# An SMT-Based Concolic Testing Tool for Logic Programs

Sophie Fortz
sophie.fortz@unamur.be
@FortzSophie

FLOPS 2020, September 16th


# An SMT-Based Concolic Testing Tool for Logic Programs

Fred Mesnard
Etienne Payet
Gilles Perrouin
Wim Vanhoof
German Vidal







# Concolic testing for Logic Programs

Linear semantics :

*Ströder, T., Emmes, F., Schneider-Kamp, P., Giesl, J., Fuhs, C.: A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog. In: LOPSTR'11. pp. 237–252. Springer LNCS 7225 (2011)*

First Approach :

*Mesnard, F., Payet, É., Vidal, G.: Concolic testing in logic programming. TPLP 15(4-5), 711–725 (2015). https://doi.org/10.1017/S1471068415000332*

✗ **Scalability issues**

✗ **Incompleteness**

# Scalability Issue

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Contest |
|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.0273 |
| Nat | 2 | nat(0) | 1 | 5 | 0.1554 |
| Nat | 2 | nat(0) | 1 | 50 | 19.5678 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 0.7096 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 4.4820 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | Crash |
| Activities | 38 | what_to_do_today( sunday,sunny,wash_your _car) | 3 | 2 | Timeout |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | Timeout |
| Family | 48 | parent(dicky,X) | 1 | 1 | 64.1838 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.4701 |

https://github.com/Anniepoo/prolog-examples

# Scalability Issue

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Contest |
|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.0273 |
| Nat | 2 | nat(0) | 1 | 5 | 0.1554 |
| Nat | 2 | nat(0) | 1 | 50 | 19.5678 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 0.7096 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 4.4820 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | Crash |
| Activities | 38 | what_to_do_today( sunday,sunny,wash_your _car) | 3 | 2 | Timeout |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | Timeout |
| Family | 48 | parent(dicky,X) | 1 | 1 | 64.1838 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.4701 |

https://github.com/Anniepoo/prolog-examples

# Scalability Issue

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Contest |
|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.0273 |
| Nat | 2 | nat(0) | 1 | 5 | 0.1554 |
| Nat | 2 | nat(0) | 1 | 50 | 19.5678 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 0.7096 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 4.4820 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | Crash |
| Activities | 38 | what_to_do_today( sunday,sunny,wash_your _car) | 3 | 2 | Timeout |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | Timeout |
| Family | 48 | parent(dicky,X) | 1 | 1 | 64.1838 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.4701 |

https://github.com/Anniepoo/prolog-examples

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Which test cases will be produced by a Concolic execution?

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a)}

Which test cases will be produced by a Concolic execution?

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a)}

Which test cases will be produced by a Concolic execution?

( ) ⇒ { } ⇒ ∅

(1) ⇒ {p(a)} ⇒ ∅

(2) ⇒ {p(X)} ⇒ p(b)

(1,2) ⇒ {p(a),p(X)} ⇒ p(a)

selective unification problem

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a), p(b)}

Which test cases will be produced by a Concolic execution?

( ) ⇒ { } ⇒ ∅

(1) ⇒ {p(a)} ⇒ ∅

(2) ⇒ {p(X)} ⇒ p(b)

(1,2) ⇒ {p(a),p(X)} ⇒ p(a)

selective unification problem

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a), p(b)}

Which test cases will be produced by a Concolic execution?

( ) ⇒ { } ⇒ ∅

(1) ⇒ {p(a)} ⇒ ∅

(2) ⇒ {p(X)} ⇒ p(b)

(1,2) ⇒ {p(a),p(X)} ⇒ p(a)

( ) ⇒ { } ⇒ ?

(3) ⇒ {q(b)} ⇒ p(b)

selective unification problem

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a), p(b)}

Which test cases will be produced by a Concolic execution?

( ) ⇒ { } ⇒ ∅

(1) ⇒ {p(a)} ⇒ ∅

(2) ⇒ {p(X)} ⇒ p(b)

(1,2) ⇒ {p(a),p(X)} ⇒ p(a)

( ) ⇒ { } ⇒ **p(a)**

(3) ⇒ {q(b)} ⇒ p(b)

selective unification problem

# Completeness Issue

(1) p(a).

(2) p(X) ← q(X).

(3) q(b).

Initial Goal: p(a)

Produced test cases: {p(a), p(b), p(c)}

Which test cases will be produced by a Concolic execution?

( ) ⇒ { } ⇒ ∅

(1) ⇒ {p(a)} ⇒ ∅

(2) ⇒ {p(X)} ⇒ p(b)

(1,2) ⇒ {p(a),p(X)} ⇒ p(a)

( ) ⇒ { } ⇒ ~~p(a)~~ p(c)

(3) ⇒ {q(b)} ⇒ p(b)

selective unification problem

# Contributions

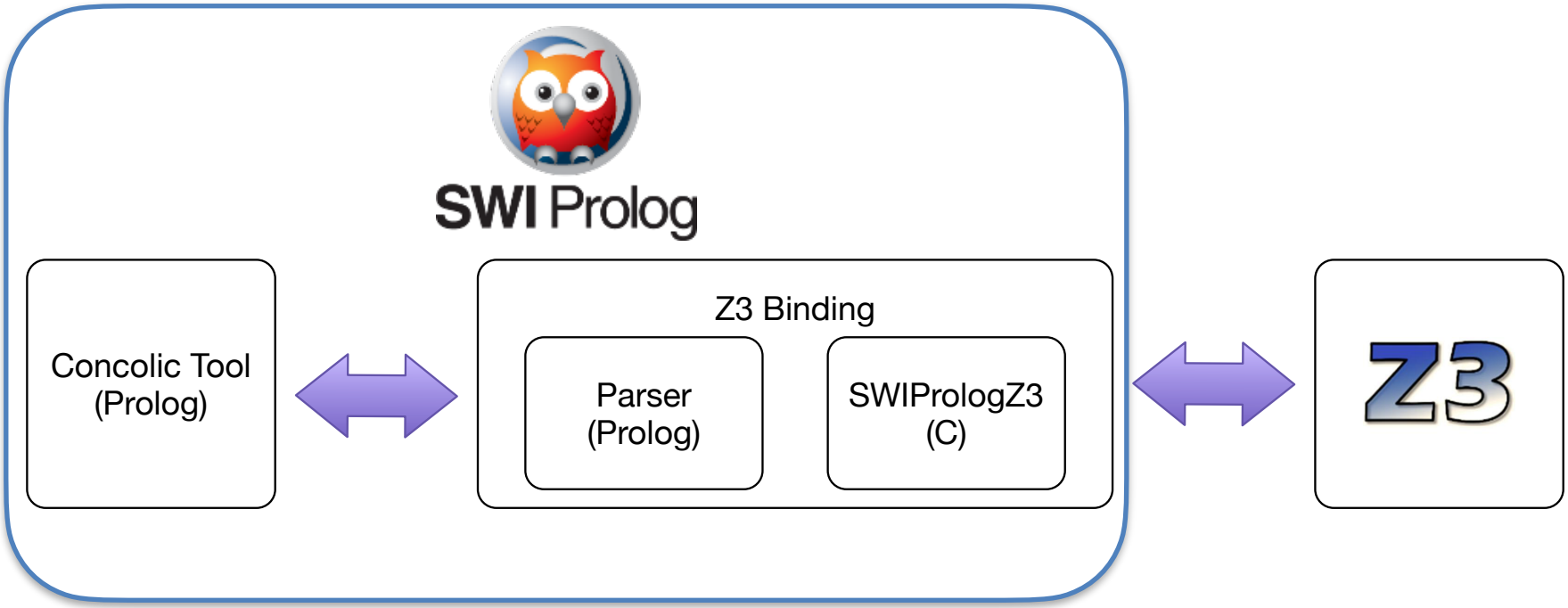
- Use of constraints to represent negative information

In our example:

$$p(X) \neq p(a) \wedge \exists Y\ (p(X) = p(Y)) \wedge q(X) \neq q(b)$$

- Use a mature constraint solver to improve performance of concolic testing

# Solution's Architecture




https://github.com/sfortz/Pl_Concolic_Testing

# Experimental Results

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Concolic | Time Contest | #TCs Concolic | #TCs Contest |
|---|---|---|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.050 | 0.0273 | 3 | 4 |
| Nat | 2 | nat(0) | 1 | 5 | 0.0897 | 0.1554 | 7 | 12 |
| Nat | 2 | nat(0) | 1 | 50 | 1.6752 | 19.5678 | 52 | 102 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 1.4517 | 0.7096 | 9 | 9 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 1.3255 | 4.4820 | 9 | 9 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | 1.3211 | Crash | 9 | N/A |
| Activities | 38 | what_to_do_today( sunday,sunny,wash_your _car) | 3 | 2 | 6.3257 | Timeout | 122 | N/A |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | 0.0535 | Timeout | 2 | N/A |
| Family | 48 | parent(dicky,X) | 1 | 1 | 20.0305 | 64.1838 | 9 | 19 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.2001 | 0.4701 | 6 | 7 |

https://github.com/Anniepoo/prolog-examples

# Experimental Results

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Concolic | Time Contest | #TCs Concolic | #TCs Contest |
|---|---|---|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.050 | 0.0273 | 3 | 4 |
| Nat | 2 | nat(0) | 1 | 5 | 0.0897 | 0.1554 | 7 | 12 |
| Nat | 2 | nat(0) | 1 | 50 | 1.6752 | 19.5678 | 52 | 102 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 1.4517 | 0.7096 | 9 | 9 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 1.3255 | 4.4820 | 9 | 9 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | 1.3211 | Crash | 9 | N/A |
| Activities | 38 | what_to_do_today(sunday,sunny,wash_your_car) | 3 | 2 | 6.3257 | Timeout | 122 | N/A |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | 0.0535 | Timeout | 2 | N/A |
| Family | 48 | parent(dicky,X) | 1 | 1 | 20.0305 | 64.1838 | 9 | 19 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.2001 | 0.4701 | 6 | 7 |

https://github.com/Anniepoo/prolog-examples

# Experimental Results

| Subject Program | Size | Initial Goal | Ground Args | Max Depth | Time Concolic | Time Contest | #TCs Concolic | #TCs Contest |
|---|---|---|---|---|---|---|---|---|
| Nat | 2 | nat(0) | 1 | 1 | 0.050 | 0.0273 | 3 | 4 |
| Nat | 2 | nat(0) | 1 | 5 | 0.0897 | 0.1554 | 7 | 12 |
| Nat | 2 | nat(0) | 1 | 50 | 1.6752 | 19.5678 | 52 | 102 |
| Generator | 7 | generate(empty,_A,_B) | 1 | 1 | 1.4517 | 0.7096 | 9 | 9 |
| Generator | 7 | generate(empty,T,_B) | 2 | 1 | 1.3255 | 4.4820 | 9 | 9 |
| Generator | 7 | generate(empty,T,H) | 3 | 1 | 1.3211 | Crash | 9 | N/A |
| Activities | 38 | what_to_do_today( sunday,sunny,wash_your _car) | 3 | 2 | 6.3257 | Timeout | 122 | N/A |
| Cannibals | 78 | start(config(3,3,0,0)) | 1 | 2 | 0.0535 | Timeout | 2 | N/A |
| Family | 48 | parent(dicky,X) | 1 | 1 | 20.0305 | 64.1838 | 9 | 19 |
| Monsters and mazes | 113 | base_score(will,grace) | 2 | 2 | 0.2001 | 0.4701 | 6 | 7 |

https://github.com/Anniepoo/prolog-examples

# Related and Future Work


1. *Giantsios, A., Papaspyrou, N., Sagonas, K.: Concolic testing for functional languages. Science of Computer Programming 147, 109–134 (2017)*
2. *Tikovsky, J.R.: Concolic testing of functional logic programs. In: Declarative Programming and Knowledge Management, pp. 169–186. Springer (2017)*
3. *Mesnard, F., Payet, É., Vidal, G.: Concolic Testing in CLP. CoRR abs/2008.00421 (2020), https://arxiv.org/abs/2008.00421*

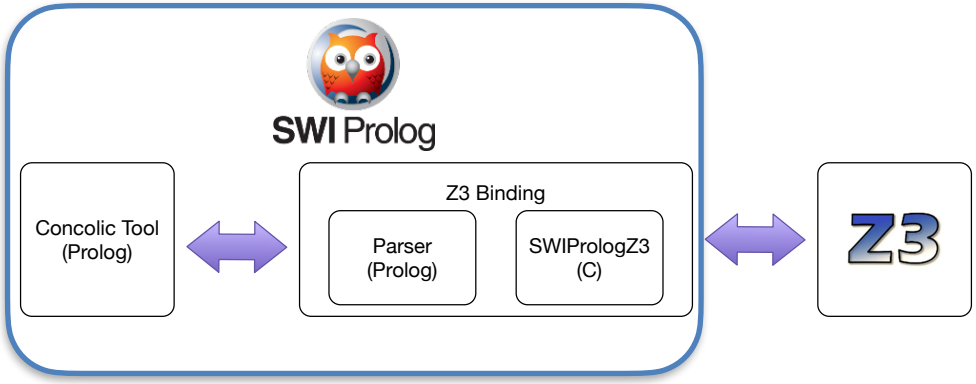
# Wrap Up

## ConTest

*Mesnard, F., Payet, É., Vidal, G.: Concolic testing in logic programming. TPLP 15(4-5), 711–725 (2015). https://doi.org/10.1017/S1471068415000332*

**Scalability issues**

**Incompleteness**




Concolic Testing in Constraint Logic Programming