: input<=0
  if (input>0) { … } }
```

obj6: input <= 0は
obj2 input >= 0を包含する

obj4: input != 0と
obj5 input > 0は重複している

https://www.icse2018.org/event/icse-2018-technical-papers-time-to-clean-your-test-objectives

問題点

- **全ての汚染されたテスト目標を検出・除去するのは不可能**
  - 汚染しているかどうかを証明するのは至難の業
    - 複雑なプログラム解析が必要となる
- **汚染されたテスト目標を除去するのに時間がかかる**
  - SQLite equivの汚染されたテスト目標を除去するのに手作業だと2.5年もかかる[1]

目的

冗長・実行不可能な汚染されたテスト目標を
広範囲で検出・除去する

LCleanによるアプローチ

LCleanとは

- **全てのテスト目標を処理しながら，汚染されたテスト目標を検出し除去するツール**
  - 本研究ではOpenSSL等の14の大規模なCプログラム(222 locs; 22万行)で実行した
- **重複率の高い対象を検出することが可能**

ラベルをアサーションで置き換える

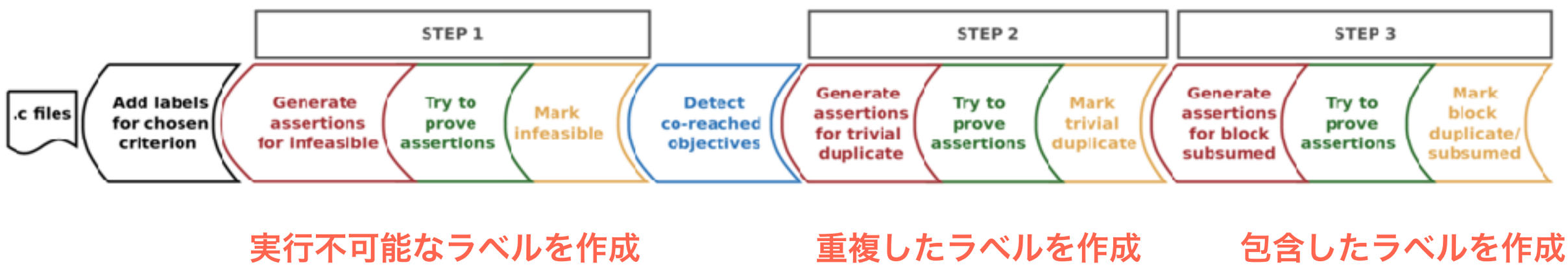
```
int function(int input) { …
// obj1: input<0
if (input<0) return -1;
… }
```

ラベル

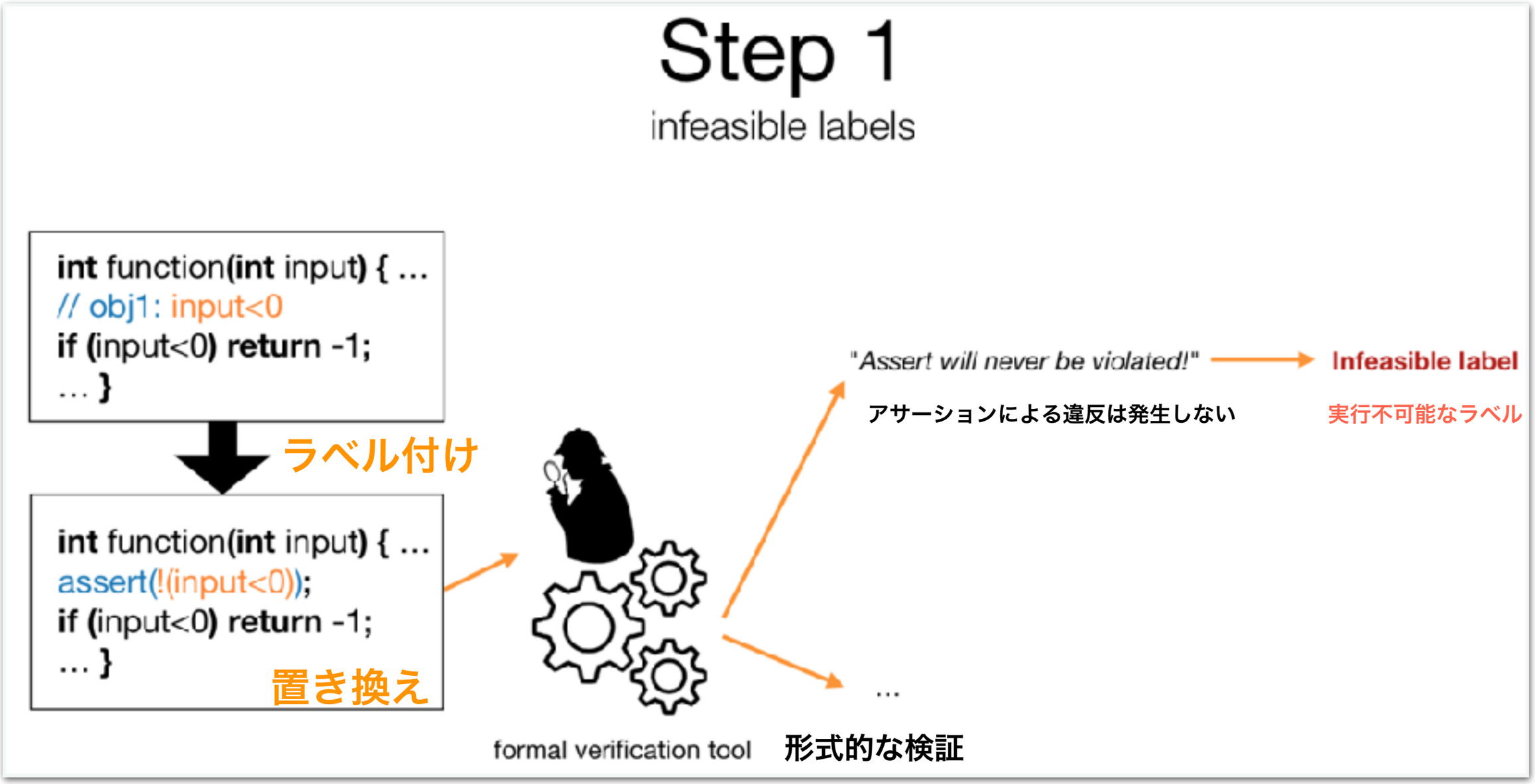
```
int function(int input) { …
assert(!(input<0));
if (input<0) return -1;
… }
```

アサーション

LCleanのプロセス

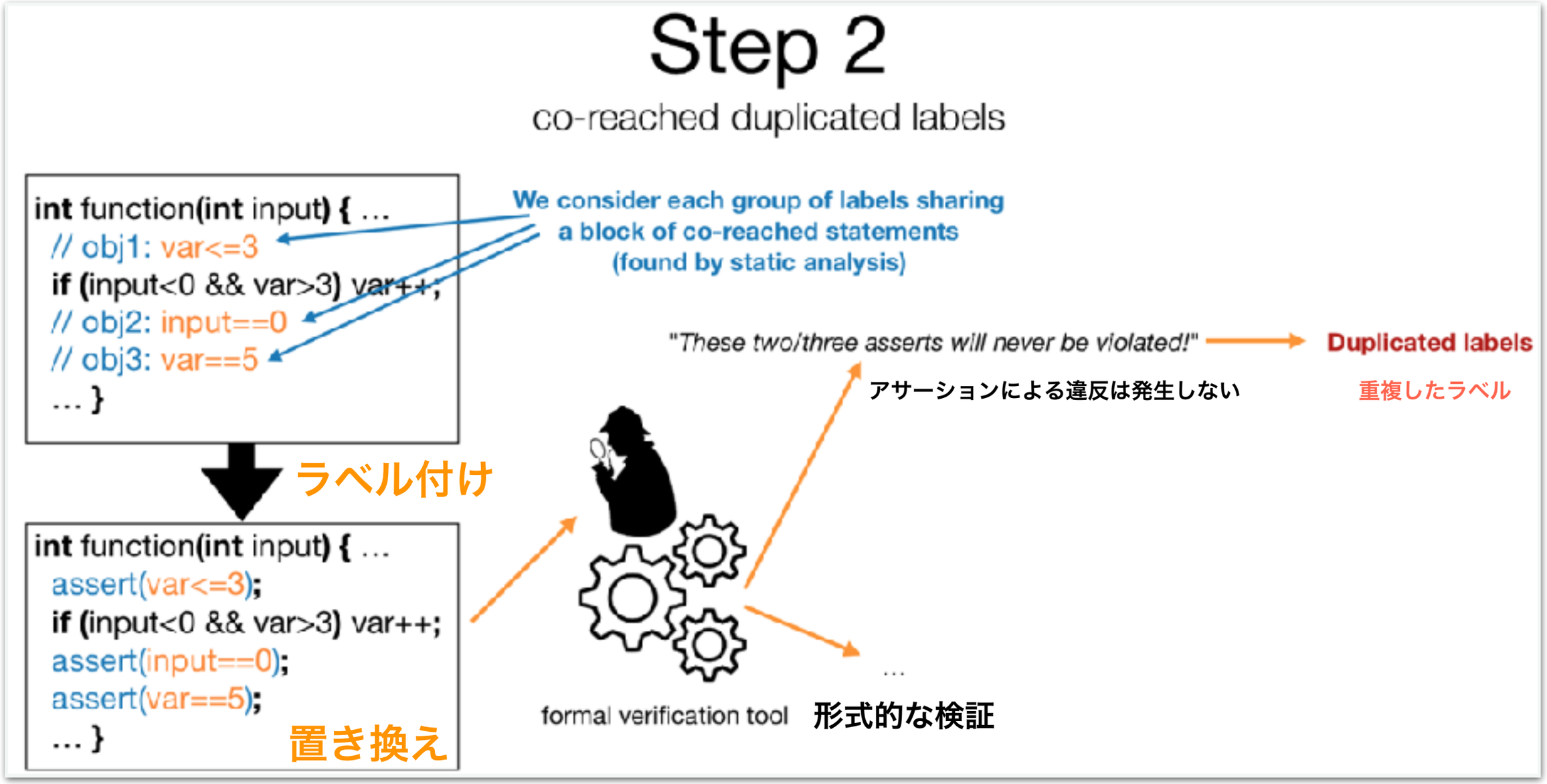


ステップ1



https://www.icse2018.org/event/icse-2018-technical-papers-time-to-clean-your-test-objectives

ステップ2



https://www.icse2018.org/event/icse-2018-technical-papers-time-to-clean-your-test-objectives

ステップ3

Step 3

co-reached subsumed labels

```
int function(int input) { ...
  // obj1: var<=3
  if (input<0 && var>3) var++;
  // obj2: input==0
  // obj3: var==5
  ... }
```

Same as step 2

ラベル付け

```
int function(int input) { ...
int obj1 = (var<=3);
if (input<0 && var>3) var++;
int obj2 = (input==0);
int obj3 = (var==5);
assert(obj1 => obj2); assert(obj1 => obj3);
assert(obj2 => obj1); assert(obj2 => obj3);
assert(obj3 => obj1); assert(obj3 => obj2);
  ... }
```

置き換え

"This assert will never be violated!" → Subsumed label

アサーションによる違反は発生しない　包含したラベル

...

formal verification tool　形式的な検証

Much more proofs than at step 2!

https://www.icse2018.org/event/icse-2018-technical-papers-time-to-clean-your-test-objectives

リサーチクエスチョンと結果

リサーチクエスチョン

1. **LCleanによるアプローチは効率的か？**

   * Siemens, Sjeng, OpenSSL, Gzip, SQLiteに対して実行

2. **アサーションによって最適化すると計算時間が改善されるのか？**

3. **LCleanと似通ったアプローチとの違いは何か？**

   * LUncov, TCE, mutant classificなど

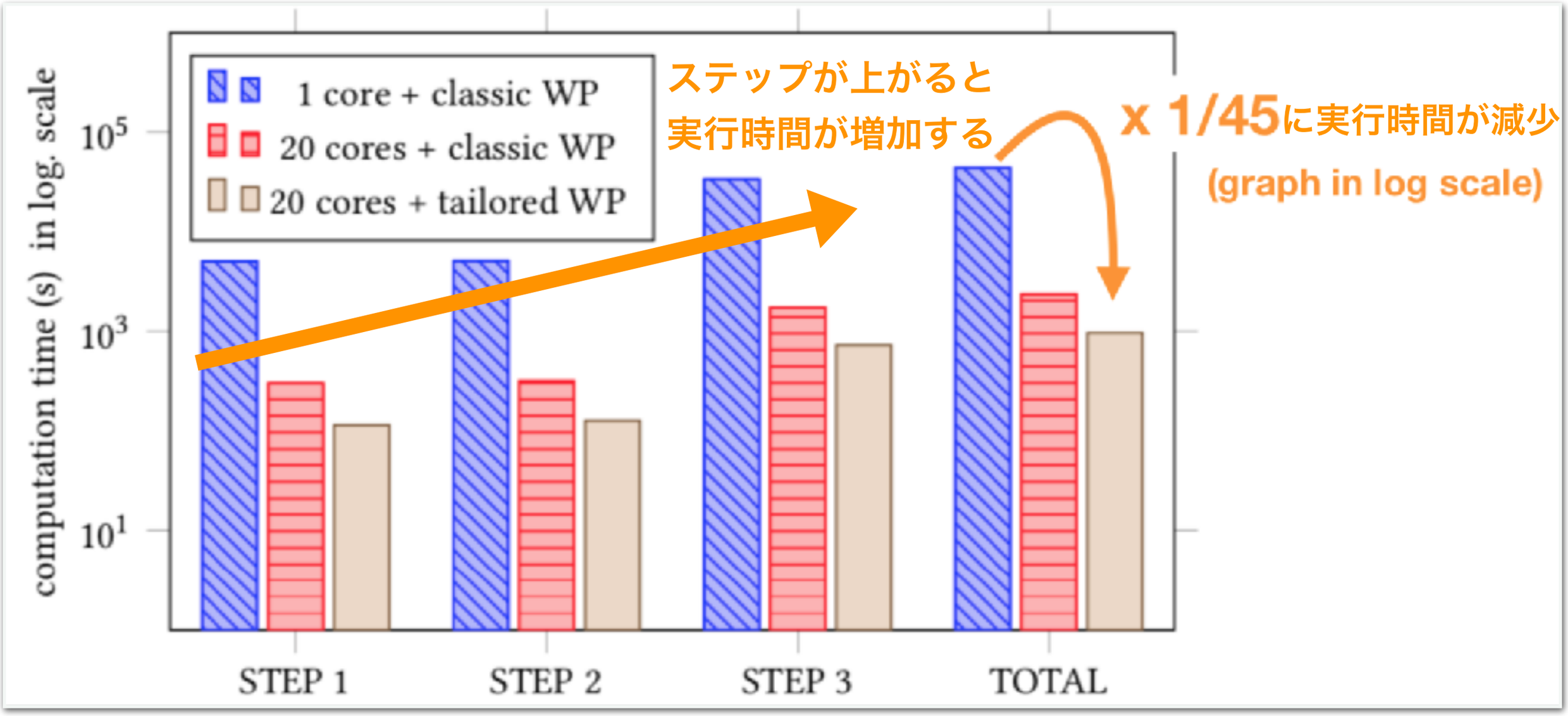
RQ1の結果例(sqlite)

Benchmark	Labels	STEP 1		STEP 2		STEP 3			TOTAL			Criterion
		marked as infeasible	*time*	*marked as duplicate*	*time*	*marked as duplicate*	*marked as subsumed*	*time*	*marked as polluting*		*time*	
									ratio	*%*		
siemens	654	0	35s	0	38s	2	41	83s	43/654	7%	156s	CC
(agg. 7 programs)	666	20	36s	0	40s	0	16	78s	36/666	5%	154s	MCC
3210 locs	654	1	37s	0	39s	18	17	77s	36/654	6%	153s	GACC
	3543	37	114s	123	126s	134	336	723s	630/3543	18%	963s	WM
openssl	1022	28	67s	3	67s	4	57	391s	92/1022	9%	525s	CC
(agg. 4 programs)	1166	134	77s	0	83s	2	24	294s	160/1166	14%	454s	MCC
4596 locs	1022	29	70s	0	81s	30	24	324s	83/1022	8%	475s	GACC
	4978	252	356s	270	372s	200	326	4214s	1048/4978	21%	5122s	WM
gzip	1670	23	149s	5	152s	19	54	578s	101/1670	6%	879s	CC
7569 locs	1726	44	170s	5	171s	11	34	628s	94/1726	5%	969s	MCC
	1670	31	154s	5	156s	43	34	555s	113/1670	7%	865s	GACC
	12270	267	1038s	942	1210s	542	895	10029s	2646/12270	22%	12277s	WM
sjeng	4090	34	351s	15	354s	82	215	798s	346/4090	8%	1503s	CC
14070 locs	4746	358	417s	9	436s	34	26	1912s	427/4746	9%	2765s	MCC
	4090	35	349s	15	353s	82	210	751s	342/4090	8%	1453s	GACC
	25722	353	5950s	483	4791s	640	706	19586s	2182/25722	8%	31478s	WM
sqlite	20202	120	1907s	3	1416s	130	456	4646s	709/20202	4%	7969	CC
196888 locs	21852	394	2295s	0	1902s	178	255	11958s	827/21852	4%	16155	MCC
	20202	129	2065s	0	1613s	803	223	4773s	1155/20202	6%	8451	GACC
	90240	878	18104s	3688	13571s	2962	6164	216140s	13692/90240	15%	247815s	WM
TOTAL	27638	205	2509s	26	2027s	237	823	6496s	**1291/27638**	**5%**	**3h3m52**	**CC**
226333 locs	30156	950	2995s	14	2632s	225	355	14870s	**1544/30156**	**5%**	**5h41m37**	**MCC**
	27638	225	2675s	20	2242s	976	508	6480s	**1729/27638**	**6%**	**3h9m57**	**GACC**
	136753	1787	25562s	5506	20070s	4478	8427	250692s	**20198/136753**	**15%**	**82h18m44**	**WM**
	222185	**3167**	**9h22m21**	**5566**	**7h29m31**	**5916**	**10113**	**77h22m18**	**24762/222185**	**11%**	**94h14m10**	**TOTAL**

手動では１年以上かかるが，LCleanでは69(5+4+60)時間で検出できる

約25,000の有害なラベルを検出できた

RQ2



WP:ACSLアサーションが成立することを証明するため,
弱い前提条件(weakest-precondition)の計算を実装するプラグイン

RQ3

Criterion	LUncov		LClean (step 1)		LClean (all steps)	
	marked	*time*	*marked*	*time*	*marked*	*time*
CC	4/162	97s	4/162	12s	51/162	46s
MCC	30/203	125s	30/203	15s	51/203	53s
WM	84/905	801s	41/905	75s	385/905	463s
TOTAL	**9%** (×1)	**17m3s** (×1)	**6%** (÷1.6)	**1m42s** (÷10)	**38%** (×4.2)	**9m22s** (÷1.8)

LClean vs LUncov

Criterion	Dynamic Detection					LClean				
	possibly infeasible	*possibly duplicate*	*possibly subsumed*	*total ratio for possibly polluting*	*time*	*marked as infeasible*	*marked as duplicate*	*marked as subsumed*	*total ratio for marked as polluting*	*time*
CC	37/654	243/654	230/654	80% (510/654)	3132s	0/654	2/654	41/654	7% (43/654)	156s
MCC	76/666	221/666	215/666	77% (512/666)	3142s	20/666	0/666	16/666	5% (36/666)	154s
GACC	46/654	249/654	212/654	78% (507/654)	3134s	1/654	18/654	17/654	6% (36/654)	153s
WM	386/3543	2327/3543	641/3543	95% (3354/3543)	8399s	37/3543	257/3543	336/3543	18% (630/3543)	963s
TOTAL	545/5517	3040/5517	1298/5517	**89%** (4883/5517)	**4h56m47** (×12)	58/5517	277/5517	410/5517	**14%** (745/5517)	**23m46s** (×1)

LClean vs Mutant Classification

他検出ツールに比べて，検出率・検出時間ともにLCleanで向上していることが示された

まとめと所感

まとめ

- 実行不可能・冗長なテスト目標によって，入力の選択を誤る可能性がある
  - 本研究では全てのテスト目標を処理しながら，汚染されたテスト目標を検出し除去するツールを提案した
- 提案手法では，手作業に比べると大幅に早く汚染されたテスト目標を除去することができた

所感

- 汚染されたテスト目標は発見しにくいことがわかった
  - 提案手法でも69時間かかる…
- 様々な既存手法と比較していたので，性能がどの程度上がったのか比較しやすかった