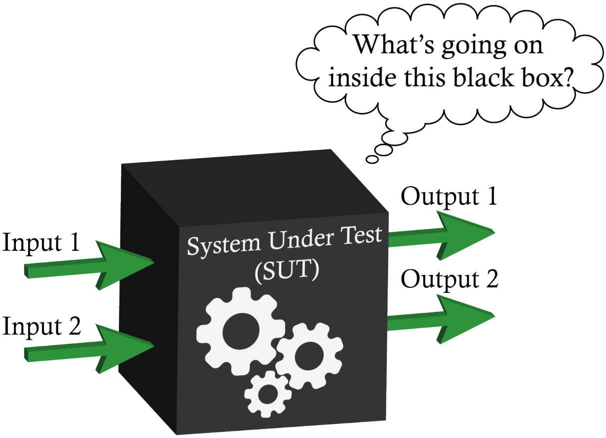
New Fault Domains for Conformance Testing of Finite State Machines

Frits Vaandrager   Ivo Melse

Radboud University Nijmegen

August 29, 2025

Black-box Conformance Testing



Question: Does SUT conform to its Spec?

Conformance Testing of FSMs: A Classic Problem

- Idea of (black-box) conformance testing can be traced back to Moore (1956) and Hennie (1964)
- Seminal results by Vasilevski (1973) and Chow (1978)
- Influential survey paper by Lee & Yannakakis (1994)
- Continued progress by model-based testing community

Conformance Testing of FSMs: A Classic Problem

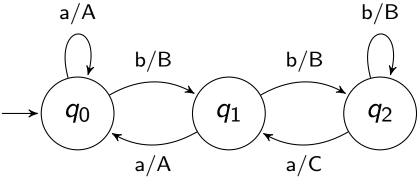
- Idea of (black-box) conformance testing can be traced back to Moore (1956) and Hennie (1964)
- Seminal results by Vasilevski (1973) and Chow (1978)
- Influential survey paper by Lee & Yannakakis (1994)
- Continued progress by model-based testing community
- Recent work on model learning poses major new challenges but also brings major new opportunities

Mealy Machines

The diagram below shows a simple Mealy machine with:

- finite sets of inputs $I = \{a, b\}$ and outputs $O = \{A, B, C\}$
- finite set of states $Q = \{q_0, q_1, q_2\}$ and initial state q_0
- transition function $\delta : Q \times I \to Q$
- output function $\lambda : Q \times I \to O$

We assume all states are reachable from the initial state.

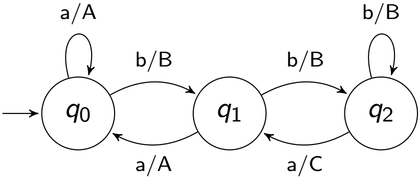


Mealy Machines

The diagram below shows a simple Mealy machine with:

- finite sets of inputs $I = \{a, b\}$ and outputs $O = \{A, B, C\}$
- finite set of states $Q = \{q_0, q_1, q_2\}$ and initial state q_0
- transition function $\delta : Q \times I \to Q$
- output function $\lambda : Q \times I \to O$

We assume all states are reachable from the initial state.



Mealy machines $\mathcal{M}$ and $\mathcal{N}$ are equivalent, $\mathcal{M} \approx \mathcal{N}$, iff for every sequence of inputs they produce the same sequence of outputs.

Conformance Testing

Definition (Conformance Testing)

Let $\mathcal{S}$ be a Mealy machine (the specification).

- A sequence $\sigma \in I^*$ is called a test for $\mathcal{S}$.
- Mealy machine $\mathcal{M}$ passes test σ for $\mathcal{S}$ iff $\mathcal{M}$ and $\mathcal{S}$ produce the same outputs in response to input sequence σ.
- A test suite T for $\mathcal{S}$ is a finite set of tests for $\mathcal{S}$.
- $\mathcal{M}$ passes T iff it passes all tests in T.

Fault Domains

A fault domain reflects assumptions about faults that may occur in an implementation and that need to be detected during testing:

Fault Domains

A fault domain reflects assumptions about faults that may occur in an implementation and that need to be detected during testing:

Definition (Fault domains and $\mathcal{U}$-completeness)

Let $\mathcal{S}$ be a Mealy machine. A fault domain is a set $\mathcal{U}$ of Mealy machines. A test suite T for $\mathcal{S}$ is $\mathcal{U}$-complete if, for each $\mathcal{M} \in \mathcal{U}$, $\mathcal{M}$ passes T implies $\mathcal{M} \approx \mathcal{S}$.

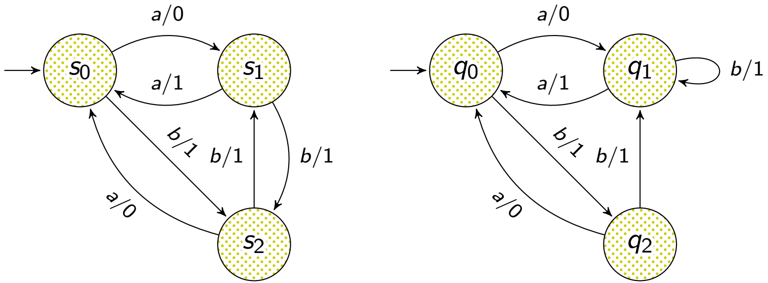
The Most Popular Fault Domain Ever?

Based on work of Moore, Hennie, and Chow, hundreds of papers about conformance testing use the following fault domain:

Definition ($\mathcal{U}_m$)

For $m > 0$, $\mathcal{U}_m$ is the set of Mealy machines with at most m states.

Example



Test suite $T = \{aaaa, abaa, baaa, bbaa\}$ for specification $\mathcal{S}$ on the left is $\mathcal{U}_3$-complete. SUT $\mathcal{M}$ on the right does not pass test $abaa$.

The Problem with $\mathcal{U}_m$

1. The size of $\mathcal{U}_m$-complete test suites grows exponentially in $m - n$, where n is the number of states of $\mathcal{S}$
2. Thus we can only run $\mathcal{U}_m$-complete test suites for small values of $m - n$ (say 2 or 3)
3. But the assumption that faults introduce at most a few extra states is not realistic

A Better Fault Domain

Definition ($\mathcal{U}_k^A$)

Let k be a natural number and let $A \subseteq I^*$. Then $\mathcal{U}_k^A$ is the set of all Mealy machines $\mathcal{M}$ such that every state of $\mathcal{M}$ can be reached by an input sequence $\sigma\rho$, for some $\sigma \in A$ and $\rho \in I^{\leq k}$.

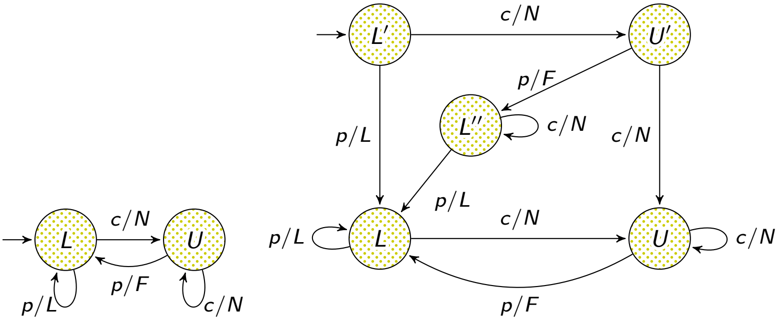
A Better Fault Domain

Definition ($\mathcal{U}_k^A$)

Let k be a natural number and let $A \subseteq I^*$. Then $\mathcal{U}_k^A$ is the set of all Mealy machines $\mathcal{M}$ such that every state of $\mathcal{M}$ can be reached by an input sequence $\sigma\rho$, for some $\sigma \in A$ and $\rho \in I^{\leq k}$.

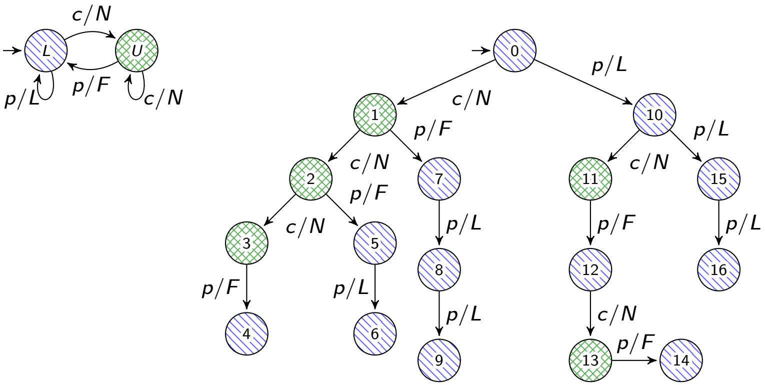
Intuition: set A describes some required behaviors, e.g. the happy flow of a protocol. An implementor will aim to support these behaviors, but may make mistakes along the way.

Turnstile Example



Mealy machine on the right is contained in fault domain $\mathcal{U}_1^A$, for $A = \{\epsilon, c\}$, since all states can be reached via at most one transition from states L' and U' that are reachable via A.

SPYH method is not $\mathcal{U}_1^A$-complete



Test suite $T = \{cccp, ccpp, cppp, pcpcp, ppp\}$, represented here as a tree, was generated using SPYH-method and is $\mathcal{U}_3$-complete for machine on the left. Test suite is not $\mathcal{U}_1^{\{\epsilon,c\}}$-complete.

A Better Fault Domain (cnt)

The number of states of FSMs in $\mathcal{U}_k^A$ grows exponentially in k:

Lemma

Let $A \subset I^*$ be prefix closed with $|A| = n$, let $|I| = l$ and $k > 0$. Then fault domain $\mathcal{U}_k^A$ contains Mealy machines with up to $n + (\sum_{j=0}^{k-1} l^j)(nl - n + 1)$ states.

Relation between $\mathcal{U}_m$ and $\mathcal{U}_k^A$

Let A be a minimal state cover for a minimal specification $\mathcal{S}$. $\mathcal{U}_m$-complete test suites typically check whether distinct sequences from A lead to distinct states in the SUT:

Definition ($\mathcal{U}^A$)

Let $A \subseteq I^*$. Then $\mathcal{U}^A$ is the set of all Mealy machines $\mathcal{M}$ such that there are $\sigma, \rho \in A$ with $\sigma \neq \rho$ and $\delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \sigma) \approx \delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \rho)$.

Relation between $\mathcal{U}_m$ and $\mathcal{U}_k^A$

Let A be a minimal state cover for a minimal specification $\mathcal{S}$.
$\mathcal{U}_m$-complete test suites typically check whether distinct sequences from A lead to distinct states in the SUT:

Definition ($\mathcal{U}^A$)

Let $A \subseteq I^*$. Then $\mathcal{U}^A$ is the set of all Mealy machines $\mathcal{M}$ such that there are $\sigma, \rho \in A$ with $\sigma \neq \rho$ and $\delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \sigma) \approx \delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \rho)$.

$\mathcal{U}_k^A \cup \mathcal{U}^A$-complete test suites are called k-A-complete.

Relation between $\mathcal{U}_m$ and $\mathcal{U}_k^A$

Let A be a minimal state cover for a minimal specification $\mathcal{S}$. $\mathcal{U}_m$-complete test suites typically check whether distinct sequences from A lead to distinct states in the SUT:

Definition ($\mathcal{U}^A$)

Let $A \subseteq I^*$. Then $\mathcal{U}^A$ is the set of all Mealy machines $\mathcal{M}$ such that there are $\sigma, \rho \in A$ with $\sigma \neq \rho$ and $\delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \sigma) \approx \delta^{\mathcal{M}}(q_0^{\mathcal{M}}, \rho)$.

$\mathcal{U}_k^A \cup \mathcal{U}^A$-complete test suites are called k-A-complete.

Theorem

Let $A \subset I^*$ be a finite set of input sequences with $\epsilon \in A$. Let k and m be natural numbers with $m = |A| + k$. Then $\mathcal{U}_m \subset \mathcal{U}_k^A \cup \mathcal{U}^A$.

A Better Fault Domain (cnt)

Let A be a set of traces describing the happy flow of TLS protocol. We computed the eccentricity of SUT models from De Ruiter et al:[1]

SUT model client	States	Eccentricity	SUT model server	States	Eccentricity
GnuTLS 3.3.12 full	9	0	GnuTLS 3.3.12 full	9	0
GnuTLS 3.3.8 full	15	1	GnuTLS 3.3.12 regular	7	0
GnuTLS 3.3.8 regular	11	1	GnuTLS 3.3.8 full	15	1
NSS 3.17.4 full	11	0	GnuTLS 3.3.8 regular	12	1
NSS 3.17.4 regular	7	0	NSS 3.17.4 regular	8	1
OpenSSL 1.0.1g regular	10	3	OpenSSL 1.0.1j regular	11	3
OpenSSL 1.0.1j regular	6	0	OpenSSL 1.0.1l regular	10	3
OpenSSL 1.0.1l regular	6	0	OpenSSL 1.0.2 regular	9	1
OpenSSL 1.0.2 full	8	0	RSA BSAFE C 4.0.4 regular	9	1
OpenSSL 1.0.2 regular	6	0	RSA BSAFE Java 6.1.1 regular	6	0
			miTLS 0.1.3 server regular	6	0

[1] Thanks to Paul Fiterau

A Better Fault Domain (cnt)

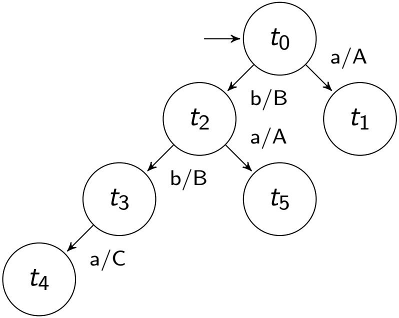
Let A be a set of traces describing the happy flow of TLS protocol. We computed the eccentricity of SUT models from De Ruiter et al:[1]

SUT model client	States	Eccentricity	SUT model server	States	Eccentricity
GnuTLS 3.3.12 full	9	0	GnuTLS 3.3.12 full	9	0
GnuTLS 3.3.8 full	15	1	GnuTLS 3.3.12 regular	7	0
GnuTLS 3.3.8 regular	11	1	GnuTLS 3.3.8 full	15	1
NSS 3.17.4 full	11	0	GnuTLS 3.3.8 regular	12	1
NSS 3.17.4 regular	7	0	NSS 3.17.4 regular	8	1
OpenSSL 1.0.1g regular	10	3	OpenSSL 1.0.1j regular	11	3
OpenSSL 1.0.1j regular	6	0	OpenSSL 1.0.1l regular	10	3
OpenSSL 1.0.1l regular	6	0	OpenSSL 1.0.2 regular	9	1
OpenSSL 1.0.2 full	8	0	RSA BSAFE C 4.0.4 regular	9	1
OpenSSL 1.0.2 regular	6	0	RSA BSAFE Java 6.1.1 regular	6	0
			miTLS 0.1.3 server regular	6	0

Indication that fault domains $\mathcal{U}_k^A$ may be practically relevant.

[1] Thanks to Paul Fiterau

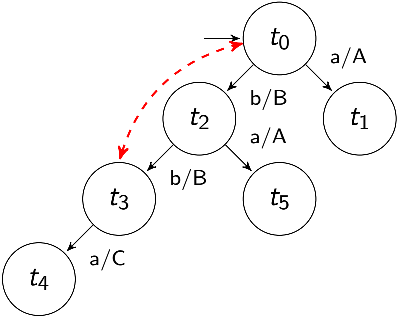
Apartness



Definition (Apartness)

Two states t, u in a testing tree are apart, $t \# u$, if they both enable an input sequence $\sigma \in I^*$ for which the outputs are different.

Apartness

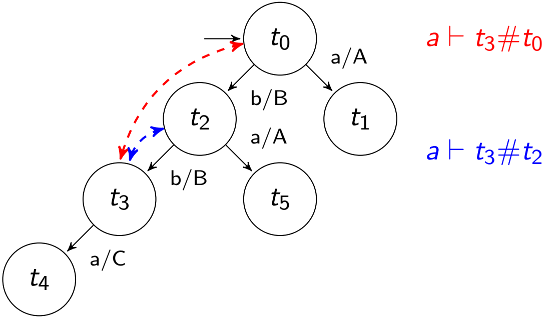


$a \vdash t_3 \# t_0$

Definition (Apartness)

Two states t, u in a testing tree are apart, $t \# u$, if they both enable an input sequence $\sigma \in I^*$ for which the outputs are different.

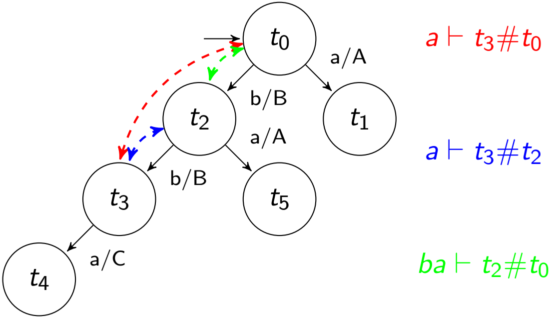
Apartness



Definition (Apartness)

Two states t, u in a testing tree are apart, $t \# u$, if they both enable an input sequence $\sigma \in I^*$ for which the outputs are different.

Apartness



Definition (Apartness)

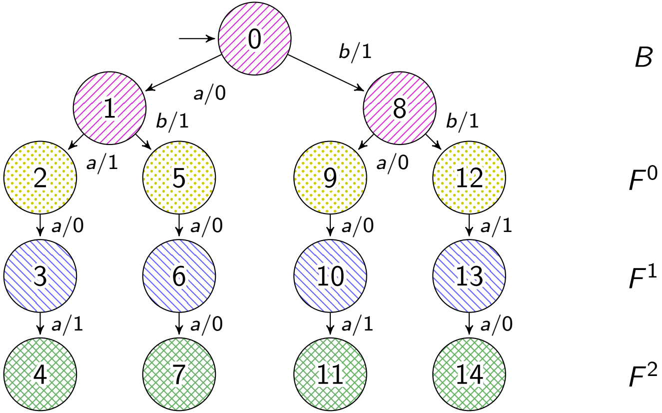
Two states t, u in a testing tree are apart, $t \# u$, if they both enable an input sequence $\sigma \in I^*$ for which the outputs are different.

Basis

Definition (Basis)

Let $\mathcal{T}$ be a testing tree. A nonempty subset of states $B \subseteq Q^{\mathcal{T}}$ is called a basis of $\mathcal{T}$ if B is ancestor-closed and all states in B are pairwise apart.
For each state q of $\mathcal{T}$, the candidate set $C(q)$ is the set of basis states that are not apart from q: $C(q) = \{q' \in B \mid \neg(q \# q')\}$.
A state q of $\mathcal{T}$ is identified if its candidate set is a singleton.

Example


0
b/1
a/0
1
8
b/1
b/1
2
5
9
12
a/1
a/0
a/0
a/0
a/0
a/1
3
6
10
13
a/1
a/0
a/1
a/0
4
7
11
14
B
F^0
F^1
F^2

Stratification

Definition (Stratification)

A basis B induces a stratification of $Q^{\mathcal{T}}$ as follows:

1. We write F^0 for the set of immediate successors of basis states that are not basis states themselves.
2. For $k > 0$, F^k is the set of immediate successors of F^{k-1}.

For $k \in \mathbb{N}$, we write $F^{<k} = \bigcup_{i<k} F^i$.

A Sufficient Condition for k-A-Completeness

Theorem

Let $\mathcal{S}$ be a Mealy machine, let T be a test suite for $\mathcal{S}$, let $\mathcal{T} = \mathrm{Tree}(\mathcal{S}, T)$, let B be a basis for $\mathcal{T}$ with $|B| = |Q^{\mathcal{S}}|$, let $A = \mathrm{access}(B)$, let $F^0, F^1, \ldots$ be the stratification induced by B, and let $k \geq 0$. Suppose all states in $B \cup F^{<k}$ are complete, all states in F^k are identified, and:

$$\forall q \in F^k \; \forall r \in F^{<k} : \qquad C(q) = C(r) \vee q \# r$$

Then T is k-A-complete.

Complexity

Theorem

Let $\mathcal{S}$ be a Mealy machine and let T be a test suite for $\mathcal{S}$. Let N be the number of states in the tree representation of T. Then there is a $O(N^2)$ algorithm that checks whether the testing tree for T satisfies our sufficient condition for k-A-completeness.

k-A-Completeness Existing Methods

Corollary

The Wp, HSI, W, UIOv, ADS and hybrid ADS methods generate k-A-complete test suites.

k-A-Completeness Existing Methods

Corollary

The Wp, HSI, W, UIOv, ADS and hybrid ADS methods generate k-A-complete test suites.

This explains why "W- and Wp-methods exhibit significantly greater test strength than conventional random testing, even for behaviors that are not contained in the fault domain" (Hübner et al, 2019)

k-A-Completeness Existing Methods

Corollary

The Wp, HSI, W, UIOv, ADS and hybrid ADS methods generate k-A-complete test suites.

This explains why "W- and Wp-methods exhibit significantly greater test strength than conventional random testing, even for behaviors that are not contained in the fault domain" (Hübner et al, 2019)

Theorem

The SPYH, SPY and H methods do not ensure k-A-completeness.

Conclusions

1. We proposed new fault domains, leading to the notion of *k-A-completeness*, which may be of practical interest since the number of extra states grows exponentially in k.

Conclusions

1. We proposed new fault domains, leading to the notion of *k-A-completeness*, which may be of practical interest since the number of extra states grows exponentially in k.
2. We presented a sufficient condition for k-A-completeness of test suites that can be checked efficiently.

Conclusions

1. We proposed new fault domains, leading to the notion of k-A-completeness, which may be of practical interest since the number of extra states grows exponentially in k.
2. We presented a sufficient condition for k-A-completeness of test suites that can be checked efficiently.
3. Our condition implies k-A-completeness of several existing test suite generation approaches, e.g. the Wp and HSI methods.

Conclusions

1. We proposed new fault domains, leading to the notion of *k-A-completeness*, which may be of practical interest since the number of extra states grows exponentially in k.
2. We presented a sufficient condition for k-A-completeness of test suites that can be checked efficiently.
3. Our condition implies k-A-completeness of several existing test suite generation approaches, e.g. the Wp and HSI methods.
4. Counterexamples show that H, SPY and SPYH methods do not guarantee k-A-completeness.

Future Work

1. Emperical study to find out whether faulty implementations are contained in fault domains $\mathcal{U}_k^A$, for small k

Future Work

1. Emperical study to find out whether faulty implementations are contained in fault domains $\mathcal{U}_k^A$, for small k
2. Extend results to partial and nondeterministic Mealy machines and LTSs, register automata, Mealy machines with timers,..

Future Work

1. Emperical study to find out whether faulty implementations are contained in fault domains $\mathcal{U}_k^A$, for small k
2. Extend results to partial and nondeterministic Mealy machines and LTSs, register automata, Mealy machines with timers,..
3. Develop efficient test suite generation algorithms based on our characterization

Future Work

1. Emperical study to find out whether faulty implementations are contained in fault domains $\mathcal{U}_k^A$, for small k
2. Extend results to partial and nondeterministic Mealy machines and LTSs, register automata, Mealy machines with timers,..
3. Develop efficient test suite generation algorithms based on our characterization
4. Bridge the gap between sensible fault domains and practical test generation methods