An Approach for the Verification and Synthesis of Complete Test Generation Algorithms for Finite State Machines

Robert Sachtleben

Agenda

1. unified implementation,
2. mechanised completeness proof, and
3. provably correct implementation

of complete test generation algorithms for finite state machines (FSM)

Motivation

- complete test strategies are of importance in model-based testing MBT
  - high guaranteed test strength
  - well-specified assumptions
- manual verification and implementation are problematic
  - large number of strategies and variants
  - distinct complex algorithms and correctness arguments
  - ambiguities in natural language descriptions

Motivation

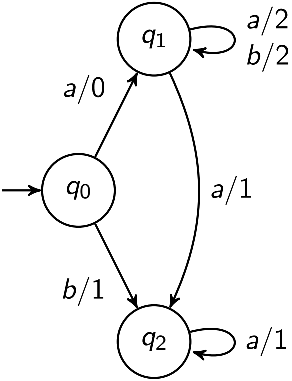
- complete test strategies are of importance in model-based testing MBT
  - high guaranteed test strength
  - well-specified assumptions
- manual verification and implementation are problematic
  - large number of strategies and variants
  - distinct complex algorithms and correctness arguments
  - ambiguities in natural language descriptions
- proposed approach: mechanised proofs and subsequent synthesis of implementations

Model-based Testing using Finite State Machines

Finite State Machines (FSMs)

$$
\begin{aligned}
M_1 &= (Q, q_0, \Sigma_I, \Sigma_O, h_1) \\
Q &= \{q_0, q_1, q_2\} \\
\Sigma_I &= \{a, b\} \\
\Sigma_O &= \{0, 1, 2\} \\
h_1 &= \{(q_0, a, 0, q_1), (q_0, b, 1, q_2), \\
&\qquad (q_1, a, 1, q_2), (q_1, a, 2, q_1), \\
&\qquad (q_1, b, 2, q_1), (q_2, a, q, q_2)\}
\end{aligned}
$$

- nondeterministic
- partially specified
- observable
- minimal



Model-based Testing using FSMs – Models

- Finite State Machines M_1 and M_2
  - both observable and minimal
  - M_1 is the *reference model*
  - M_2 represents the behaviour of the *System under Test* (SUT)
    - Black Box
    - same alphabets as M_1, at most m states
  - *Fault Domain* $\mathcal{F}(M_1, m)$ contains all such M_2

- goal: check whether M_1 are M_2 *language-equivalent*:

$$\mathcal{L}(M_2) = \mathcal{L}(M_1)$$

Definition

M_2 *passes* test suite TS für M_1, denoted $M_2 \sim_{TS} M_1$, if the following holds

$$\mathcal{L}(M_2) \cap TS = \mathcal{L}(M_1) \cap TS$$

Definition

test suite TS is m-complete for reference model M_1 if for *all* $M_2 \in \mathcal{F}(M_1, m)$ it holds that

$$M_2 \sim_{TS} M_1 \Longleftrightarrow \mathcal{L}(M_2) = \mathcal{L}(M_1)$$

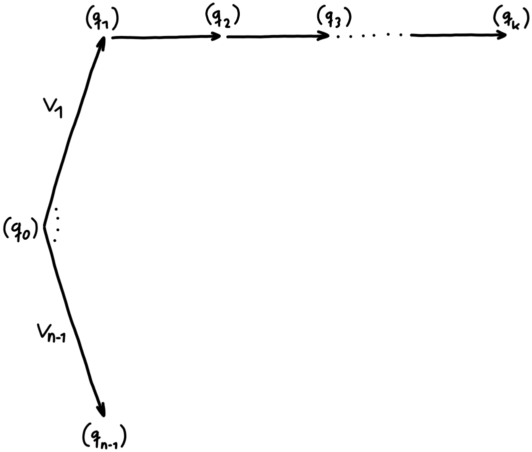
H-Condition / H-Method

TS is m-complete if

H-Condition / H-Method

TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V



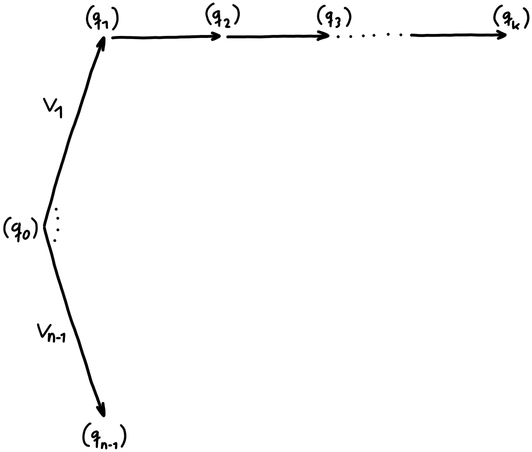
H-Condition / H-Method

TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)

via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



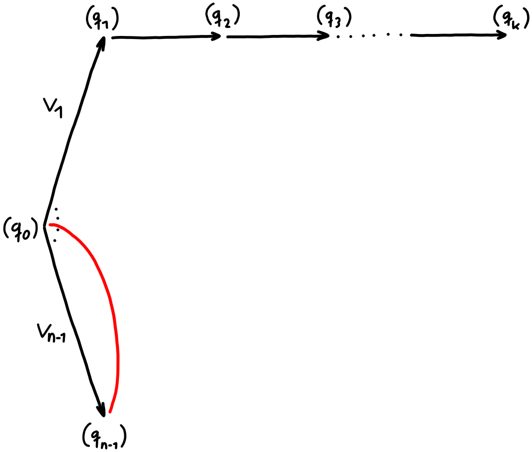
H-Condition / H-Method

TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* *V*

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



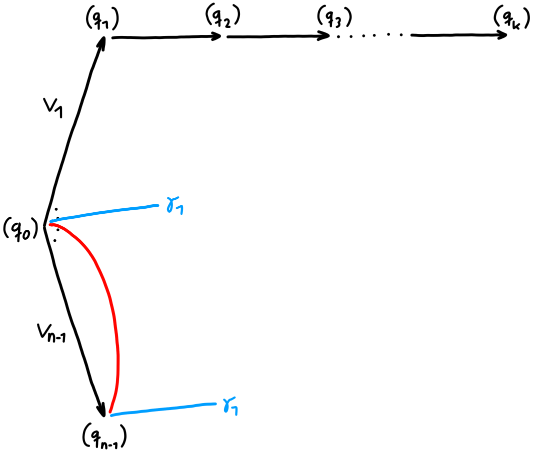
H-Condition / H-Method

TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$

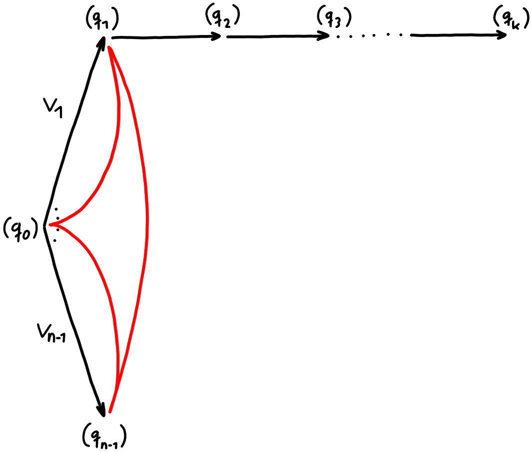


TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



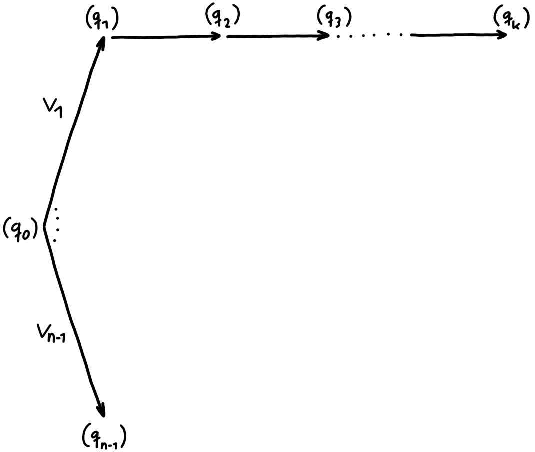
TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



H-Condition / H-Method

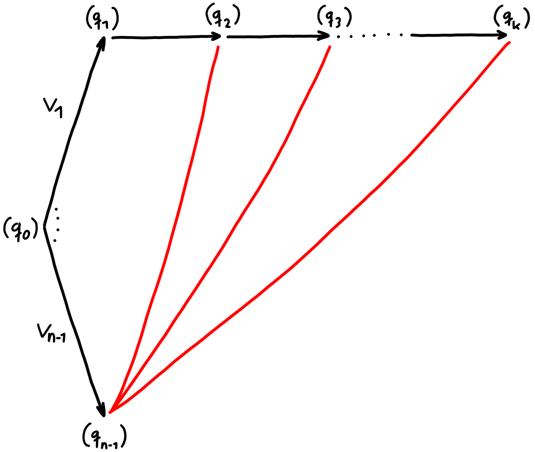
TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



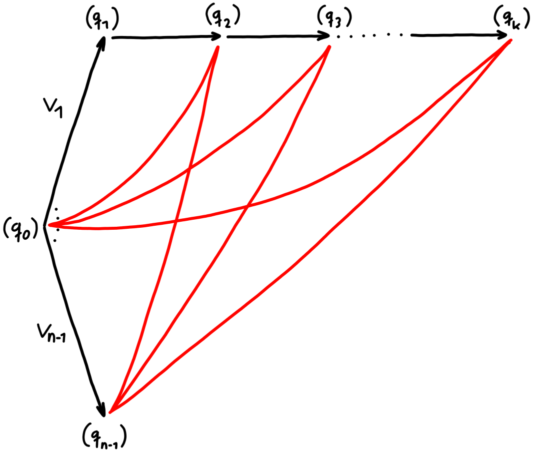
TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* *V*

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



H-Condition / H-Method

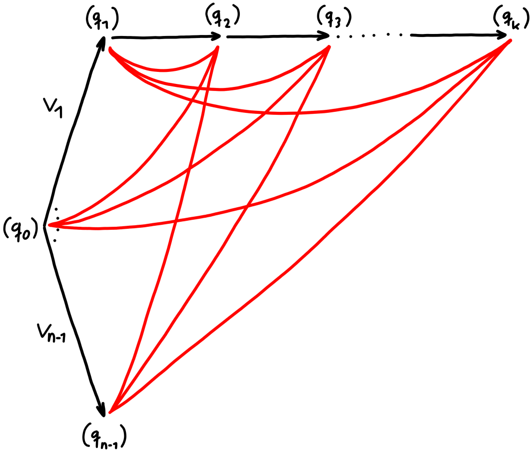
TS is *m*-complete if

1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



TS is *m*-complete if

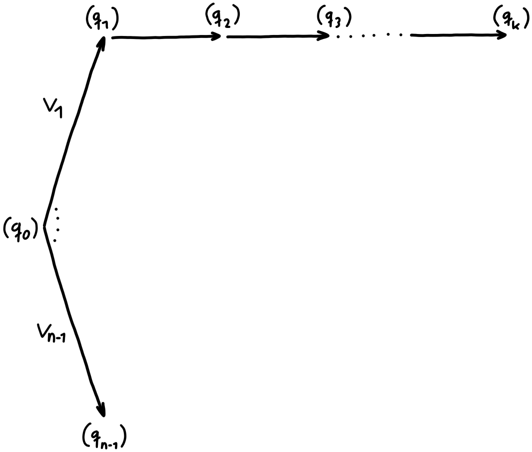
1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

   (C) $(v.\omega', v.\omega)$ for $v \in V, \omega \in \Sigma^{\leq m-n+1}, \omega' \in pref(\omega)$

   via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



H-Condition / H-Method

TS is *m*-complete if

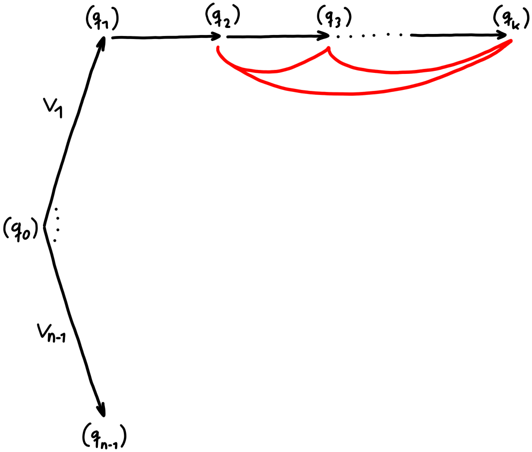
1. $V.\Sigma^{\leq m-n+1} \subseteq TS$
   - for some *state cover* V

2. *TS* preserves divergences of pairs (α, β)
   (A) (v, v') for $v, v' \in V$

   (B) $(v, v'.\omega)$ for $v, v' \in V, \omega \in \Sigma^{\leq m-n+1}$

   (C) $(v.\omega', v.\omega)$ for
   $v \in V, \omega \in \Sigma^{\leq m-n+1}, \omega' \in pref(\omega)$

   via *distinguishing traces* γ, added to *TS* as $\alpha.\gamma, \beta.\gamma$



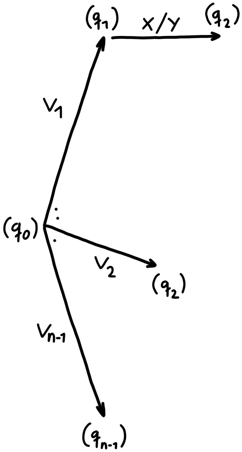
SPY-Condition / SPY-Method

TS is *m*-complete if

SPY-Condition / SPY-Method

TS is m-complete if

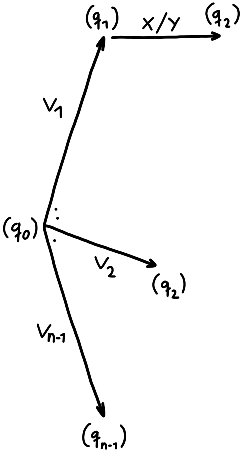
1. $V.\Sigma \subseteq TS$



SPY-Condition / SPY-Method

TS is *m*-complete if

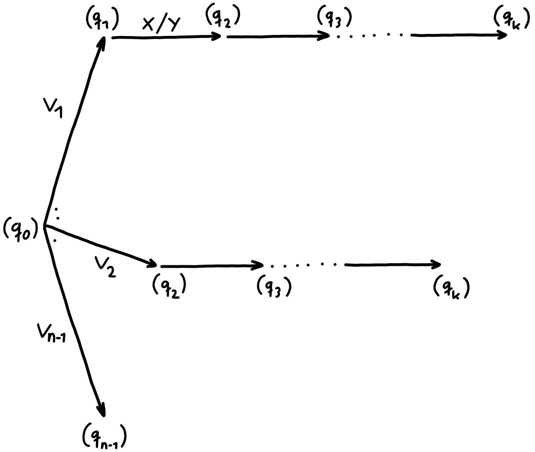
1. $V.\Sigma \subseteq TS$
2. *TS* preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$



SPY-Condition / SPY-Method

TS is *m*-complete if

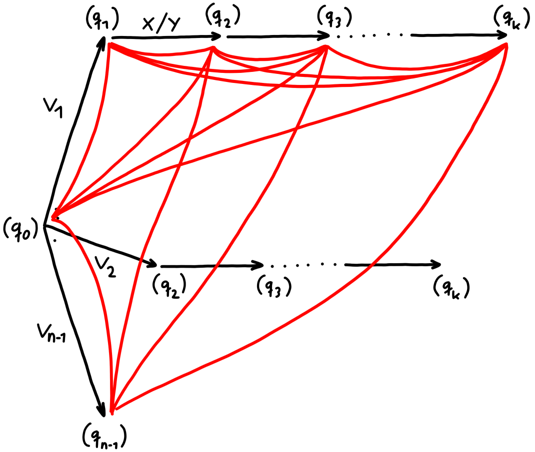
1. $V.\Sigma \subseteq TS$
2. TS preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$



SPY-Condition / SPY-Method

TS is m-complete if

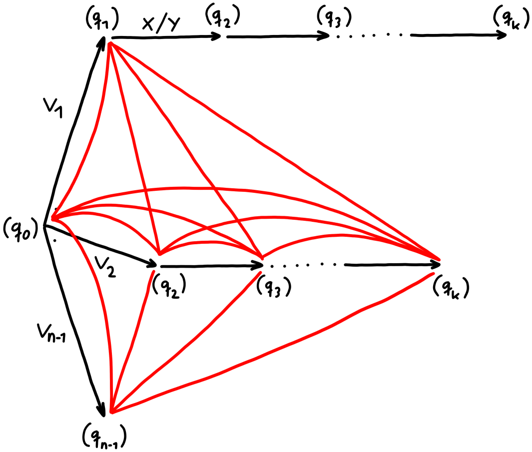
1. $V.\Sigma \subseteq TS$
2. TS preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$



SPY-Condition / SPY-Method

TS is *m*-complete if

1. $V.\Sigma \subseteq TS$
2. TS preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$

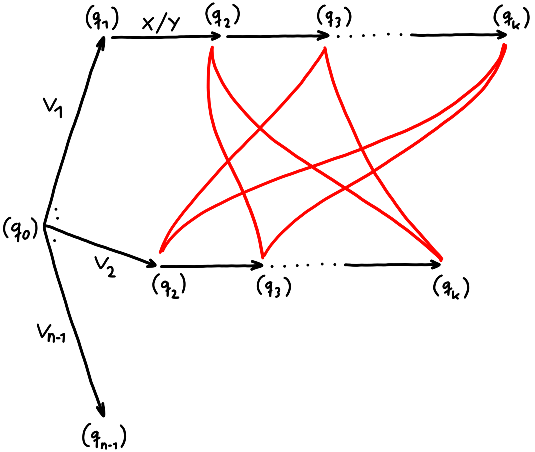


SPY-Condition / SPY-Method

TS is *m*-complete if

1. $V.\Sigma \subseteq TS$

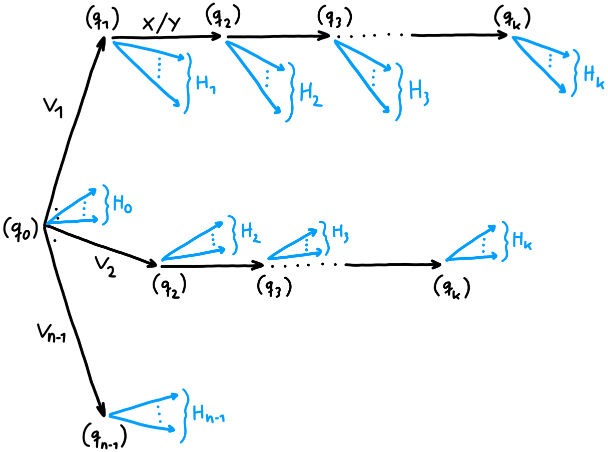
2. TS preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$



SPY-Condition / SPY-Method

TS is *m*-complete if

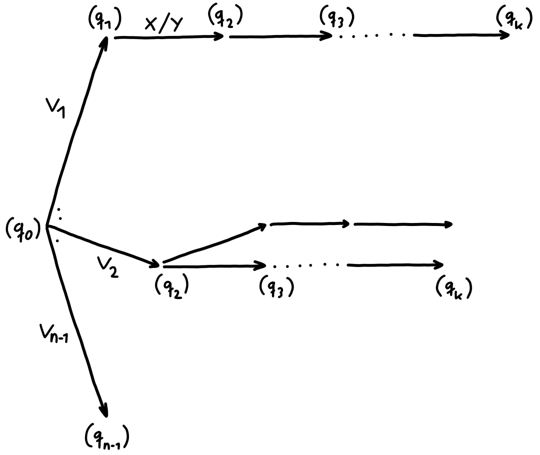
1. $V.\Sigma \subseteq TS$
2. TS preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$
   - SPY-Method: realised via *harmonised state identifiers*



SPY-Condition / SPY-Method

TS is *m*-complete if

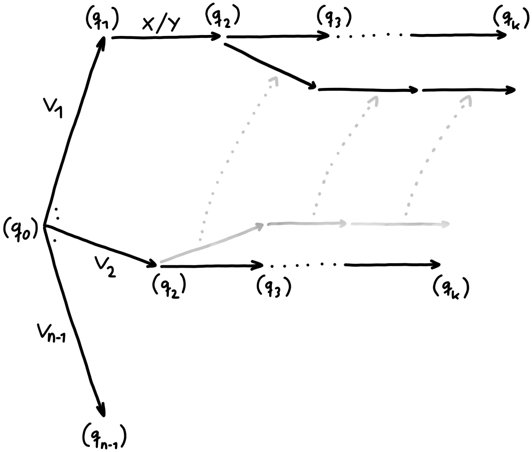
1. $V.\Sigma \subseteq TS$
2. *TS* preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
    - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.\mathit{pref}(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$
    - SPY-Method: realised via *harmonised state identifiers*
    - enables distribution of extensions over previously established convergences



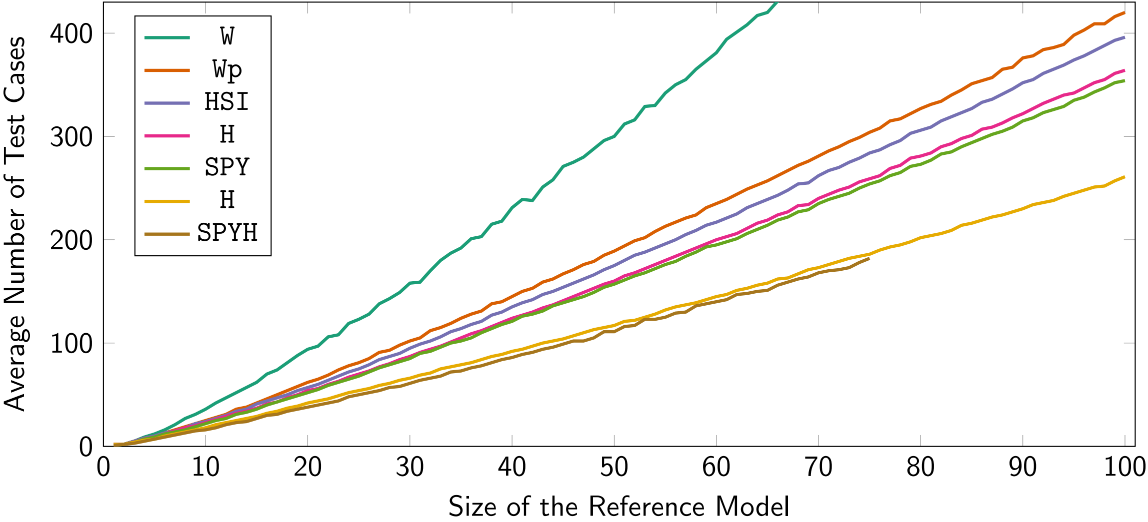
SPY-Condition / SPY-Method

TS is *m*-complete if

1. $V.\Sigma \subseteq TS$
2. *TS* preserves convergence of $\upsilon_q.(x/y)$ and $\upsilon_{q'}$ for all $(q, x, y, q') \in h_1$
   - by establishing divergences in $\{\upsilon_q.(x/y), \upsilon_{q'}\}.pref(\omega)$ for all $\omega \in \Sigma^{\leq m-n}$
   - SPY-Method: realised via *harmonised state identifiers*
   - enables distribution of extensions over previously established convergences

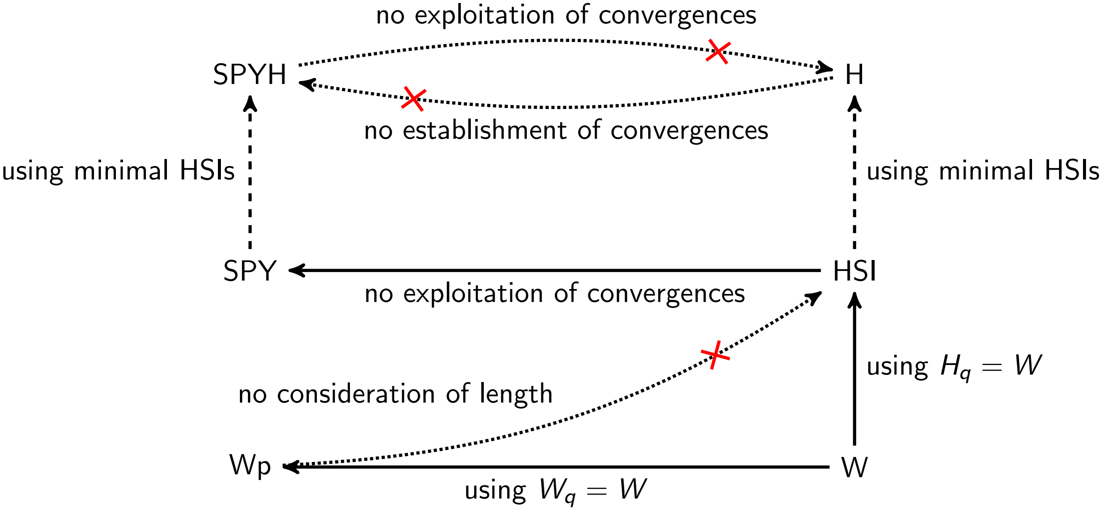


Historical Development of Test Strategies



Unified Implementation

Are these Strategies Merely Special Cases of Each Other?



Frameworks

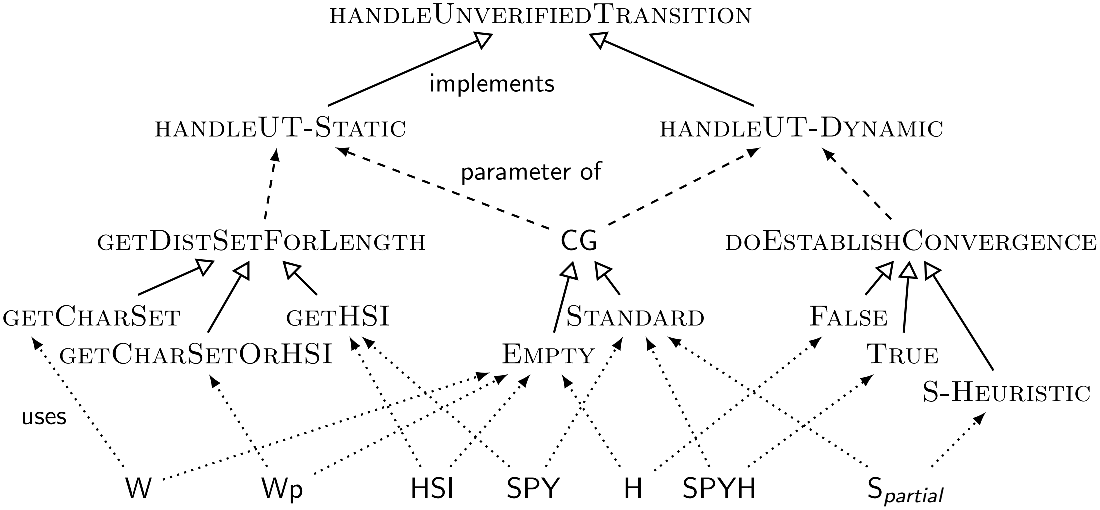
- goal: unify representation and avoid duplication
- implementation using a generic framework
  - "framework" here denotes a higher order function
  - behaviour shared by all strategies is implemented directly in the framework
  - differing behaviours are supplied via procedural parameters, e.g.
    - selection of distinguishing traces
    - exploitation of convergences
  - tool-independent

H-Framework

Input : minimal OFSM $M_1 = (Q, q_0, \Sigma_I, \Sigma_I, h_1)$ with $|Q| = n$
Input : integer m
Input : functions $\textsc{getStateCover}$, $\textsc{handleStateCover}$, $\textsc{sortTransitions}$, $\textsc{handleUnverifiedTransition}$, $\textsc{handleUndefinedIOPair}$
Output: test suite $TS \subseteq \Sigma^*$

$V \leftarrow \textsc{getStateCover}(M_1)$
$(TS, G) \leftarrow \textsc{handleStateCover}(M_1,\ V)$
$U \leftarrow \left\{(q, x, y, q') \in h_1 \;\middle|\; \upsilon_q.(x/y) \neq \upsilon_{q'}\right\}$ `// unverified transitions`
$U \leftarrow \textsc{sortTransitions}(U, V)$
foreach $t \in U$ **do**
    $(TS, G) \leftarrow \textsc{handleUnverifiedTransition}(M_1, V, t, m, TS, G)$
foreach $q \in Q, x \in \Sigma_I, y \in \Sigma_O$ *such that* $x/y \notin \mathcal{L}_{M_1}(q)$ **do**
    $TS \leftarrow \textsc{handleUndefinedIOPair}(M_1, V, q, x, y, TS, G)$
return TS

H-Framework – Nesting of Higher Order Functions



Frameworks

Framework	Strategy						Completeness Condition
	W	Wp	HSI	H	SPY	SPYH	
H	✓	✓	✓	✓	✓	✓	H-Condition
SPY	✓	✓	✓	(✓)	✓	✓	SPY-Condition
Pair	✓	✓	✓	✓			H-Condition, simplified

Mechanised Completeness Proofs using Isabelle/HOL

Proof Concept – Interface Lemmata

$$\forall f, g, h, i, j.\ \phi_1(f) \wedge \phi_2(g) \wedge \phi_3(h) \wedge \phi_4(i) \wedge \phi_5(j) \longrightarrow \text{H-}\textsc{Framework}(M_1, m, f, g, h, i, j)$$
$$\text{is } m\text{-complete}$$

- proof via satisfaction of the H-Condition
- maintainability
  - definition of function f_1 needs to be unfolded only in the proof of $\phi_1(f_1)$, limiting the impact in changes to f_1
- extensibility
  - arbitrary replacement of f_1 by f_2 if $\phi_1(f_2)$ holds
  - completeness of arbitrary combinations of arguments follows "for free"
    - SPY-W, SPY-Wp, partial S-Method

Structured Proofs in Isabelle/HOL

```
lemma acyclic_path_length_limit :
  assumes "path M q p"  and  "distinct (visited_states q p)"
shows "length p < size M"
proof (rule ccontr)
  assume "¬ length p < size M"
  then have "length p ≥ card (states M)" using size_def by auto
  then have "length (visited_states q p) > card (states M)" by auto
  moreover have "set (visited_states q p) ⊆ states M"
    by (metis assms path_prefix path_target_is_state visited_states_prefix)
  ultimately have "¬ distinct (visited_states q p)"
    by (metis distinct_card List.finite_set card_mono fsm_states_finite)
  then show "False" using assms(2) by blast
qed
```

Proof Mechanisation Effort

- basic FSM library
  - data types
  - transformations (observable, minimal, initially connected, *prime*)
  - computation of minimal length distinguishing traces
  - generalisation of convergence
- frameworks
  - H, SPY, Pair
  - H- and SPY-Condition
- concrete implementations of strategies (W, Wp, HSI, H, SPY, SPYH)
  - functions for procedural parameters

Proof Mechanisation Effort

- basic FSM library
  - data types
  - transformations (observable, minimal, initially connected, *prime*)
  - computation of minimal length distinguishing traces
  - generalisation of convergence
- frameworks
  - H, SPY, Pair
  - H- and SPY-Condition
- concrete implementations of strategies (W, Wp, HSI, H, SPY, SPYH)
  - functions for procedural parameters

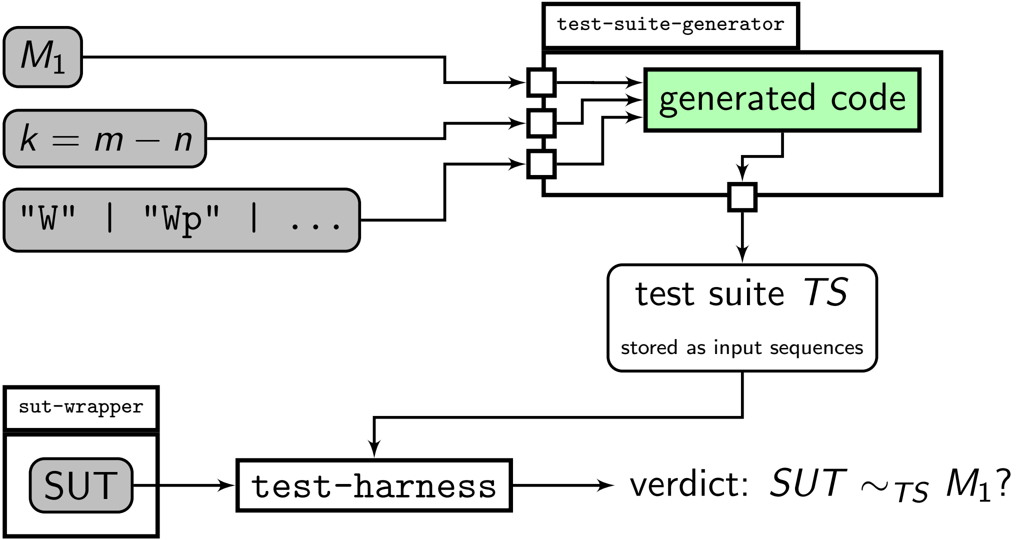
36.6 kLoC
329 definitions
856 lemmata
$\approx$ 50% fully automated top-level proofs

Generating Provably Correct Implementations

Code Generation via Isabelle/HOL

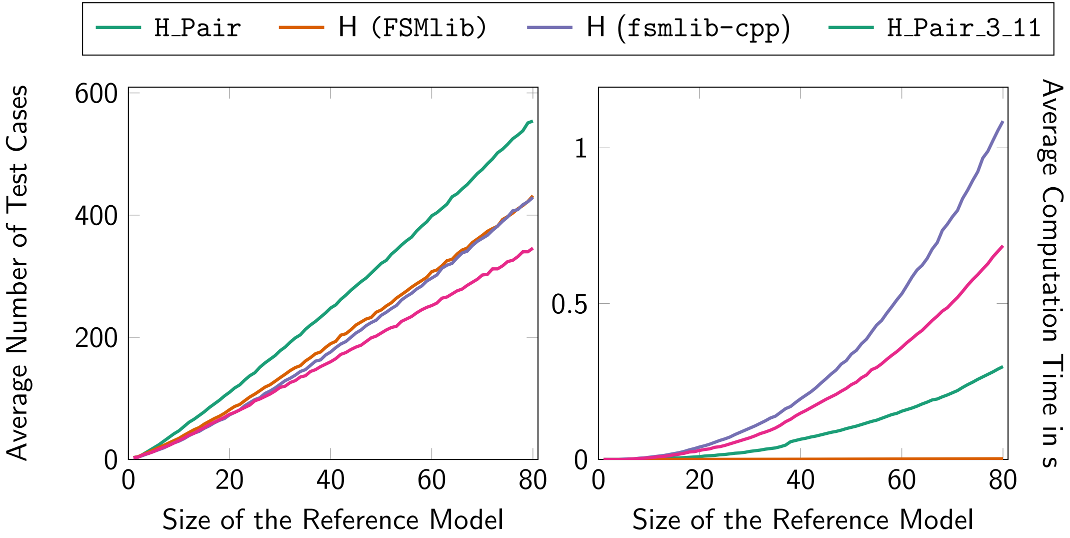
- translation to Haskell / SML / Scala / OCaml
- many definitions immediately translatable
- *code equations* specify alternative implementations
- refinements
  - data structures – extended FSM data type, *Containers*
  - algorithms
  - much potential for further improvement

Tool Set

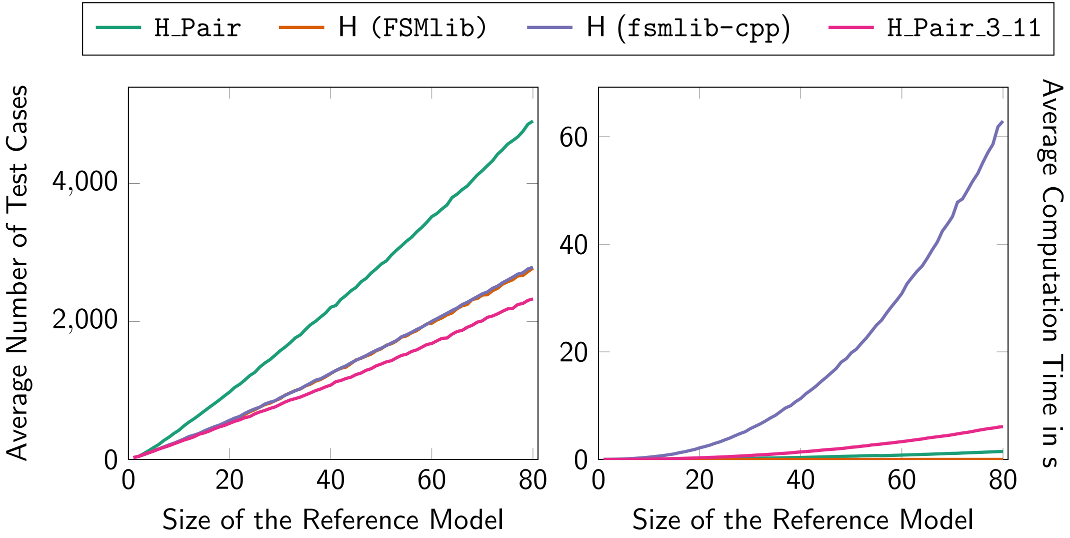


M_1
$k = m - n$
"W" | "Wp" | ...
test-suite-generator
generated code
test suite TS
stored as input sequences
sut-wrapper
SUT
test-harness
verdict: $SUT \sim_{TS} M_1$?

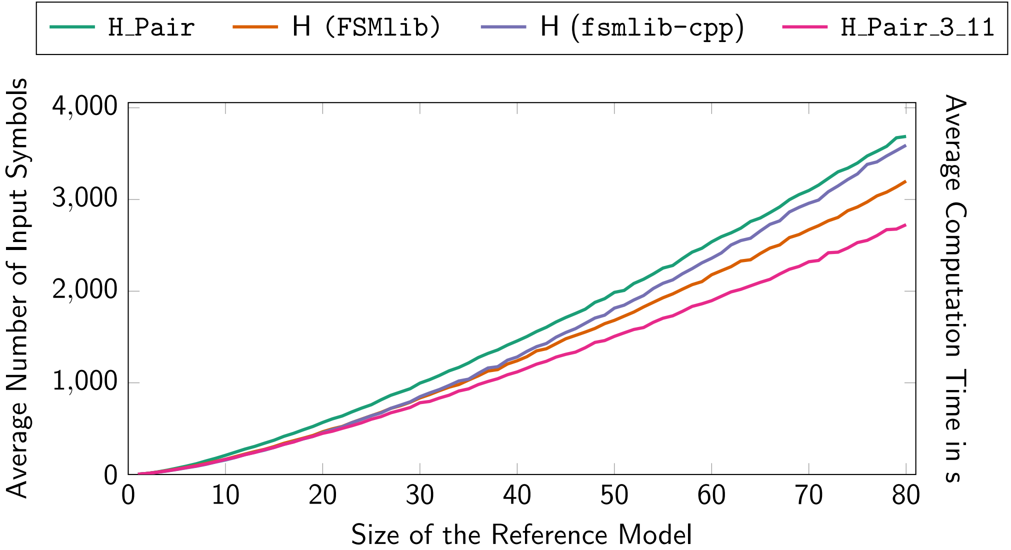
Evaluation – DFSMs with 3 Inputs and Outputs each, using $m - n = 0$



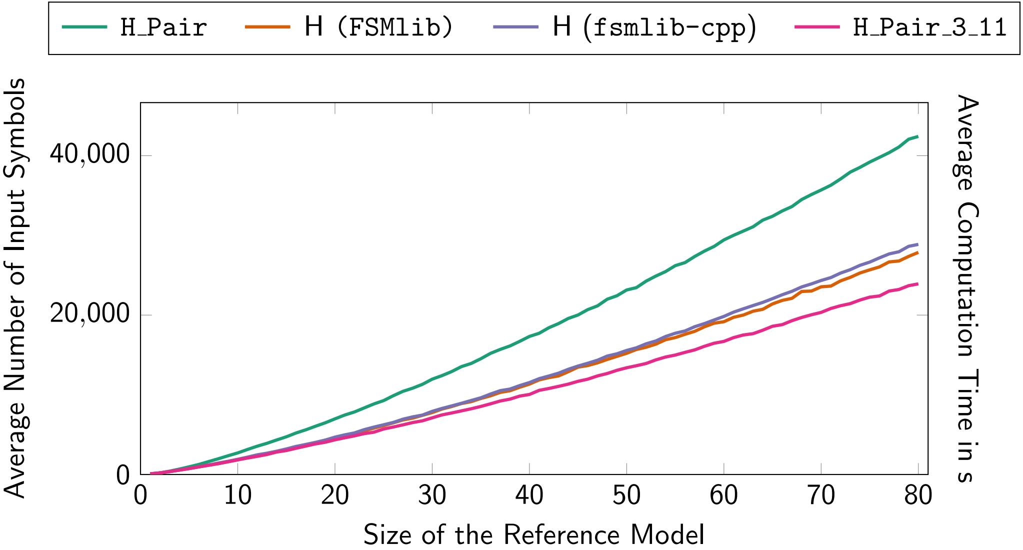
Evaluation – DFSMs with 3 Inputs and Outputs each, using $m - n = 2$



Evaluation – Average Test Suite Length $\sum_{\alpha \in TS} |\alpha|$ for $m - n = 0$



Evaluation – Average Test Suite Length $\sum_{\alpha \in TS} |\alpha|$ for $m - n = 2$



Conclusion and Future Work

Conclusion

- main contributions
  1. unified implementation of complete test strategies using frameworks
  2. mechanised correctness and completeness proofs
  3. provably correct implementations embedded in a practical tool set
- further contributions
  - generalisation of SPY and SPYH to partial, nondeterministic FSMs
  - provably correct library of basic FSM operations
  - analogous results for a test strategy for the *reduction* conformance relation

Future Work

- extend the approach to
  - further conformance relations
    - quasi-reduction/equivalence, *strong reduction*, ...
  - further modelling formalisms
    - EFSMs, SFSMs, timed Automata, LTSs, ...
  - further test strategies
    - S-Method, Safety-H-Method, Property Oriented Testing via FSMs, ...
- completeness checking instead of test suite generation

Bibliography I

- Achim D. Brucker and Burkhart Wolff. "On theorem prover-based testing". In: *Formal Aspects of Computing* 25.5 (2013), pp. 683–721. DOI: 10.1007/s00165-012-0222-y. URL: https://doi.org/10.1007/s00165-012-0222-y.
- Tsun S. Chow. "Testing Software Design Modeled by Finite-State Machines". In: *IEEE Transactions on Software Engineering* 4.3 (1978), pp. 178–187. DOI: 10.1109/TSE.1978.231496. URL: https://doi.org/10.1109/TSE.1978.231496.
- Rita Dorofeeva, Khaled El-Fakih, and Nina Yevtushenko. "An Improved Conformance Testing Method". In: *Formal Techniques for Networked and Distributed Systems - FORTE 2005, 25th IFIP WG 6.1 International Conference, Taipei, Taiwan, October 2-5, 2005, Proceedings*. Ed. by Farn Wang. Vol. 3731. Lecture Notes in Computer Science. Springer, 2005, pp. 204–218. DOI: 10.1007/11562436_16. URL: https://doi.org/10.1007/11562436%5C_16.

Bibliography II

- Gang Luo, Gregor von Bochmann, and Alexandre Petrenko. "Test Selection Based on Communicating Nondeterministic Finite-State Machines Using a Generalized WP-Method". In: *IEEE Trans. Software Eng.* 20.2 (1994), pp. 149–162. DOI: 10.1109/32.265636. URL: https://doi.org/10.1109/32.265636.
- Gang Luo, Alexandre Petrenko, and Gregor von Bochmann. "Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines". In: *Protocol Test Systems: 7th workshop 7th IFIP WG 6.1 international workshop on protocol text systems.* Ed. by Tadanori Mizuno, Teruo Higashino, and Norio Shiratori. Boston, MA: Springer US, 1995, pp. 95–110. ISBN: 978-0-387-34883-4. DOI: 10.1007/978-0-387-34883-4_6. URL: https://doi.org/10.1007/978-0-387-34883-4_6.
- Adenilso da Silva Simão, Alexandre Petrenko, and Nina Yevtushenko. "On reducing test length for FSMs with extra states". In: *Software Testing, Verification and Reliability* 22.6 (2012), pp. 435–454. DOI: 10.1002/stvr.452. URL: https://doi.org/10.1002/stvr.452.

Bibliography III

Michal Soucha. "Testing and active learning of resettable finite-state machines". PhD thesis. University of Sheffield, 2019.

Michal Soucha and Kirill Bogdanov. "SPYH-Method: An Improvement in Testing of Finite-State Machines". In: *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops, ICST Workshops, Västerås, Sweden, April 9-13, 2018*. IEEE Computer Society, 2018, pp. 194–203. DOI: 10.1109/ICSTW.2018.00050. URL: http://doi.ieeecomputersociety.org/10.1109/ICSTW.2018.00050.

M. P. Vasilevskii. "Failure diagnosis of automata". In: *Kibernetika (Transl.)* 4 (1973), pp. 98–108.