SOFTWARE ENGINEERING
COMPUTER SCIENCE

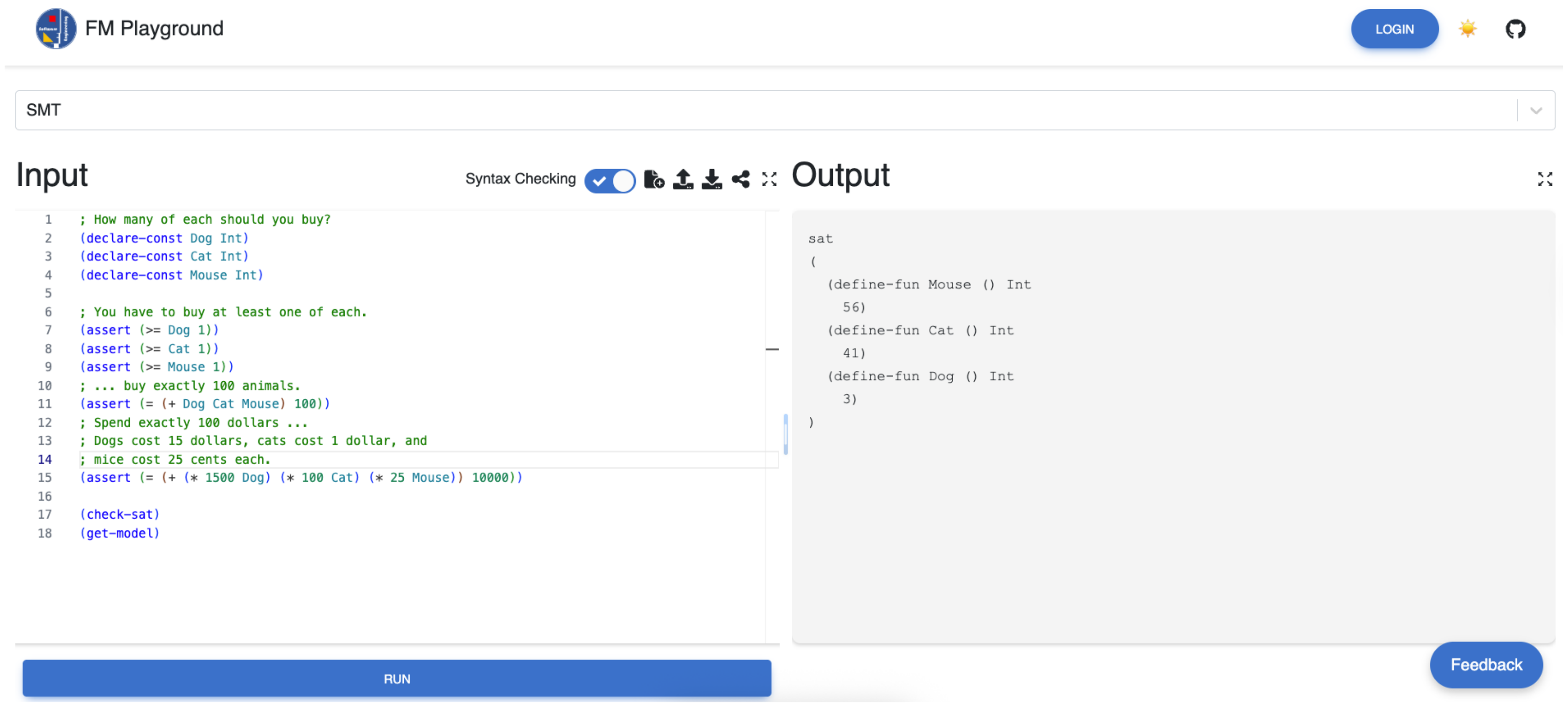
Bauhaus-Universität Weimar

On Writing SMT-LIB Scripts: Metrics and a New Dataset

Soaibuzzaman and Jan Oliver Ringert

SMT 2025

August 10, 2025, Glasgow, UK



```
; How many of each should you buy?
(declare-const Dog Int)
(declare-const Cat Int)
(declare-const Mouse Int)

; You have to buy at least one of each.
(assert (>= Dog 1))
(assert (>= Cat 1))
(assert (>= Mouse 1))
; ... buy exactly 100 animals.
(assert (= (+ Dog Cat Mouse) 100))
; Spend exactly 100 dollars ...
; Dogs cost 15 dollars, cats cost 1 dollar, and
; mice cost 25 cents each.
(assert (= (+ (* 1500 Dog) (* 100 Cat) (* 25 Mouse)) 10000))

(check-sat)
(get-model)
```

```
sat
(
  (define-fun Mouse () Int
    56)
  (define-fun Cat () Int
    41)
  (define-fun Dog () Int
    3)
)
```

Motivation

- Popular datasets have been collected to benchmark SMT solvers
- But: **little is known** about how people write SMT-LIB scripts, especially novices
  - Gap in understanding of the user experience
  - Challenges and behaviors
- Our findings can inform better **tool support and teaching materials**

SMT-LIB

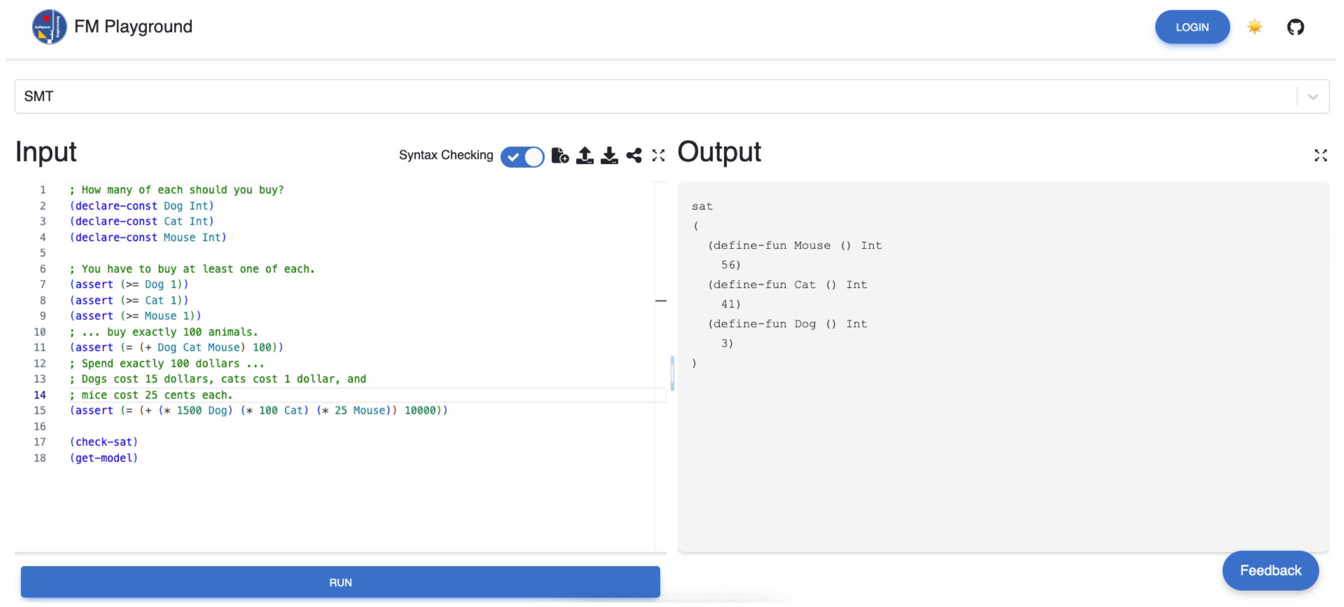
- SMT-LIB is the standard input format for SMT solvers
- Widely used in verification, modeling, and synthesis
- Disclaimer: APIs are another popular way to express SMT problems

```
(set-logic UF)
; datatype for people in mansion
(declare-datatype Person (Agatha Butler Charles))
(declare-const Killer Person)
; a function/predicate to represent killing
(declare-fun killed (Person Person) Bool)
(declare-fun hates (Person Person) Bool)

; Charles hates no one that Agatha hates
(assert (forall ((x Person)) (=> (hates Agatha x)
                         (not (hates Charles x)))))
; ...
(assert (killed Killer Agatha))
(check-sat)
(get-model)
```

Formal Methods Playground

- A web app for writing and analyzing specifications in various modeling and specification languages
- Provides basic language support for SMT-LIB
- Offers storage of permalinks, histories, etc
- Try it at: https://play.formal-methods.net



```
; How many of each should you buy?
(declare-const Dog Int)
(declare-const Cat Int)
(declare-const Mouse Int)

; You have to buy at least one of each.
(assert (>= Dog 1))
(assert (>= Cat 1))
(assert (>= Mouse 1))
; ... buy exactly 100 animals.
(assert (= (+ Dog Cat Mouse) 100))
; Spend exactly 100 dollars ...
; Dogs cost 15 dollars, cats cost 1 dollar, and
; mice cost 25 cents each.
(assert (= (+ (* 1500 Dog) (* 100 Cat) (* 25 Mouse)) 10000))

(check-sat)
(get-model)
```

```
sat
(
  (define-fun Mouse () Int
    56)
  (define-fun Cat () Int
    41)
  (define-fun Dog () Int
    3)
)
```

Bauhaus-Universität Weimar

Our Contribution

- FMP$_{smt}$ Dataset: a collection of 18,133 SMT-LIB scripts from the Formal Methods Playground
  - 2,415 fine-grained editing paths (revision histories)
  - Often starts from a blank canvas
  - Scripts created by MSc students (Computer Science & Digital Engineering) from >= 2 Universities
- Analysis
  - Structural metrics
  - Syntactic + semantic script evolution
  - Error patterns and edit distances

Research Questions

- **RQ1:** What are the key characteristics of the FMP_{smt} dataset?
- **RQ2:** Where do users most commonly introduce syntactic errors?
- **RQ3:** How do consecutive SMT-LIB scripts differ?
- **RQ4:** How large are the edit distances between consecutive scripts?
- **RQ5:** How do users fix errors over multiple edit steps?

RQ1: Dataset Characteristics

- **Sizes**:
  - Median ELOC: 26
  - Median operator nesting depth in asserts: 5
- **38 logics** used (typos included)
- **Execution times**: most < 0.03s

- **Edit paths**:
  - 2,415 paths, median length = 6
  - 58% have ≥5 revisions
- **Error:**
  - 59% of edit paths contain at least one invalid script

	Q1	Median	Q3	Max
ELOC	10	26	65	1,531
Max Nesting Depth	5	5	6	42
# assert commands	2	7	23	287
# declare-const commands	1	4	14	371
# declare-fun commands	0	0	3	299
Time taken (s) (timeout of 600s)	0.02	0.02	0.03	318.27

RQ2: Syntax Errors

- ~40% of all scripts have syntax errors
- **Most frequent:**
  - Unknown constant (50%)
- **Most error-prone commands**: get-value, eval, declare-fun

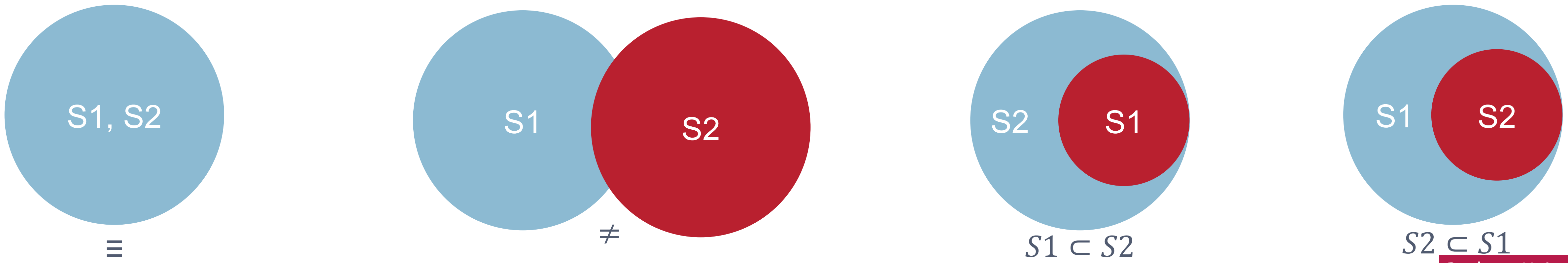
command	abs. # error	# total elements	rel. % of command
assert	**35,132**	**319,049**	11.01%
declare-const	10,920	224,947	4.85%
declare-fun	6,870	46,489	14.78%
get-value	5,404	17,648	**30.62%**
define-fun	3,237	22,548	14.36%
get-model	2,071	12,762	16.23%
declare-datatype	1,787	18,152	9.84%
check-sat	134	28,318	0.47%
eval	104	372	27.96%
quantifiers	49	1,342	3.65%

Category	Count	Percentage
Unknown constant *constant_name*	**35,509**	**50.13%**
Invalid constant declaration *sort_name*	6,941	9.80%
Parsing function declaration *sort_name*	5,070	7.16%
Logic does not support	4,325	6.11%
Invalid declaration	3,629	5.12%
Model is not available	3,506	4.95%
Invalid sort	2,921	4.12%
Unknown sort *	2,587	3.65%
Unexpected character	928	1.31%
Invalid function decleration	856	1.21%

Bauhaus-Universität Weimar

RQ3: Semantic Comparison of SMT-LIB Scripts

- Verifies the semantic entailment between the assertion sets collected from two compared scripts
  - $S1 \vDash S2$ and $S2 \vDash S1$
- Naive
  - Oblivious to variable renaming
  - Ignores push and pop scopes
  - Less intuitive for scripts that contain unsatisfiable assertions



RQ3: How Scripts Change

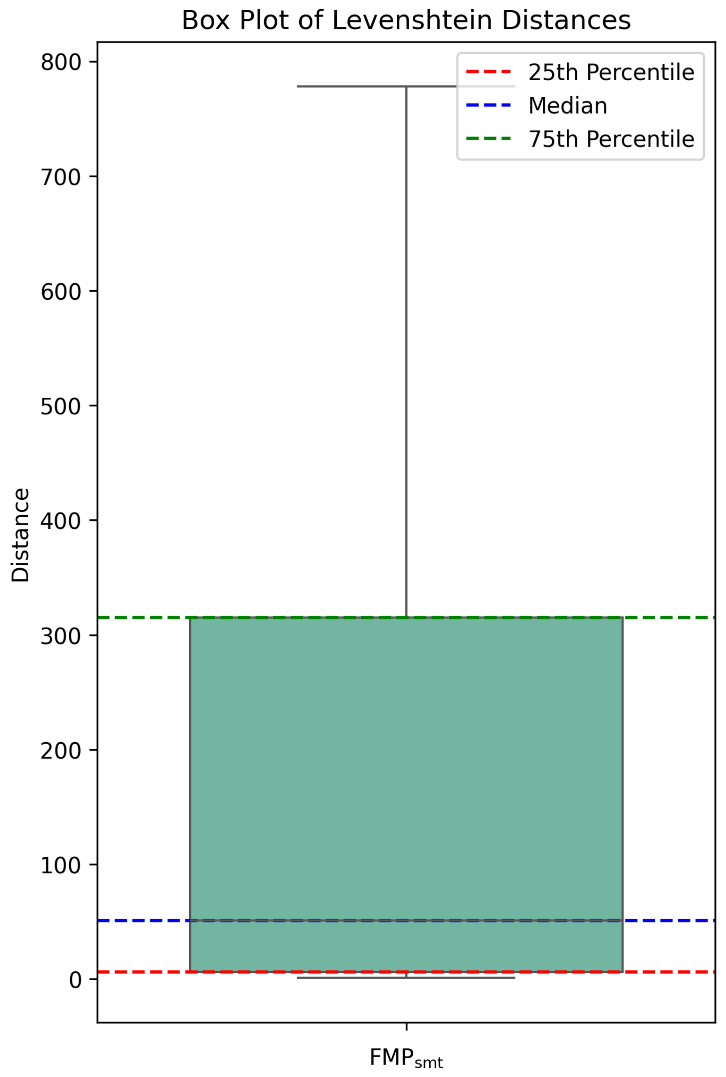
- 4.3K consecutive edits were **identical** (users re-run same script)
- Consecutive semantic relationships (only if no syntax error in both scripts):
  - 24% are equivalent
  - 11% are refinements
  - 10% are incomparable
- Users often **refine** or **weaken** scripts

	Syntactically	**Semantically**			
	Identical	**$\equiv$**	**$\not\equiv$**	**S1 $\subset$ S2**	**S2 $\subset$ S1**
Consecutive	4,319	6,332	2,805	1,149	1,748
Non-Consecutive	2,121	877	2,125	908	1,542

Bauhaus-Universität Weimar

RQ4: Edit Distance

- Median Levenshtein distance: 51 characters
- Most edits are small and local
- Long tail (max = 38,659):
  - Some major rewrites
  - Starts over completely



Bauhaus-Universität Weimar

RQ5: Fixing Errors

- Most syntax errors fixed in 1–3 steps
- Most UNSAT-to-SAT edits also fixed quickly (median = 1)
- Indicates trial-and-error debugging with occasional struggle
  - Max steps to fix syntax error: 52
  - Max steps from UNSAT-to-SAT: 58

Key Findings & Conclusion

- Writing SMT-LIB scripts is error-prone for novices
  - tooling matters!
- Edits are mostly small
  - Suitable for interactive feedback
- Many errors could be mitigated with:
  - Context-aware editors
  - Scoping + reference checking
  - Better error messages
- Data availability:
  - Formal Methods Playground (public, open source)
  - Dataset updated on Zenodo
- Language Support (ongoing):
  - https://github.com/se-buw/smt-langium

Questions?

Bauhaus-Universität Weimar