A Formal Introduction to Model-Based Testing
Part I: Exhaustive Testing Methods

Jan Peleska
jp@verified.de

Verified Systems International GmbH and University of Bremen

ICTAC 2008

Why will testing remain a crucial verification and validation activity ?

- Simple answer: because standards for safety-critical systems development will never allow certification without testing
- More elaborate answers:
  - Complex HW/SW systems cannot be captured in a completely formal way – therefore at least **HW/SW integration and system integration testing** will remain important for system verification
  - Software testing plays an increasingly important role for the verification of automatic code generators
  - 100% software correctness is not always the main issue, because
    - 100% software correctness does not imply system safety (recall Leveson: “*Safety is an emergent property*”)
    - Systems containing software bugs can still be safe

Model-based equivalence testing . . .

. . . is a variant of **exhaustive testing**:

- The goal of the test suite is to establish an **equivalence relation** between specification model and implementation
- Typical equivalence relations are
  - **Bi-similarity**
  - **Failures equivalence**
- From a practical point of view, proof of **refinement properties** by means of exhaustive testing is often more relevant than equivalence testing

Model-based equivalence testing versus model checking

- White-box equivalence testing identical to model (equivalence) checking
- Grey-box equivalence testing differs from model checking:
  - The implementation model is only partially known, e. g., the maximal number of states and the interface latency of the implementation
- Black-box equivalence testing is impossible, due to the **time-bomb problem**: The SUT may behave properly for an unknown number of execution loops and fail after some hidden state condition (e. g., a counter overflow) arises
- In principle, all tests could be assumed to be grey box, since hardware limitations always impose a finite state system. This limit, however, will be so large that no practical application of equivalence testing is feasible.

Chow's Theorem (1)

- Tsun S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering* SW-4, No. 3, pp. 178-187(1978).
- Equivalence testing for deterministic Mealy automata
- One of the first contributions showing that equivalence proof by grey-box testing is possible with a **finite number of test cases**
- The test case construction method according to Chow is also called **W-Method**
- For a more detailed error classification extending the examples below see Chow's paper and
  Robert. V. Binder: *Testing Object-Oriented Systems.* Addison Wesley (1999).

Chow's Theorem (2): Pre-requisites

- A and B are Mealy automata over the same alphabet $\Sigma = I \cup O$
- I contains input symbols, O output symbols
- Transition functions
  $\delta_A : Q(A) \times I \rightarrow Q(A) \times O$ and $\delta_B : Q(B) \times I \rightarrow Q(B) \times O$
  are total functions
- For $\delta(q_1, x) = (q_2, y)$ we also write $q_1 \xrightarrow{x/y} q_2$.
- If input sequence $p = \langle x_1, \ldots, x_k \rangle$ leads from state q_1 to final state q_2, we write $q_1 \stackrel{p}{\Longrightarrow} q_2$.
- We require A and B to be minimal (this simplifies the proof, but is not essential)
- A is used as the **model**, B as the **implementation**.

Chow's Theorem (3): Pre-requisites

- The set of states $Q(A)$ has cardinality n, $card(Q(B)) = m$
- Initial states: q_A, q_B.
- **Test cases** are input traces $p \in I^*$.
- The specification automaton A serves as **test oracle**: The generated input trace, when exercised on B, leads to an output trace which can be observed, and the resulting I/O-trace $u \in \Sigma^*$ can be automatically checked against A, whether it is a word of $\mathcal{L}(A)$
- $P \subseteq I^*$ is called **transition cover** of A, if:

$$\forall q_1 \xrightarrow{x/y} q_2 \in \delta_A : \exists p \in P : q_A \overset{p}{\Longrightarrow} q_1 \wedge p \frown \langle x \rangle \in P$$

Chow's Theorem (4): Pre-requisites

- $W \subseteq I^*$ is called **characterisation set** of A if for all $q_1, q_2 \in Q(A)$, there exists a $w \in W$ **distinguishing** q_1 and q_2, i. e.: w applied to q_1 results in an output trace which differs from the one resulting from application of w to q_2.
- Define $X^n = \{p \in I^* \mid \#p = n\}$ for $n \geq 0$.
- Define $U_1 \cdot U_2 = \{u_1 \frown u_2 \mid u_i \in U_i, i = 1, 2\}$ for $U_1, U_2 \subseteq I^*$.
- Define $\mathcal{W}(A)$, the set of **W-test cases** of A by

$$\mathcal{W}(A) = P \cdot (\bigcup_{i=0}^{m-n} (X^i \cdot W))$$

Chow's Theorem (5)

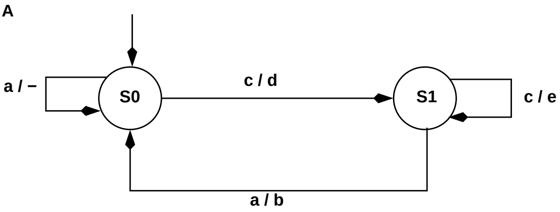
Chows Theorem If B passes all W-test cases from $\mathcal{W}(A)$ then A and B are bi-similar (written $A \approx B$).

Remarks.

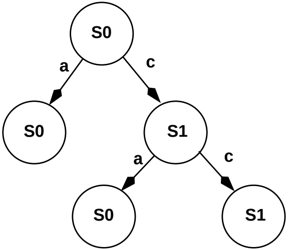
- "Passing a test case from $\mathcal{W}(A)$" means to generate the same outputs as A for every input sequence $w \in \mathcal{W}(A)$
- Bi-similarity for finite deterministic Mealy automata just means language equivalence.
- Bi-similarity of minimal Mealy automata is equivalent to the existence of an **isomorphism** $f : A \longrightarrow B$: f is bijective and satisfies $f(q_A) = f(q_B)$ and

$$\forall q_1, q_2 \in Q(A) : q_1 \xrightarrow{x/y} q_2 \implies f(q_1) \xrightarrow{x/y} f(q_2)$$

Chow's Theorem (5b) – Illustration



Transition cover P



Characterisation set W

W = { c }

Assume card(Q(B)) <= card(Q(A))+1

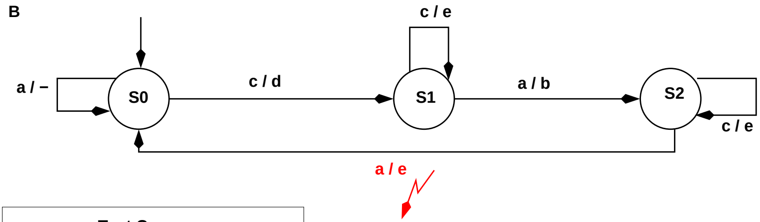
X^1 = { a, c }

P = { <>, a, c, ca, cc }

Test Cases:

$P\,X^0\,W$:	c ac cc cac ccc
$P\,X^1\,W$:	ac aac cac caac ccac
$P\,X^1\,W$:	cc acc ccc cacc cccc

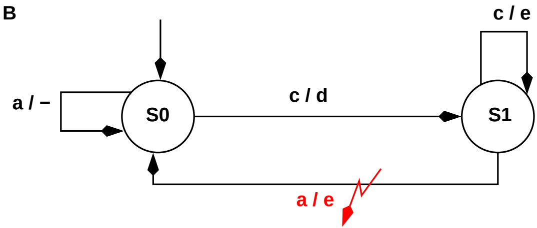
Chow's Theorem (5c) – Illustration: Time Bomb



	Test Cases:
$P X^0 \dot{W}$	c ac cc cac ccc
$P X^1 W$:	ac aac cac caac ccac
$P X^1 W$:	cc acc ccc cacc cccc

Failure is found by caac
(last c input not needed to uncover failure)

Chow's Theorem (5d) – Illustration: Output failure



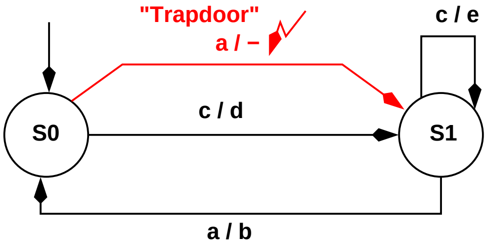
	Test Cases:
$P X^0 W$:	c ac cc cac ccc
$P X^1 W$:	ac aac cac caac ccac
$P X^1 W$:	cc acc ccc cacc cccc

Failure is found by ca(c)

Only transition cover is required to uncover output failures

Chow's Theorem (5e) – Illustration: Transition failure

B



	Test Cases:
$P X^0 W$:	c ac cc cac ccc
$P X^1 W$:	ac aac cac caac ccac
$P X^1 W$:	cc acc ccc cacc cccc

Failure is found by **ac**

Chow's Theorem (6): Preparations for the proof

Definition 1: Let $V \subseteq I^*$ a set of input traces

1. Two states $q_i \in Q(A), q_j \in Q(B)$ are **V-equivalent** ($q_i \sim_V q_j$), if each $p \in V$ produces the same outputs when exercised from q_i as when exercised from q_j.
2. Automata A and B are **V-equivalent** ($A \sim_V B$), if their initial states are V-equivalent, i. e., $q_A \sim_V q_B$

Obviously $\sim_V$ is an equivalence relation on $Q(A) \times Q(B)$

Chow's Theorem (7): Proof

Obviously,

$$A \approx B \implies (\forall V \subseteq I^* : A \sim_V B)$$

holds for all bi-similar automata ($A \approx B$). Therefore we can re-write Chow's theorem as

Chow's Theorem – Variant 2: $A \sim_{\mathcal{W}(A)} B \implies A \approx B$

The proof of variant 2 results from the lemmas below. We assume that A has n states and B $m \geq n$ states and that both are minimal. The characterisation set of A is denoted by W.

Chow's Theorem (8): Proof

Lemma 1: Suppose characterisation set W of A partitions $Q(B)$ into at least n equivalence classes. Then $Z = \bigcup_{i=0}^{m-n}(X^i \cdot W)$ partitions $Q(B)$ into m classes. This means that every two states $Q(B)$ can be distinguished by $\mathcal{W}(A)$

Proof.: Define $Z(\ell) = \bigcup_{i=0}^{\ell}(X^i \cdot W)$. Obviously $Z(m-n) = Z$. Perform induction proof for $\ell = 0, 1, \ldots m-n$:

$$Z(\ell) \text{ partitions } Q(B) \text{ into } \ell + n \text{ classes} \qquad (*)$$

Choosing $\ell = m - n$ implies the lemma.

Chow's Theorem (9): Proof of Lemma 1

Proof of (∗) – induction start: For $\ell = 0$ (∗) coincides with the assumptions of the lemma.

Assumption: For given $\ell \in \{0, 1, \ldots m - n - 1\}$ $Z(\ell)$ partitions $Q(B)$ into at least $\ell + n$ classes

Induction step: We show that $Z(\ell + 1)$ partitions $Q(B)$ into at least $\ell + n + 1$ classes

If $Z(\ell)$ already partitions $Q(B)$ into $\ell + n + 1$ or more classes then we have nothing to prove. Otherwise there exists $k > \ell$ such that (observe that $Z(k) = Z(k - 1) \cup X^k \cdot W$)

$$\exists r_1, r_2 \in Q(B) : r_1 \sim_{Z(k-1)} r_2 \wedge r_1 \not\sim_{(X^k \cdot W)} r_2$$

Chow's Theorem (10): Proof of Lemma 1

If $k = \ell + 1$ there is nothing more to show since $(*)$ holds for $Z(k) = Z(\ell + 1)$.
Otherwise, if $k \geq \ell + 2$, let $p = \langle x_1, \ldots, x_k \rangle \frown w, w \in W$ the input sequence distinguishing r_1 and r_2.
Choose r_1', r_2' such that $r_1 \stackrel{\langle x_1, \ldots x_{k-\ell-1} \rangle}{\Longrightarrow} r_1'$, $r_2 \stackrel{\langle x_1, \ldots x_{k-\ell-1} \rangle}{\Longrightarrow} r_2'$. Then r_1', r_2' can be distinguished by $Z(\ell + 1)$. $\square$

Chow's Theorem (11): Lemma 2

Lemma 2: Let $Z = \bigcup_{i=0}^{m-n}(X^i \cdot W)$ as introduced in Lemma 1. Then $A \approx B$ if and only if the following conditions are fulfilled

1. The initial states of A and B are Z-equivalent: $q_A \sim_Z q_B$.
2. For all $a \in Q(A)$ exists $b \in Q(B)$ such that $a \sim_Z b$.
3. For all $a_i \xrightarrow{x/y} a_j$ in A exists $b_i, b_j \in Q(B)$, such that $a_i \sim_Z b_i$, $a_j \sim_Z b_j$ and $b_i \xrightarrow{x/y} b_j$.

Chow's Theorem (12): Proof of Lemma 2

Proof Step (a). If $A \approx B$, then (1,2,3) are directly implied by the existence of an isomorphism $f : Q(A) \longrightarrow Q(B)$.

Proof Step (b). Suppose (1,2,3) hold. We have to establish the existence of an isomorphism $f : Q(A) \longrightarrow Q(B)$. To this end we will show that function f specified by

$$f(q_A) = q_B$$
$$(q_A \stackrel{\langle x_1,\dots,x_\ell\rangle}{\Longrightarrow} a \wedge q_B \stackrel{\langle x_1,\dots,x_\ell\rangle}{\Longrightarrow} b) \Longrightarrow f(a) = b$$

is well-defined, one-one and surjective. Then (3) additionally implies that $\forall a \in Q(A) : a \sim_Z f(a)$ holds, too.

Chow's Theorem (13): Proof of Lemma 2

Well-definedness of f. It has to be shown that *different* input traces $q_A \stackrel{\langle x_1,\dots,x_\ell\rangle}{\Longrightarrow} a$, $q_A \stackrel{\langle x'_1,\dots,x'_k\rangle}{\Longrightarrow} a$, leading to the same target state a in A will also lead to the same target state in B.

Therefore suppose $q_B \stackrel{\langle x_1,\dots,x_\ell\rangle}{\Longrightarrow} b$ and $q_B \stackrel{\langle x'_1,\dots,x'_k\rangle}{\Longrightarrow} b'$ in B. It has to be shown that $b = b'$.

Because of (3) we can conclude

$$a \sim_Z b \wedge a \sim_Z b' \qquad (**)$$

We will now show that Z distinguishes every pair of states in B, so that (**) implies $b = b'$. This establishes well-definedness of f.

Chow's Theorem (13): Proof of Lemma 2

Z distinguishes every pair of B-states. The characterisation set W of A partitions $Q(A)$ into $n = card(Q(A))$ classes (since A is minimal).
Now (2) and (3) imply that W also partitions $Q(B)$ into at least n classes: Suppose a_1 and a_2 are distinguished by $w \in W$. Suppose $q_A \stackrel{\langle x_1,\dots,x_\ell\rangle}{\Longrightarrow} a_1$ and $q_A \stackrel{\langle x'_1,\dots,x'_k\rangle}{\Longrightarrow} a_2$. These two input traces will lead us according to (3) to states $b_1, b_2 \in Q(B)$ such that $a_i \sim_Z b_i, i = 1, 2$.

Chow's Theorem (14): Proof of Lemma 2

Because of (3) and $W \subseteq Z$, sequence $b_1 \stackrel{w}{\Longrightarrow}$ has to generate the same outputs as $a_1 \stackrel{w}{\Longrightarrow}$ and $b_2 \stackrel{w}{\Longrightarrow}$ the same outputs as $a_2 \stackrel{w}{\Longrightarrow}$.
Since w produces different outputs when applied to a_1 and a_2, respectively, the same has to hold for $b_1 \stackrel{w}{\Longrightarrow}$ and $b_2 \stackrel{w}{\Longrightarrow}$. Therefor w also distinguishes b_1 and b_2, and therefore $b_1 \neq b_2$.
Since $W \subseteq Z$ and since W partitions $Q(B)$ into at least n classes, we can apply Lemma 1 to conclude that Z distinguishes *all* states of B.
Let $b \in Q(B)$, then $b \sim_Z b'$ implies $b = b'$ which shows well-definedness of f.

Chow's Theorem (15): Proof of Lemma 2

f **is one-one.** Let $a_i \in Q(A), i = 1, 2, a_1 \neq a_2$ and $b_i = f(a_i) \in Q(B)$. We have to show that $b_1 \neq b_2$.
Since $a_1 \not\sim_W a_2$ and $W \subseteq Z$ we conclude $a_1 \not\sim_Z a_2$. (3) implies $a_i \sim_Z f(a_i) = b_i, i = 1, 2$ and therefore $b_1 \not\sim_Z b_2$, and therefore also $b_1 \neq b_2$.

Chow's Theorem (16): Proof of Lemma 2

f is surjective. Given $b \in Q(B)$ and an input sequence $q_B \stackrel{\langle x_1,\dots,x_\ell \rangle}{\Longrightarrow} b$. Since A and B are deterministic, the target states $b \in Q(B), a \in Q(A)$ are uniquely determined by $q_B \stackrel{\langle x_1,\dots,x_\ell \rangle}{\Longrightarrow} b$ and $q_A \stackrel{\langle x_1,\dots,x_\ell \rangle}{\Longrightarrow} a$. Since we already know that that f is well-defined this implies $f(a) = b$. $\square$

Chow's Theorem (17): Lemma 3

Lemma 3: Let $\mathcal{W}(A) = P \cdot Z$, where P is the transition cover of A and $Z = \bigcup_{i=0}^{m-n}(X^i \cdot W)$. Then $A \sim_{\mathcal{W}(A)} B$ if and only if

1. The initial states of A and B are Z-equivalent: $q_A \sim_Z q_B$.
2. For all $a \in Q(A)$ exists $b \in Q(B)$ such that $a \sim_Z b$.
3. For all $a_i \xrightarrow{x/y} a_j$ in A exists $b_i, b_j \in Q(B)$, such that $a_i \sim_Z b_i$, $a_j \sim_Z b_j$ and $b_i \xrightarrow{x/y} b_j$.

Observation. Since (1,2,3) are identical with the only-if condition of Lemma 2, and therefore imply $A \approx B$, Lemma 3 directly implies Chow's theorem, variant 2, because with Lemma 3

$$A \sim_{P \cdot Z} B \Leftrightarrow A \approx B$$

holds.

Chow's Theorem (18): Proof of Lemma 3

Proof of Lemma 3 – (a). Suppose (1,2,3) hold. Then Lemma 2 implies $A \approx B$ and this trivially implies $A \sim_{\mathcal{W}(A)} B$.
Proof of Lemma 3 – (b). Suppose $A \sim_{P.Z} B$. Given $a \in Q(A)$ and input sequence $p \in P$ with $q_A \stackrel{p}{\Longrightarrow} a$. This sequence p exists because P is a transition cover. Since A and B are deterministic b is uniquely determined by $q_B \stackrel{\langle x_1,\ldots,x_\ell \rangle}{\Longrightarrow} b$. Since $q_A \sim_{P.Z} q_B$ and $p \in P$, $a \sim_Z b$ follows, and this shows (2) and (3) (observe that $\langle\ \rangle \in P$).

Chow's Theorem (19): Proof of Lemma 3

Let $a_1 \xrightarrow{x/y} a_2$ a transition in A. Let $p \in P$ with $q_A \stackrel{p}{\Longrightarrow} a_1$. Since P is a transition cover, p exists and also $p \frown \langle x \rangle \in P$. Define $b_1, b_2 \in Q(B)$ uniquely by $q_B \stackrel{p}{\Longrightarrow} b_1$ and $q_B \stackrel{p \frown \langle x \rangle}{\Longrightarrow} b_2$.
Now $A \sim_{P.Z} B$ implies $a_i \sim_Z b_i, i = 1, 2$. In addition, transition $b_1 \xrightarrow{x/y'} b_2$ has to satisfy $y' = y$, because otherwise a_1 and b_1 could be distinguished by input x, and this would be a contradiction to $a_1 \sim_Z b_1$. □

Chow's Theorem (20): BFS-Algorithm for Transition Cover Construction

Overview over the algorithm presented on the next slide by function *tc*:

- Breadth-first search (BFS) over deterministic finite (Mealy) automaton (DFA) A
- *tc* returns set of input traces representing the transition cover
- α is the "usual" queue used in BFS-algorithms
- $N \subseteq Q(A)$ is an auxiliary subset of A-states which should not be inserted into queue α anymore.
- τ maps states q from where the transition graph of A should be further explored to the previously constructed input trace leading from q_A to q.

Chow's Theorem (21): Transition Cover Construction

function tc(**in** $A : DFA$) : $\mathbb{P}(I^*)$
begin
$tc := \{\langle\ \rangle\};\ \alpha := \langle q_A\rangle;\ N := \{q_A\};\ \tau := \{q_A \mapsto \langle\ \rangle\};$
while $0 < \#\alpha$ **do**
$u = head(\alpha);$
foreach $x \in I$ **do**
$q := \delta_A(u, x);$
$tc := tc \cup \{\tau(u) \frown \langle x\rangle\};$
if $q \notin N$ **then**
$N := N \cup \{q\};$
$\tau := \tau \oplus \{q \mapsto \tau(u) \frown \langle x\rangle\};$
$\alpha := \alpha \frown \langle q\rangle;$
endif
enddo
$\alpha := tail(\alpha);$
enddo
end

Chow's Theorem (22): Characterisation set construction

- Characterisation set W can be generated as a "by-product" of the standard procedure for constructing a minimal DFA A for given DFA A'
- Using a minimal DFA as specification model is not necessary, but desirable for the W-method application, since this keeps the size of the transition cover as small as possible.
- Therefore, given possibly non-minimal DFA A', we simultaneously reduce A' to its minimal DFA A and construct W.
- It is reasonable to assume that
  - A' does not contain any unreachable states q
  - A' has no accepting state (since as a reactive system it should not terminate)

Chow's Theorem (23): Characterisation set construction

Notation:

- $\omega_A : Q(A) \times I \longrightarrow O; \omega_A(q,x) = y \Leftrightarrow (\exists q' \in Q(A) : \delta_A(q,x) = (q', y))$ maps *(Source state,Input)* to the associated output y. In other words, $\omega_A = \pi_2 \circ \delta_A$.
- $\lambda_A : Q(A) \times I \longrightarrow Q(A); \lambda_A(q,x) = q' \Leftrightarrow (\exists y \in O : \delta_A(q,x) = (q', y))$ maps *(Source state,Input)* to the associated target state q', that is, $\lambda_A = \pi_1 \circ \delta_A$.
- We suppose that all states $q, q' \in Q(A)$ are uniquely numbered, so that a relation $<\subseteq Q(A) \times Q(A)$ is well-defined and $q \neq q'$ either implies $q < q'$ or $q' < q$.

Chow's Theorem (24): Characterisation set construction

Notation (continued):

- Specification

$$od : Q(A) \times Q(A) \not\longrightarrow Q(A) \times Q(A)$$
$$od(q, q') = \begin{cases} (q, q') & \text{falls } q < q' \\ (q', q) & \text{falls } q' < q \end{cases}$$

  defines a map on pairs $(q, q') \in Q(A) \times Q(A)$ which sorts pairwise distinct states according to their $<$-order.

- For input traces $w, w' \in I^*$ we write $w < w'$, if w is a true prefix of w'
- $\beta : Q(A) \times Q(A) \not\longrightarrow I^*$ is defined as a function mapping distinguishable states $(q, q') \in Q(A) \times Q(A)$ to non-empty input traces revealing this distinction by producing different outputs when exercised on q and q'.

Chow's Theorem (25): Characterisation set construction

procedure W(**inout** A : DFA, **inout** $W : \mathbb{P}(I^*)$)
begin
  $D : \mathbb{P}(Q(A) \times Q(A))$; // Ordered distinguishable state pairs
  $\beta : Q(A) \times Q(A) \not\longrightarrow I^*$; // Map elements from D to input trace
  $D := \{\}$; $\beta := \{\}$;
  // Initialisation: Insert all ordered pairs of states into D
  // which can be distinguished by a single input
  distinguishedByOne(A, D, β);
  // Identify all distinguishable state pairs, while constructing W
  generateW(A, D, β, W);
  // Optionally, reduce the DFA
  reduceA(A, D, β);
end

Chow's Theorem (26): Characterisation set construction

```
procedure distinguishedByOne(in A : DFA,
                             inout D : ℙ(Q(A) × Q(A)),
                             inout β : Q(A) × Q(A) ⇸ I*)
begin
  foreach p < q ∈ Q(A) × Q(A) do
    foreach x ∈ I do
      if ω_A(p, x) ≠ ω_A(q, x) then
        D := D ∪ {(p, q)};
        β := β ⊕ {(p, q) ↦ ⟨x⟩};
      endif
    enddo
  enddo
end
```

Chow's Theorem (27): Characterisation set construction

```
procedure generateW(in A : DFA,
                    inout D : ℙ(Q(A) × Q(A)),
                    inout β : Q(A) × Q(A) ⇸ I*,
                    out W : ℙ(I*))
begin
  b : bool; b := false;
  do
    foreach p < q ∈ (Q(A) × Q(A)) − D do
      foreach x ∈ I do
        v := λ_A(p, x); z := λ_A(q, x);
        if od(v, z) ∈ D then
          b := true;
          w := ⟨x⟩ ⌢ β(od(v, z));
          //Remove traces which are prefixes of the new (longer) one
          foreach (p′, q′) ∈ D do
            if β(p′, q′) < w then
              β := β ⊕ {(p′, q′) ↦ w};
            endif
          enddo
          β := β ⊕ {(p, q) ↦ w};
          D := D ∪ {(p, q)};
        endif
  while b;
  W := ran(β);
end
```

Chow's Theorem (27): Characterisation set construction

procedure *reduceA*(**inout** A : DFA,
inout $D : \mathbb{P}(Q(A) \times Q(A))$)
begin
$A_r : DFA$;
// Definition of equivalence classes:
// $[p] = \{q \in Q(A) \mid od(p,q) \notin D\}$
// States of the minimised DFA are equivalence classes,
// each class represented by a state p of A which is
// member of a distinguishable pair (p,q) or (q,p) in D.
$Q(A_r) := \{[p] \mid \exists q \in Q(A) : od(p,q) \in D\}$;
$q_{A_r} := [q_A]$;
$\delta_{A_r} := \{([p],x) \mapsto ([\lambda_A(p,x)], \omega_A(p,x)) \mid (p,x) \in Q_A \times I\}$;
// Well-definedness of δ_{A_r} follows from properties of
// equivalence classes $[p]$.
$A := A_r$;
end

Similar results for other formalisms – overview

- Hennessy and deNicola showed that refinement properties can be established by (possibly infinite) number of tests for CCS-like process algebras
- Brinksma and Tretmans produced similar results for conformance testing against Lotos models
- Peleska and Siegel provided solutions for testing against CSP models
- Vandraager et. al. extended Chow's theorem to timed automata

Conclusion of Part I

- Equivalence or refinement proofs by means of exhaustive grey-box testing are possible for untimed and timed automata and process algebras with synchronous (blocking) communication
- Exhaustive testing has exponential complexity in the number of states
- Apart from the complexity problem, the results presented here do not handle the problem of complex data structures and guard conditions: The state space has to be unfolded completely in order to apply the algorithms in a direct way.
  The next part of the tutorial shows how to cope with this problem