Theoretical Computer Science 37 (1985) 151–181
North-Holland

A CONTRIBUTION TO THE THEORY OF PROGRAM TESTING

Luc BOUGÉ

LITP, U.E.R. de Mathématiques, Université Paris 7, 2 Place Jussieu, 75251 Paris, France, and Université d'Orléans, U.E.R. Sciences Fondamentales et Appliquées, Laboratoire d'Informatique, Rue de Chartres, 45046 Orléans, France

Communicated by M. Nivat
Received October 1982
Revised November 1984

Abstract. We propose an abstract formalism for program testing which extends a previous paper of Goodenough and Gerhart. We define the notion of a battery of tests for a given testing context. Its properties are studied: reliability, validity, bias and acceptability. A preorder is defined and studied, which yields an equivalence relation among batteries of tests. This equivalence turns out to be of great interest, both theoretical and practical.

We study the application of this model to some classical questions: effective (automatic) test generation, test optimization, quality assessment of the testing process, and relationships between program proving and program testing.

Contents

Introduction

Program testing, as Goodenough and Gerhart stated some years ago [9], is certainly the most widely used way of assessing program correctness. It is also certainly one of the least studied from a theoretical viewpoint.

Program testing is a part of the program validation process. To validate a program, or better, a *given implementation* of this program, is to *decide*, with a *certain level of confidence*, whether this implementation is correct or incorrect with respect to a *given specification*. We are here mostly interested in functional specifications (a compiler specification for example). The many problems arising in a rigorous approach to program validation have been often analyzed [1, 2, 6, 12].

It has been written that program testing and program proving are essentially two complementary activities, both supporting program correctness assessment [8]. Program proving has received for a long time some very strong and efficient theoretical interest, whereas program testing has been regarded as an empirical and less interesting problem. Some formalization attempts may be found in the scientific literature, but, to our opinion, they do not seem to have tackled the problem at an abstract enough level.

In this paper, we propose an abstract formalism for program testing notions which extends Goodenough and Gerhart's proposal [9]. It uses first-order predicate logic as an underlying mathematical tool (see [13] for an introduction to mathematical logic). Its design is intended to meet two requirements:

(1) to be as faithful as possible to the common intuitive understanding of program testing (for a discussion about this point, see [2]);

(2) to be as faithful as possible to the internal requirements of our underlying mathematical tool, remaining at an abstraction level which enables some powerful references to testing theory, putting into evidence their complementary.

Section 1 sums up briefly some earlier work in this area of research and states the basic framework of this work: the *testing process diagram*.

Main definitions are then stated: *testing context*, and *battery of tests* for a given testing context.

The basic properties of a battery of tests are studied in Section 3: *projective reliability*, *asymptotic validity*, *lack of bias*, *acceptability*. Testability of specifications is discussed according to their syntactical form.

Section 4 is devoted to (pre-) *ordering* testing contexts and batteries of tests for a given context according to their 'quality'. The notions of *conservative context restriction* and *asymptotic sharpness* are introduced. The equivalence relation deduced from the latter is studied, and yields the *Equivalence theorem*: two comparable acceptable batteries of tests are in fact equivalent.

Application to practical problems about testing is finally discussed: *effective test construction method*, *test optimization*, *testing assessment*.

In conclusion, we consider the expressive power gained by using a sound but rather heavy mathematical tool. It makes more precise the better understanding we

have got about the relationships between testing and proving. Some further research goals are described.

> *"We know less about the theory of testing*
> *that we do often*
> *than about the theory of proving*
> *that we do seldom.*
> *This paper is a step toward redressing*
> *this imbalance."*
>
> *Goodenough and Gerhart* [9]

1. Background: The testing process diagram

We sum up here briefly some of the main attempts to express what it is to test a program. Rather than describing their very technical details, we shall emphasize the intuitions which underlie them. We then describe a general model for the testing process: the *testing process diagram.* This model can be mainly characterized by a *formal approach* and an *asymptotic approach* to the notion of testing.

1.1. Goodenough and Gerhart's proposal

Goodenough and Gerhart [9] were some of the first to proclaim the need for a testing theory bearing comparison to proving theory. As a first attempt to solve this problem, they described the following notions.

Consider a *given* implementation of a program, say F, working over an input domain D. Suppose that, for each data d of D, one is able to decide whether the implementation behaves correctly or not with respect to the given specification. This induces a (calculable) predicate over D, say OK; $OK(d)$ means that the implementation behaves 'correctly' for $d \in D$. The correctness of the implementation with respect to the specification is thus expressed by $\forall d \in D\, OK(d)$.

In this framework, a *test* can be viewed as a subset of D, say T (authors seem to allow an infinite subset of D to be a test in their sense). The elements of T can be called test cases, or *elementary experiments.*

One of the main ideas is that "test cases are chosen typically to satisfy some data selection criterion, C, where C denotes a predicate over subsets of D". T is a test if and only if $C(T)$. Thus, a key to testing theory is to focus on the properties of a *criterion* with respect to a certain predicate OK over D instead of considering the properties of a test.

Those authors define two fundamental properties for a criterion C. *Reliability* "refers to the consistency with which results are produced, regardless of whether the results are meaningful". It can be expressed by

$$\begin{aligned}&\textsc{Reliable}(C)\\ &= (\forall T_1, T_2 \subseteq D)[(C(T_1) \wedge C(T_2)) \rightarrow\\ &\qquad (\textsc{Successful}(T_1) \leftrightarrow \textsc{Successful}(T_2))],\end{aligned}$$

where $\textsc{Successful}(T) \equiv \forall t \in T\, OK(t)$.

On the other hand, *validity* "refers to the ability to produce meaningful results, regardless of how consistently such results are produced". It can be expressed by

$$\text{VALID}(C) = (\forall d \in D)[\neg OK(d) \rightarrow (\exists T \subseteq D)(C(T) \vee \neg \text{SUCCESSFUL}(T))].$$

Obviously a 'good' criterion should be both reliable and valid. Such a criterion is said to be *ideal*. The successful execution of any test satisfying such a criterion *demonstrates* the correctness of the implementation with respect to the specification.

Let us point out some features of the above proposal. Note first that it is based on the *functional behaviour* of the implementation being tested (we use the word 'implementation' where they used the word 'program' to underline this fact). Correctness has nothing to do with the internal structure of the program or of the machine executing it. Then, a *first-order logic-like* language arises naturally as the best choice for dealing with those objects, though the reasons for this choice are not stated by those authors. Note that a sentence like $\forall d \in D\, OK(d)$ can be more clearly expressed by $\mathcal{D} \models \forall d\, OK(d)$, where $\mathcal{D}$ is the obvious structure with universe D.

The most important feature of the proposal is, to our mind, the notion of a *criterion* and the properties which are defined about it.

Notice that passing any test satisfying an ideal criterion demonstrates the correctness. So *testing is viewed as a special case of proving* with certain restrictions on the kind of proof that is used. It must in fact be split into a *transcendental* part (proving ideality of the criterion and adequacy of the test) and a *calculable* part (running the test cases).

Lastly, it should be noticed that such a formalism is '*anisotrope*' with respect to correctness. It only deals with successful execution of test cases, and does not say anything about failure. In fact, any failure demonstrates, *in this case*, the uncorrectness of the implementation. But we feel that this 'anisotropy' is a general feature in this area. The famous Dijkstra's statement "Program testing can be used to show the presence of bugs, but never to show their absence!" should be understood in this context. Any testing theory should focus rather on success of tests than on failure, because only the former is actually informative.

1.2. Related work

Several authors, following the above paper, have described related notions. Weyuker [14] has shown the extreme importance of the so-called *Oracle Problem* captured above by the predicate *OK*. One must be able, in a sensible way, to decide whether a given test has been passed or not. It follows that a test should contain at most a finite number of test cases, the successful execution of each being decidable. This will lead us to the notion of an *experiment*.

Weyuker and Ostrand [15] have pointed out the central problem in the testing area, that is, to *infer* an infinitary conclusion $(\forall d \in D\, OK(d))$ from some finite knowledge $(\forall t \in T\, OK(t))$, where T is a finite subset of D. We thus need some a priori infinitary hypothesis. They claim the need to part the domain D into sub-

domains that are *uniform* with respect to correctness. Whenever any data in such a subdomain is correctly handled, so are all of them. A uniform subdomain D_i of D is thus such that $[(\exists d \in D_i\, OK(d)) \rightarrow (\forall d \in D_i\, OK(d))]$ or, more clearly, $\mathscr{D} \models [\exists d (d \in D_i \wedge OK(d))] \rightarrow [\forall d (d \in D_i \rightarrow OK(d))]$. We will call such a hypothesis a *uniformity hypothesis.* Notice that it is only concerned with the specification (the predicate OK) and the given implementation (the structure $\mathscr{D}$), but not with the selection criterion. It is only a *postulate* about the testing context independent of the criterion being used.

Budd, Lipton, Sayward and De Millo [5], followed by Howden [11] have been developing a genuine approach to testing that is called *mutation testing.* Its most important feature is, to our opinion, the consideration not only of the implementation to be tested, but instead of a "neighbourhood of potential implementations". This expresses the idea that we do not know perfectly the implementation we are testing, but only to some extent. We thus have to deal with a set of potential implementations, which the actual one is known to belong to, without being able to pick it out precisely. Any definition we state should only depend on this set rather than on the actual implementation. It must be *uniform* with respect to this set of objects.

This set itself is, in practice, only known through certain hypotheses about the implementation to be tested (among them are, for example, some uniformity hypotheses). Thus, the properties of a selection criterion must depend on those *formal hypotheses,* rather than on the actual implementation being run. We will refer to this principle as the *formal approach* to the notion of testing.

The advantages of this approach are twofold. Though we are only considering implementations as functions, the description of the set of potential implementations will take account of the syntactical features of the given program. It will in fact most likely contain the syntactical mutants of this program, as described by the above authors.

On the other hand, this makes the properties of a criterion independent of the syntactical evolutions of the program (design improvements), which is highly valuable.

Complementarity between testing and proving has been advocated by Gerhart [8]. Depending on the correctness assessment which is considered, one or the other can be used. Geller [7] shows that both methods can even be mixed together to demonstrate that a given property holds. Testing demonstrates that it holds in some particular cases. One then proves that this implies that it holds in all cases. Our theory of testing is specially designed to put into evidence this complementarity. Testing is shown to be a particular kind of proof. Also, extrapolating testing to infinity (infinite cost, infinite information) is shown to prove correctness with certainty.

Gourlay's testing theory [10] provides a unified mathematical framework for most classical testing methodologies, extending Goodenough and Gerhart's proposal. Its theory is mainly concerned with the abstract *reliability* (not to be confused with our projective reliability) of test methods with respect to a set of programs and a set of

specifications. Our theory rather focuses on conditions which let a test method be effective, or at least as effective as proving (oracle problem). Those theories of testing were designed independently but are very similar in their results.

1.3. Reliability, validity and bias

Let us focus on the fundamental properties defined by Goodenough and Gerhart. Firstly, reliability means that all the tests selected by a reliable criterion are equivalent with respect to the correctness assessment of the implementation. Such a criterion should be viewed as 'flat', just as a partial order is. It is obviously more interesting to consider some more complex criteria, where test power (and cost) may vary. It suffices, in fact, to find one test for each level of power (or cost). We are thus led to consider, rather than a flat ordering, a total ordering, for example natural numbers $\mathbb{N}$. Reliability expresses in this case that a test which is 'higher' than another is better for correctness assessment. We call this notion *projective reliability.*

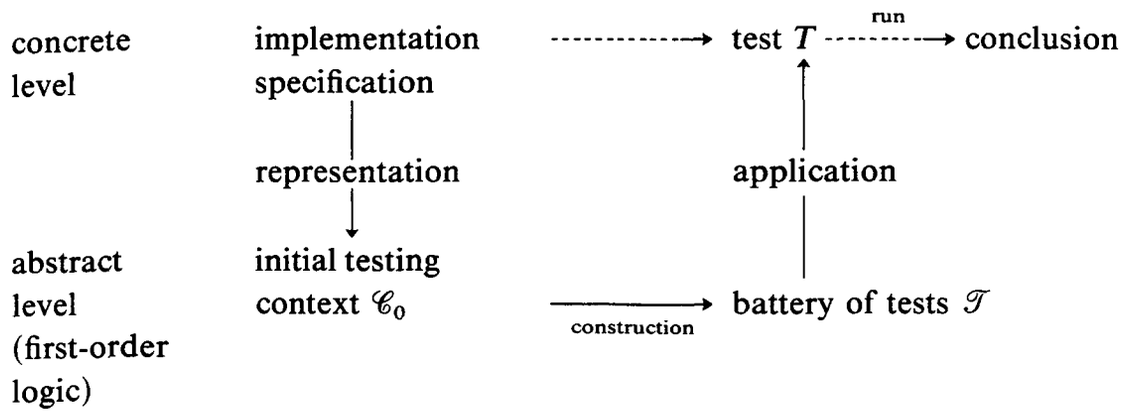
Validity of a criterion C expresses that, for each 'error' in implementation behaviour, one can find in C an adequate test. But we are now dealing with a chain, and we can consider the 'limit' of the tests of C, which is more powerful than any of them, and which can be 'approximated' as closely as wanted if a sufficient cost is allowed. A sensible condition is thus to require this limit test to be adequate for any 'error'. This means that the above adequate test may be rejected to infinity. This is the main feature of what we call the *asymptotic approach* to the notion of testing. We will thus consider *asymptotic validity.*

However, our feeling is that those two notions are not able to give a precise account of the intuitive notion of testing. Statistical test theory is based on the notion of *bias.* A test is said to be unbiased if it is passed by an implementation more probably than failed if and only if this implementation is correct. This could be described as a *soundness property.* This property has ben left implicit by Goodenough and Gerhart since all the tests they consider are actually unbiased, because $\forall d \in D\ OK(d)$ implies $\forall t \in T\ OK(t)$. In other words, whenever the implementation is correct, it will pass any test. In our proposal, which deals with a more abstract notion of testing, the notion of bias arises naturally as a basic property.

1.4. Testing process diagram

We are now in a position to draft the *testing process diagram* (see top of p. 157). It should model the sequence of operations that take place from the definition of the property to be tested, up to the decision of whether the implementation is correct or not with respect to the above property (specification).

From the problem to be solved: "Is this implementation correct with respect to this specification?", we build, in the *representation* phase, an abstract problem, (intuitively) equivalent to the concrete one, expressed in first-order logic. This defines an initial *testing context* $\mathscr{C}_0$.



We *construct* for this testing context an acceptable *battery of tests* exactly as Goodenough and Gerhart were seeking an ideal criterion. However, this will not be possible in general, because at this point we do not have enough knowledge about the implementation to create any precise enough test.

We thus have at first to *postulate* some new hypotheses about the implementation (uniformity hypotheses, typically). We thus *restrict* the initial context $\mathscr{C}_0$ to a new context $\mathscr{C}_1$, for which we can effectively construct an acceptable battery of tests $\mathscr{T}$. Of course, new hypotheses must be consistent with the property to be tested; the restriction must be *conservative.* Note that the construction phase is entirely abstract, and deals only with first-order logical objects.

Having obtained an abstract battery of tests, we can pick some test from it, according to our quality/cost requirements, and *apply* it to the given problem, leading to the conclusion.

2. Basic notions: Testing context and battery of tests

2.1. Mathematical tools

Most of the work in the area of program testing has been carried out in a logic-like mathematical framework ([5, 8, 15, 16], etc.) We choose also first-order mathematical logic as a basis for this work. Yet, one must look carefully at the results we get. They are direct consequences of this choice. Another one, say statistical test formalism for instance, would probably have led to a very different approach.

We use mainly the notation of [13]. Logical validity is denoted by $\models$ and formal provability by $\vdash$. We consider here only first-order languages L. A language is identified with its set of non-logical symbols. If S is a set, $L(S)$ is the language obtained by adding to L the members of S as constants. Extension of languages is denoted by $\subseteq$. If $L \subseteq L'$, then $L(S) \subseteq L'(S)$.

Given a language L and a set S we consider structures obtained by giving meaning to the symbols of L. A structure for L over S yields canonically a structure for $L(S)$ over S. Both will be identified.

A theory T on a language L is a set of L-formulas (axioms). $T \sqcup T'$ denotes the set-theoretic union of T and T'. Observe that $\models T \sqcup T'$ iff $\models T$ and $\models T'$. T is a finite theory if it contains a finite number of (non-logical) axioms.

We write $T \vdash T'$ if all the axioms of T' can be proved using those of T. If $T \vdash T'$ and $T' \vdash T$ then T and T' are formally equivalent $T \dashv\vdash T'$. Observe that $T \vdash (T_1 \sqcup T_2)$ iff $T \vdash T_1$ and $T \vdash T_2$.

Most of our statements will be split into a logical part and a formal part. The logical part expresses as precisely as possible the intuition one has in mind. The formal part is usually more restrictive and, as a rule, it implies the logical one. On the other hand, it allows for handy mathematical treatment. This *duality principle* will be shown to be a very fair bridge between practice and theory.

2.2. Testing context

We make more precise the components of the testing process diagram. Let us first turn to the notion of testing context. This notion should model the problem to be solved (i.e., "Is the implementation correct for the given specification?") independently of the battery of tests we will build to solve it.

Recall that we are interested in the validation of large programs, for example compilers. They are considered by (most) users as 'black boxes'. Users are only concerned with input/output behaviour, and generally not with their internal structure. An abstract data type-like model thus arises naturally. An implementation will be so modelled by its *functional* behaviour. In the first-order logic formalism we use it will be thought of as a *structure.*

We thus take a universe S, usually the domain of the program, and a language L, containing usually at least the functional symbols occurring in the program, plus a symbol representing its functional behaviour. The model of our implementation will be an $L(S)$-structure $\mathscr{S}$. But, as we argued before, we must not only deal with one implementation, but with a family of potential implementations, to which belongs the one under test. We thus deal with a family $(\mathscr{S})$ of $L(S)$-structures.

The given specification to be tested can usually be modelled by a set of first-order formulas. The implementation is then correct if and only if the actual structure abstracting it (recall that we do not know which one it is) validates those formulas. It is not restrictive to consider a (first-order) theory A whose non-logical axioms are those formulas.

2.1. Definition. A *testing context* $\mathscr{C}$ is a 4-uple $\langle L, S, (\mathscr{S}), A\rangle$ where L is a first-order language; S is a set; $(\mathscr{S})$ is a family of $L(S)$-structures; A is an $L(S)$-theory.

2.2. Example. Consider, as an example, that we are testing a square-root program P, which should output the square-root of any natural number given as input. Then a sensible testing context $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ would be the following:

- $L = \{F; \cdot^2; \cdot \leqslant \cdot < \cdot; \cdot + \cdot\}$, the symbol F represents the functional behaviour of the implementation;
- $S = \mathbb{N}$;

- $(\mathcal{S})$ could be the family of the $L(S)$-structures such that the symbols of L different from 'F' have their standard meanings, and such that F has a *calculable meaning*;
- $A = \{\forall x[F(x)^2 \leqslant x < (F(x)+1)^2]\}$.

Notice that many first-order formulas are validated by all the structures of the family $(\mathcal{S})$: for example, $0+1=1$ or $\forall n[n \leqslant n+1 < n+2]$. These formulas can be viewed as some formal hypotheses about the implementation being tested. Again, it is not restrictive to structure them in a $L(S)$-theory H.

This theory H captures formal knowledge we a priori postulate about the implementation. We may, for example, postulate that the implementation is not 'too' incorrect: take $H = \{\forall x[0 \leqslant F(x)^2 < (x+10)]\}$. Then we need only to consider those structures which actually validate H.

2.3. Definition. Let $\mathscr{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context, and let H be an $L(S)$-theory. $\mathscr{C}$ is an *H-context* if every structure of $(\mathcal{S})$ validates H.

Notice that though the family $(\mathcal{S})$ is a part of the testing context, H is not. Even if the actual conditions of the testing process do not change, the formal knowledge we make postulates about may vary (in most cases increase) during the testing process.

2.3. Battery of tests

We can now define what *a battery of tests for a given context* is. As we saw in the first section, a test for a given problem (a given testing context) is a finite set of experiments. An *experiment* is a question about implementation whose answer can be decided finitarily.

2.4. Definition. Let $\mathscr{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context. An *experiment* E for $\mathscr{C}$ is a (closed) $L(S)$-formula *without quantifier*, such that for any non-logical symbol p of E and for any structure $\mathcal{S}$ of $(\mathcal{S})$, the meaning of p in $\mathcal{S}$ is *calculable.*

We do not want to make more precise what 'calculable' means; in most cases, $S = \mathbb{N}$ and we will use the classical definition of a calculable function (predicate). We just note that a (theoretically) calculable, but extremely complex function should not, in practice, be considered to be a 'humanly' calculable one. This is the so-called Oracle Problem [14].

2.5. Definition. Let $\mathscr{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context. A *test* T for $\mathscr{C}$ is an $L(S)$-theory with only a *finite* number of non-logical axioms (*finite theory*), each of them being an experiment of $\mathscr{C}$.

Note that the property that T is a test does not depend on A (the property to be tested), but only on L, S and $(\mathscr{S})$ (the postulated features of the implementation). The distinction between experiment and test is only a matter of convenience, as a test can always be identified with the conjunction of the experiments it is made up of.

Let us now turn to the definition of a *battery of tests for a given context.* It must cover the formal approach and the asymptotic approach we have outlined. A battery of tests should be a family of tests, ordered by a quality criterion that depends only on some formal hypotheses about the implementation. Two batteries of tests built on distinct formal hypotheses must, in our approach, be considered to be distinct.

2.6. Definition. Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a given testing contest. A *battery of tests* $\mathscr{T}$ *for* $\mathscr{C}$ is a pair $\langle H, (T_n)_{n\in\mathbb{N}}\rangle$ where H is an $L(S)$-theory such that $\mathscr{C}$ is an H-context (valid formal hypotheses); $(T_n)_{n\in\mathbb{N}}$ is a family of tests for $\mathscr{C}$ such that for all $n\in\mathbb{N}$ $T_{n+1}\sqcup H \vdash T_n$ (formal projective reliability).

The second condition is discussed more extensively in Section 3.1. We note here that, for any structure $\mathscr{S}$ of $(\mathscr{S})$, and any $n\in\mathbb{N}$, if $\mathscr{S}\models T_{n+1}$, then $\mathscr{S}\models T_n$: tests T_n are more and more precise, uniformly with respect to $(\mathscr{S})$. These properties thus express the essence of the asymptotic approach.

We sometimes write (T_n) instead of $\langle H, (T_n)_{n\in\mathbb{N}}\rangle$ for the sake of conciseness.

2.7. Examples. We consider $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ with $L = \{P\}$, $S = \mathbb{N}$, $A = \{\forall x\, P(x)\}$, and $(\mathscr{S})$ being the family of the $L(S)$-structures such that P has a calculable meaning. $\mathscr{C}$ is a $\emptyset$-testing context.

(1) Take $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$, with $H = \emptyset$ and $T_n = \emptyset\ \forall n\in\mathbb{N}$. $\mathscr{T}$ is actually a battery of tests for $\mathscr{C}$. It is called the empty battery of tests for $\mathscr{C}$.

(2) Take $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ with $H = \emptyset$ and $T_n = \{P(0)\wedge\cdots\wedge P(n)\}$. $\mathscr{T}$ is also a battery of tests for $\mathscr{C}$. It corresponds to the exhaustive testing strategy for A; if a structure $\mathscr{S}$ validates all the T_n, then $\mathscr{S}$ validates A. But notice that $H \bigsqcup_{n\in\mathbb{N}} T_n$ does not prove A.

(3) Idem with $T_n = \{P(n)\wedge\cdots\wedge P(0)\}$. This *new* battery of tests should obviously be equivalent to the previous one for any sensible notion of equivalence (and fortunately it will, see Section 4.3).

One can easily generalize these examples with an enumeration (possibly not monic) of $\mathbb{N}$.

(4) Consider $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$, two batteries of tests for a testing context $\mathscr{C}$. Then it is easy to prove that $\mathscr{T}\sqcup\mathscr{T}' = \langle H\sqcup H', (T_n\sqcup T'_n)_{n\in\mathbb{N}}\rangle$ is also a battery of tests for $\mathscr{C}$. Note that the empty battery of tests is a neutral element for union.

(5) Consider another testing context $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$, with L, S and $(\mathscr{S})$ as before, and $A = \{P(0), P(1), \ldots\}$ $(= \{P(i), i\in\mathbb{N}\})$. Notice that for any $L(S)$-structure $\mathscr{S}$,

$$\mathscr{S}\models\{\forall x\, P(x)\} \text{ if and only if } \mathscr{S}\models\{P(0), P(1), \ldots\},$$

but that the latter theory *cannot prove* the former. Look back at example (2). It is still a battery of tests for this new context, but we now have the property $H \sqcup_{n\in\mathbb{N}} T_n \vdash A$. Yet, for any $k\in\mathbb{N}$, $H \sqcup_{n\leq k} T_n \nvdash A$.

Let us consider a more restricted (and realistic) family $(\mathcal{S})$ of potential implementations. Consider for example, $k\in\mathbb{N}$ being fixed, the theory H

$$H = \{P(0)\wedge\cdots\wedge P(k)\rightarrow\forall x\, P(x)\}.$$

This hypothesis means that, for any potential implementation, if this implementation works correctly for any data of 'complexity' less than k then it works correctly for any data. This is a basic (but often hidden) hypothesis in path analysis testing methods where implicitly the 'complexity' of a data is the 'complexity' of its path in the program.

Consider the H-context $\mathcal{C} = \langle L, S, (\mathcal{S}), A\rangle$, with $L=\{P\}$, $S=\mathbb{N}$, $A=\{\forall x\, P(x)\}$ and $(\mathcal{S})$ being the family of all those $L(S)$-structures $\mathcal{S}$ such that $\mathcal{S}\models H$ and the meaning of P for $\mathcal{S}$ is calculable. Then $\mathcal{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$, with $T_n = \{P(0)\wedge\cdots\wedge P(n)\}$ is a battery of tests for $\mathcal{C}$, with the property $H \sqcup_{n\in\mathbb{N}} T_n \vdash A$ (in fact, $H \sqcup T_k \vdash A$, and this remark is general, see Section 3.2).

This kind of hypothesis is called a *regularity hypothesis.* Another classical (and hidden) hypothesis in path analysis testing strategy is the *uniformity hypothesis.* Two data items which follow the same path in the program are equivalent with respect to correctness: if one is correctly handled by the implementation, the other will be as well. We can give an account of this fact by postulating a hypothesis $H = \{\exists x\, P(x)\rightarrow\forall x\, P(x)\}$. These two kinds of hypotheses play an essential role in this work. In particular we can see that a regularity hypothesis is minimal in some sense in order to get a 'good' battery of tests, and that using different constants leads to equivalent batteries of tests (*virtual constants*).

3. Fundamental properties of batteries of tests

We have defined the notions of a testing context and a battery of tests for a given testing context. Those definitions extend Goodenough and Gerhart's ones; also, they take care of the asymptotic approach to the notion of testing, and of the fact that the implementation is only partially known, if not partially specified.

3.1. Reliability

The reliability property expresses the internal consistency of the battery of tests which is being considered [9, 15]. From our viewpoint, consistency means that a test with a higher index is more powerful, with respect to success, than a test with a lower index. In fact, for technical reasons, this property was already required when we defined a battery of tests for a given testing context.

3.1. Definition. Let $\mathcal{C} = \langle L, S, (\mathcal{S}), A\rangle$ be an H-context, and $(T_n)_{n\in\mathbb{N}}$ a countable family of tests for this context.

$(T_n)_{n\in\mathbb{N}}$ is *logically projectively reliable* if for every $n\in\mathbb{N}$ and for every structure $\mathcal{S}$ of $(\mathcal{S})$ such that $\mathcal{S}\models T_{n+1}$ then $\mathcal{S}\models T_n$.

$(T_n)_{n\in\mathbb{N}}$ is *formally projectively reliable* if for every $n\in\mathbb{N}$ $T_{n+1}\sqcup H\vdash T_n$.

Obviously, since $\mathcal{C}$ is an H-context, the second statement implies the first one. As we stated before, a battery of tests for a given context (more precisely, its family of tests) is formally projectively reliable (and thus, logically too). We will only deal with such families of tests: in practice, it would be strange to consider a collection of tests (remember that a test is a set of experiments) such that an expensive test is not necessarily better than a cheaper one!

3.2. Validity

We now turn to validity. Reliability was concerned with the consistency of a family of tests. Validity is concerned with its power and its usefulness. Goodenough and Gerhart gave a strong definition of this property: a collection (criterion) is valid if, whenever the program is incorrect, it fails at least some test. Our formalism allows us to give a slightly more general statement: the failed test may be incidentally rejected to infinity, i.e. may be (virtually) T_∞.

3.2. Definition. Let $\mathcal{C}=\langle L, S, (\mathcal{S}), A\rangle$ be a testing context, and $\mathcal{T}=\langle H, (T_n)_{n\in\mathbb{N}}\rangle$ a battery of tests for this context.

$\mathcal{T}$ is *logically asymptotically valid* if for every structure $\mathcal{S}$ of $(\mathcal{S})$, if $\mathcal{S}\models T_n$ for every $n\in\mathbb{N}$ then $\mathcal{S}\models A$.

$\mathcal{T}$ is *formally asymptotically valid* if $\bigsqcup_{n\in\mathbb{N}} T_n\sqcup H\vdash A$.

Again, the second statement implies the first one, because every $\mathcal{S}$ of $(\mathcal{S})$ is such that $\mathcal{S}\models H$. Notice that by (formal projective) reliability, the first statement is equivalent to the following one:

for every $\mathcal{S}\in(\mathcal{S})$, there exists $n_{\mathcal{S}}\in\mathbb{N}$ such that

if $\mathcal{S}\models T_n\ \forall n\geqslant n_{\mathcal{S}}$, then $\mathcal{S}\models A$.

In the same way, the second one can be written

$$\exists n_0\in\mathbb{N}\qquad \bigsqcup_{n\geqslant n_0} T_n\sqcup H\vdash A.$$

These facts justify the informal description stated above. As before, we write validity for logical or formal asymptotic validity when the addressed property is evident.

3.3. Examples. (1) Let $\mathcal{C}$ be a context, and $\mathcal{T}=\langle H, (T_n)_{n\in\mathbb{N}}\rangle$, with $H=\emptyset$, $T_n=\emptyset$ for all $n\in\mathbb{N}$. $\mathcal{T}$ is formally valid (for $\mathcal{C}$) iff $A=\emptyset$.

(2) Let $\mathcal{C}=\langle L, S, (\mathcal{S}), A\rangle$ be a context and $\mathcal{T}=\langle A, (T_n)_{n\in\mathbb{N}}\rangle$ a battery of tests for $\mathcal{C}$. This implies that every structure $\mathcal{S}$ of $(\mathcal{S})$ is a model of A, i.e., the property

being tested is satisfied by all the potential implementations being considered. $\mathcal{T}$ is formally valid for $\mathcal{C}$.

(3) Let $\mathcal{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a context, with $L = \{P\}$, $S = \mathbb{N}$, $A = \{\forall x\, P(x)\}$, and $(\mathcal{S})$ being the family of all the $L(S)$-structures such that P has a calculable meaning. Take $T_n = \{P(0) \wedge \cdots \wedge P(n)\}$. Then $\mathcal{T} = \langle \emptyset, (T_n)_{n\in\mathbb{N}}\rangle$ is a logically asymptotically valid battery of tests for $\mathcal{C}$, but *not* a formally valid one, because $\{P(n), n \in \mathbb{N}\} \nvdash \forall x\, P(x)$.

(4) Take $\mathcal{C}$ as in (3), but restrict $(\mathcal{S})$ to those structures such that $\mathcal{S} \models H$, with $H = \{P(0) \wedge \cdots \wedge P(k) \to \forall x\, P(x)\}$. Then $\mathcal{T}$ is formally valid. Notice that this does not depend on the value of k. k is called a virtual constant. In the example (3), we had virtually $k = \infty$. H is called a *regularity hypothesis.*

(5) As (4), but now with $H = \{\exists x\, P(x) \to \forall x\, P(x)\}$ the same conclusions hold. Following Weyuker and Ostrand [15], H is called a *uniformity hypothesis.*

Notice that in Example 3.3(4), $\mathcal{T}$ is in fact valid in a stronger sense, since $H \sqcup T_k \vdash A$. This fact is general indeed. More precisely, we define the finite validity as follows.

3.4. Definition. Let $\mathcal{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context, and $\mathcal{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ a battery of tests for $\mathcal{C}$. $\mathcal{T}$ is *formally finitely valid if* $\exists n_0 \in \mathbb{N}\ T_{n_0} \sqcup H \vdash A$.

Note that if $\mathcal{T}$ is finitely valid then $\forall n \geqslant n_0\ T_n \sqcup H \vdash A$, so $\mathcal{T}$ is formally asymptotically valid. Recall that a *finite theory* is a theory having only a finite set of non-logical axioms. A straightforward consequence of the compactness theorem of the first-order logic is the following.

3.5. Compactness Theorem. *Let $\mathcal{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context, with A a finite theory, and $\mathcal{T}$ a battery of tests for $\mathcal{C}$. $\mathcal{T}$ is formally asymptotically valid iff it is formally finitely valid.*

Proof. If $\mathcal{T}$ is asymptotically valid, $H \bigsqcup_n T_n \vdash A$. But each non-logical axiom of A can be proved by using only finitely many axioms of $H \bigsqcup_n T_n$. Because of the finiteness of A, we can find $i_1, \ldots, i_p$ such that $H \sqcup T_{i_1} \sqcup \cdots \sqcup T_{i_p} \vdash A$. Take $n_0 = \sup\{i_1, \ldots, i_p\}$. By reliability,

$$H \sqcup T_{n_0} \vdash H \sqcup T_{i_k} \quad \text{for } k = 1, \ldots, p.$$

So $H \sqcup T_{n_0} \vdash A$. □

Thus, if A is finite, our definition is nothing more than a restatement of the previous one. If A is *not* finite, this is not true. Take $H = \emptyset$, $A = \{P(n), n \in \mathbb{N}\}$; $T_n = \{P(0) \wedge \cdots \wedge P(n)\}$ is formally asymptotically valid, but *not* finitely valid.

Example 3.3(3) shows that no logical dual of this theorem may be expected. In this case, and in many others, formal properties are more fruitful in their theoretical

consequences than the logical ones. They allow us to understand better the relationships between the many objects we are dealing with. On the other hand, they are sometimes too strong with respect to our intuitive perception of the testing process, which leads us to use logical definitions. Our approach, taking account of duality, is a possible mathematical answer to bridge the gap between theory and practice.

3.3. Bias

The concept of bias is fundamental in statistics. A (statistical) test is said to be unbiased if it leads to the right conclusion with a higher probability than to the wrong one. Because we do not have any probability concept in our formalism, we simply state that a battery of tests is unbiased if any correct implementation passes all the tests. So, if the tested implementation fails a test, we can conclude that it is certainly incorrect with respect to the considered specification. This is therefore a basic requirement for a battery of tests.

3.6. Definition. Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a testing context, and let $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ be a battery of tests for this context.

$\mathscr{T}$ is said to be *logically unbiased* if for every structure $\mathscr{S}$ of $(\mathscr{S})$ such that $\mathscr{S} \models A$ then $\mathscr{S} \models T_n$ for all $n \in \mathbb{N}$.

$\mathscr{T}$ is said to be formally unbiased if $H \sqcup A \vdash T_n \ \forall n \in \mathbb{N}$.

Once again, the second statement implies the first one. Both are the converses of the corresponding asymptotic validity definitions. A similar property is stated by Gourlay [10]. In fact, lack of bias was implicit in most of the previous work on this subject. Let us consider for example Goodenough and Gerhart's theory. A test T is a (finite) subset of the input domain D of P. The implicit theory associated with it is

$$T = \{\forall t \in T\, OK(t)\}.$$

The implicit theory A to be tested is

$$A = \{\forall d \in D\, OK(d)\},$$

i.e., the program is correct for every input value. But, of course, $A \vdash T$, so the criterion $C = \{T\}$ is formally unbiased. Our formalism is more general than their's, and allows us to point out this property, left implicit so far.

3.7. Examples. (1) Let $\mathscr{C}$ be a context, and $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$, where $H = \emptyset$, $T_n = \emptyset$ for all $n \in \mathbb{N}$. $\mathscr{T}$ is formally unbiased.

(2) Take now $\mathscr{T} = \langle A, (T_n)_{n\in\mathbb{N}}\rangle$; it is unbiased too.

(3) Take $S = \mathbb{N}$, $H = \emptyset$, $A = \{\forall x(\neg(x = 0) \to P(x))\}$; let $(\mathscr{S})$ be the family of the $L(S)$-structures such that P has a calculable meaning. Then $T_n = \{P(0) \wedge \cdots \wedge P(n)\}$ *is* biased (not even logically unbiased). But $T'_n = \{P(1) \wedge \cdots \wedge P(n)\}$ for $n \geqslant 1$ with $T_0 = \emptyset$ is unbiased.

(4) (*non-calculable properties*) Consider the case where a property about a non-calculable symbol is being tested. The natural way to test it is to consider a stronger calculable property. Take $S=\mathbb{N}$, $H=\{\forall x(Q(x)\rightarrow P(x)\}$, $A=\{P(0)\}$, with $(\mathscr{S})$ the family of the $L(S)$-structures such that Q (and not necessarily P) has a calculable meaning. The natural battery of tests is then $T_n=\{Q(0)\}$. It is valid, but biased.

(5) (*existential properties*) Take $S=\mathbb{N}$, $L=\{P\}$, $A=\{\exists x\, P(x)\}$, and let $(\mathscr{S})$ be the family of the $L(S)$-structures such that P has calculable meaning. Take $H=\emptyset$. It can be shown that the only $\mathscr{T}=\langle H,(T_n)_{n\in\mathbb{N}}\rangle$ unbiased in the context $\mathscr{C}=\langle L, S, (\mathscr{S}), A\rangle$ is such that $T_n=\emptyset$ for all $n\in\mathbb{N}$.

If we can now restrict $(\mathscr{S})$ to those structures that validate $H=\{\exists x\, P(x)\rightarrow P(0)\vee\cdots\vee P(k)\}$, with k a (virtual) constant, then $T_n=\{P(0)\vee\cdots\vee P(k)\}$ is unbiased (and valid), but depends on k.

3.4. Acceptability

We can now define what a 'good' battery of tests for a given problem is. We do not want to reject a correct program, so we need the lack of bias property. We obviously cannot demand conversely, that a program which passes some test is necessarily correct. Yet we can require that any incorrect program fails some test, namely asymptotic validity.

3.8. Definition. Let $\mathscr{C}$ be a testing context, and $\mathscr{T}$ a battery of tests for this context.

$\mathscr{T}$ is said to be *logically acceptable* if it is logically asymptotically valid and logically unbiased.

$\mathscr{T}$ is said to be *formally acceptable* if it is formally asymptotically valid and formally unbiased.

From the previous remarks, it is obvious that formal acceptability implies logical acceptability. With the now usual notation, logical acceptability means that for every structure $\mathscr{S}$ of $(\mathscr{S})$

$$\mathscr{S}\models T_n \text{ for all } n\in\mathbb{N} \text{ iff } \mathscr{S}\models A.$$

Formal acceptability simply means

$$H \bigsqcup_{n\in\mathbb{N}} T_n \vdash\!\dashv H \sqcup A,$$

i.e., with respect to H (the formal hypotheses about the program), A (the property being tested) is equivalent to $\bigsqcup_{n\in\mathbb{N}} T_n$ (which can be viewed as T_∞).

We define a 'good' battery of tests for a given context as a (*formally*) *acceptable* one. As we pointed out before, a natural choice might rather have been logical acceptability. But in this case, the acceptability of a given battery of tests would have been very strongly dependent on the underlying testing context (more precisely, on $(\mathscr{S})$). The context can be viewed as the way of using the battery of tests, and in practice, it is not a well-defined entity. Specifically, it is very difficult to define

precisely the family $(\mathscr{S})$ of all the potential implementations of the program (including incorrect implementations). Therefore, we choose a more intrinsic notion: namely, formal acceptability. This property can be stated as an entirely abstract relation between abstract entities A, H, and $(T_n)_{n\in\mathbb{N}}$. It therefore depends on the way of using the battery of tests *only through H*, that is the formal hypotheses about the implementation being considered. The more precise they are, the closer is this notion to logical acceptability.

3.9. Examples. (1) Let $\mathscr{C}=\langle L, S, (\mathscr{S}), A\rangle$ be an A-context. Then $\mathscr{T}=\langle A, (T_n)_{n\in\mathbb{N}}\rangle$ with $T_n=\emptyset$, is (formally) acceptable. The problem is therefore to describe an acceptable battery of tests *with hypotheses H as weak as possible.*

(2) Take $L=\{P\}$, $S=\mathbb{N}$, $H=\emptyset$, $A=\{P(n), n\in\mathbb{N}\}$. Let $(\mathscr{S})$ be a family of $L(S)$-structures such that P has a calculable meaning in each of them; then $T_n=\{P(0)\wedge\cdots\wedge P(n)\}$ is acceptable.

(3) The same with $A=\{\forall x\, P(x)\}$. Notice that, with respect to $(\mathscr{S})$, $\{P(n), n\in\mathbb{N}\}$ and $\{\forall x\, P(x)\}$ are logically equivalent. But now, $(T_n)_{n\in\mathbb{N}}$ is *not* acceptable.

(4) Let us restrict $(\mathscr{S})$ to those structures which validate the *regularity hypothesis* $H=\{P(0)\wedge\cdots\wedge P(k)\to\forall x\, P(x)\}$, k being a virtual constant. Then $(T_n)_{n\in\mathbb{N}}$ is acceptable: it is nothing more than *the exhaustive testing strategy.*

(5) Instead of a regularity hypothesis, let us consider a *uniformity hypothesis* $H=\{\exists x\, P(x)\to\forall x\, P(x)\}$. Add to the language, if there is not one already, a special constant symbol a. Then $T_n=\{P(a)\}$ is acceptable. This is actually the *random sampling testing strategy.*

(6) (*existential properties*) In Section 3.3, we examined the case of not purely universal properties, for instance $\exists x\, P(x)$. We can construct an acceptable battery of tests by assuming the hypothesis $H=\{\exists x\, P(x)\to P(0)\vee\cdots\vee P(k)\}$. Note that this can be written $\neg P(0)\wedge\cdots\wedge\neg P(k)\to\forall x\,\neg P(x)$, i.e., as a regularity hypothesis for $\neg P$. Note also that the natural family of tests is then $T_n=\{P(0)\vee\cdots\vee P(k)\}$, *which depends on k.* In fact, it is *not* possible to build any acceptable family $(T_n)_{n\in\mathbb{N}}$ that does not depend on k.

This is why we state informally that existential (more precisely not purely universal) properties are *not testable.* Another good reason will be stated in Section 4.1.

(7) (*non-calculable properties*) We already noticed that non-calculable properties 'naturally' lead to biased batteries of tests.

(8) (*mutation testing strategy*) This strategy (including testing-quality measure) has been described by Lipton et al. (see [5] for details and references). It can easily be expressed in our framework.

Let $\mathscr{C}=\langle L, S, (\mathscr{S}), A\rangle$ be a testing context, and $\mathscr{T}=\langle H, (T_n)_{n\in\mathbb{N}}\rangle$ a battery of tests for $\mathscr{C}$. By the *Competent Programmer Hypothesis* there exists at least one $\mathscr{S}_0$ in $(\mathscr{S})$ such that $\mathscr{S}_0\models A$. Suppose that $(T_n)_{n\in\mathbb{N}}$ discriminates between the structures of $(\mathscr{S})$: for every pair of distinct structures $\mathscr{S}, \mathscr{S}'$ of $(\mathscr{S})$, there exists $n\in\mathbb{N}$ such that $\neg[(\mathscr{S}\models T_n)\wedge(\mathscr{S}'\models T_n)]$. Suppose that $\mathscr{T}$ is (logically) unbiased (this hypothesis is in most cases left implicit). Then $\mathscr{T}$ is *logically acceptable* for $\mathscr{C}$. It suffices to show

that it is logically asymptotically valid. Note that, by lack of bias, $\mathscr{S}_0 \models T_n \ \forall n \in \mathbb{N}$. Consider now any structure $\mathscr{S} \in (\mathscr{S})$ such that $\mathscr{S} \models T_n \ \forall n \in \mathbb{N}$. Because (T_n) discriminates between the structures of $(\mathscr{S})$, we can conclude that $\mathscr{S}$ actually is $\mathscr{S}_0$, so that $\mathscr{S} \models A$.

We defined a 'good' battery of tests to be a *formally* valid and unbiased one. By this choice, we got a quite intrinsic notion, which depends on the actual utilisation conditions only through the formal hypotheses H. This means that the acceptability property is very stable with respect to the evolution of the actual implementation being tested. Furthermore, it gives us more confidence when extrapolating, in the conclusion of the testing process, the success of one test say T_p, to the success of all the tests T_n, $n \in \mathbb{N}$.

4. Ordering contexts and batteries

We now turn to some new ideas. In the previous examples we sometimes found it necessary to restrict the family of structures $(\mathscr{S})$ to construct an acceptable battery of tests. That meant that, in some way, we had obtained more knowledge about the program being tested. The new context was then more precise than the old one. This remark leads us to a preorder on testing contexts.

On the other hand, once a context has been given, we need to compare its batteries of tests. Thus, we want to describe precisely what to improve, to optimize, a testing strategy means. We want to describe, as well, what two equivalent testing strategies are. This leads us to a preorder on batteries of tests for a given testing context. The induced equivalence turns out to be of great interest.

4.1. Conservative restriction of a testing context

In most cases, there does not exist any acceptable battery of tests for a given context $\mathscr{C} = \langle L, S, (\mathscr{S}), A \rangle$. One reason could be that L does not allow us to express precise enough properties. Another more crucial reason could be that $(\mathscr{S})$ is too large, too heterogeneous. We need to restrict it to a more homogeneous family, but without eliminating any correct potential implementations of the program being tested. The restriction must be *conservative.*

Let L be a language, and L' an extension of L. Let $\mathscr{S}'$ be an L'-structure. We denote by $\mathscr{S}'|_L$ the L-structure obtained by removing the symbols of $L' - L$. Let A be a L-theory. We denote by $A|_{L'}$ the L'-theory having the same non-logical axioms as A (but considered as L'-formulas).

4.1. Definition. Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A \rangle$ and $\mathscr{C}' = \langle L', S', (\mathscr{S}'), A' \rangle$ be two testing contexts. $\mathscr{C}'$ is a *conservative restriction* of $\mathscr{C}$ ($\mathscr{C}' \subseteq \mathscr{C}$) if:

- L' is an extension of L,
- $S' = S$,

- $(\mathcal{S}')|_{L(S)} \subseteq (\mathcal{S})$,
- $A|_{L'(\mathcal{S})} = A'$;
- *conservation condition*: for every structure $\mathcal{S}$ of $(\mathcal{S})$ such that $\mathcal{S} \models A$, there exists a structure $\mathcal{S}'$ of $(\mathcal{S}')$ such that

$$\mathcal{S}'|_{L(S)} = \mathcal{S}.$$

Note that if L' is an extension of L then $L'(S)$ is an extension of $L(S)$. Note, too, that for any $L'(S)$-structure $\mathcal{S}'$, $\mathcal{S}' \models A|_{L'(S)}$ iff $\mathcal{S}'|_{L(S)} \models A$, because we are dealing with *the same universe* S in both cases. It can easily be shown that the relation $\subseteq$ is a preorder on the class of testing contexts. Therefore, by transitivity, we may deal only with elementary conservative restrictions, putting them together to get the desired result.

The following theorem shows that a (conservative) restriction cannot, in any case, discard any acceptable battery of tests. It is therefore a safe tool.

4.2. (Conservative) Restriction Theorem. *Let $\mathscr{C} = \langle L, S, (\mathcal{S}), A\rangle$ be a testing context, and let $\mathscr{C}' = \langle L', S', (\mathcal{S}'), A'\rangle$ be a conservative restriction of $\mathscr{C}$. Let $\mathcal{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ be a battery of tests for $\mathscr{C}$. Let $\mathcal{T}' = \langle H|_{L'(S)}, (T_n|_{L'(S)})_{n\in\mathbb{N}}\rangle$.*

If $\mathcal{T}$ is logically acceptable for $\mathscr{C}$, so is $\mathcal{T}'$ for $\mathscr{C}'$. If $\mathcal{T}$ is formally acceptable for $\mathscr{C}$, so is $\mathcal{T}'$ for $\mathscr{C}'$.

4.3. Lemma. *Let U and V be two L-theories and L' be an extension of L. Then $U \vdash V$ iff $U|_{L'} \vdash V|_{L'}$.*

Proof of Theorem 4.2. We show that $\mathcal{T}'$ is actually a battery of tests for $\mathscr{C}'$. Let $\mathcal{S}' \in (\mathcal{S}')$; then $\mathcal{S}'|_{L(S)} \in (\mathcal{S})$, $\mathcal{S}'|_{L(S)} \models H$ and $\mathcal{S}' \models H|_{L'(S)}$. Because $(\mathcal{S}')|_{L(S)} \subseteq (\mathcal{S})$, the calculability of the meanings of the non-logical symbols of T_n is preserved. $T_n|_{L'(S)}$ is thus actually a test for $\mathscr{C}'$. By applying Lemma 4.3 with $L(S)$ and $L'(S)$, reliability is obvious.

Suppose $\mathcal{T}$ is logically acceptable for $\mathscr{C}$. For any $\mathcal{S}' \in (\mathcal{S}')$ $\mathcal{S}' \models A|_{L'(S)}$ iff $\mathcal{S}'|_{L(S)} \models A$ iff $\mathcal{S}'|_{L(S)} \models T_n$ for any $n \in \mathbb{N}$ (because $\mathcal{S}'|_{L(S)} \in (\mathcal{S})$) iff $\mathcal{S}' \models T_n|_{L'(S)}$ for any $n \in \mathbb{N}$. Thus $\mathcal{T}'$ is logically acceptable for $\mathscr{C}'$.

Suppose $\mathcal{T}$ is formally acceptable for $\mathscr{C}$. By Lemma 4.3, the relation $H \bigsqcup_n T_n \vdash\!\dashv H \sqcup A$ implies

$$H|_{L'(S)} \bigsqcup_n T_n|_{L'(S)} \vdash\!\dashv H|_{L'(S)} \sqcup A|_{L'(S)},$$

which expresses formal acceptability of $\mathcal{T}'$ for $\mathscr{C}'$. □

Note that the converse of the above theorem is true in the formal case: a battery of tests that was not formally acceptable *cannot* become acceptable as if by (conservative) magic! This means that formal acceptability is a rather intrinsic property which depends little on the concrete application conditions. Dependence only occurs through the formal hypotheses H.

Yet, this is not true in the logical case. Logical acceptability may depend on external conditions. This feature agrees with practical intuition, but leads to conceptual difficulties.

Conservative restrictions of testing contexts are used in two main ways in practice. On the one hand, we add some new symbols to L, extending the family of structures by giving to those new symbols some specific meanings. For instance, in Example 3.9(5) the new constant symbol a models an arbitrary element of the domain. To choose a structure $\mathscr{S}'$ of the restricted context means to pick up randomly a value in S. We thus give a precise model of *random sampling strategy.*

On the other hand, we will use some new hypotheses H_1 about the potential implementation, for example the regularity or uniformity hypotheses. We then restrict $(\mathscr{S})$ to those structures that validate H_1. It can be easily shown that a sufficient condition for the restriction to be conservative is $A \sqcup H \vdash H_1$. The restricted context is then an $H \sqcup H_1$-context.

As a special case, we can take $H_1 = A$. We then get an A-context. A trivial acceptable battery of tests is $\mathscr{T} = \langle A, (T_n)_{n \in \mathbb{N}} \rangle$ with $T_n = \emptyset$ for all $n \in \mathbb{N}$. If we consider the property to be tested as hypothesis, testing becomes trivial!

This is why we must only consider some *likely* hypotheses H_1. A must obviously not be considered as a likely hypothesis. The regularity or uniformity hypothesis may, in most cases. This 'likely' feature will get a crucial place for quality assessment.

For purely universal properties, the regularity or uniformity hypothesis leads actually to conservative restrictions. The reason is that

$$[\forall x\, P(x)] \to [P(0) \wedge \cdots \wedge P(k) \to \forall x\, P(x)]$$

is a theorem (for any k). But this is not true in general. In Example 3.7(5) consider the structure $\mathscr{S}$ such that $P_{\mathscr{S}}(x) = (x = k+1)$ then $\mathscr{S} \models A = \{\exists x\, P(x)\}$, but $\mathscr{S}$ does not validate $H_1 = \{\exists x\, P(x) \to P(0) \vee \cdots \vee P(k)\}$. The restriction is not conservative; we implicitly eliminated some correct potential implementations of our program. That is why *properties which are not purely universal are not testable in general.*

4.2. Asymptotic sharpness

We now consider a given testing context, and its set of batteries of tests. Informally, we need to compare two of them with respect to the quality of information we can gain. But we consider only positive information, i.e., information about correctness of the implementation, and not information about its possible incorrectness. As with conservative restrictions, we are interested rather in correct implementations than in incorrect ones.

A battery of tests is sharper than another if the success of its tests implies the success of the tests of the other, irrespective of indexes. This is the intuitive, logical definition. For the formal one, we have to deal with formal hypotheses H and H'. A battery of tests should involve hypotheses as weak as possible, because we do not know whether they are actually valid for the actual implementation. Our definition must reflect this viewpoint.

4.4. Definition. Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a testing context. Let $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$ be two batteries of tests for $\mathscr{C}$.

$\mathscr{T}$ is *logically asymptotically sharper* than $\mathscr{T}'$ if for every $i\in\mathbb{N}$ there exists $j\in\mathbb{N}$, such that for every structure $\mathscr{S}$ of $(\mathscr{S})$ if $\mathscr{S}\models T_j$ then $\mathscr{S}\models T'_i$.

$\mathscr{T}$ is *formally asymptotically sharper* than $\mathscr{T}'$ if $\forall i\in\mathbb{N}\ \exists j\in\mathbb{N}\ H\sqcup T_j\vdash H'\sqcup T'_i$ and $H'\sqcup A\vdash H\sqcup A$.

As with previous definitions formal sharpness implies logical sharpness. Both of them define partial orders on the set of batteries of tests for the context $\mathscr{C}$. However, the logical order is more dense than the formal one. For example, two batteries of tests have a logical least upper bound, but not necessarily a formal one. The sharpness orderings will be denoted $\geqslant$. The equivalence relations canonically associated with them will be denoted $\sim$.

4.5. Examples. (1) Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a context, and $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ be a battery of tests for $\mathscr{C}$ such that $A\vdash H$ (this is in practice often true). Then $\mathscr{T}' = \langle\emptyset, (\emptyset)_{n\in\mathbb{N}}\rangle$ is a battery of tests for $\mathscr{C}$, and $\mathscr{T}'\leqslant\mathscr{T}$ strictly.

(2) Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a context, and let $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$ be two batteries of tests for $\mathscr{C}$. Then $\mathscr{T}\sqcup\mathscr{T}' = \langle H\sqcup H', (T_n\sqcup T'_n)_{n\in\mathbb{N}}\rangle$ is a battery of tests for $\mathscr{C}$, and it is the least upper bound of $\mathscr{T}$ and $\mathscr{T}'$ for logical sharpness. If $A\sqcup H\vdash\!\dashv A\sqcup H'$ then this holds for formal sharpness as well.

(3) Let $\mathscr{C}$ be a context, and $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ be a battery of tests for $\mathscr{C}$. Let $\mathscr{T}' = \langle H, (T'_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}'' = \langle H, (T''_n)_{n\in\mathbb{N}}\rangle$ with $T'_n = T_{2n}$ and $T''_n = T_{\lfloor n/2\rfloor}$. Then $\mathscr{T}'$ and $\mathscr{T}''$ are two batteries of tests for $\mathscr{C}$, formally equivalent to $\mathscr{T}$.

(4) Take $L = \{P, Q\}$, $S = \mathbb{N}$, $H = H' = \{\forall x(Q(x)\to P(x))\}$. $A = \{\forall x\, P(x)\}$. Consider $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$ where $T_n = \{P(0)\wedge\cdots\wedge P(n)\}$, $T'_n = \{Q(0)\wedge\cdots\wedge Q(n)\}$. Then $\mathscr{T}'\geqslant\mathscr{T}$ strictly.

(5) Take $L = \{P\}$, $S = \mathbb{N}$, $A = \{P(n), n\in\mathbb{N}\}$, $T_n = \{P(0)\wedge\cdots\wedge P(n)\}$, $H = \{P(n), n\leqslant k, n \text{ odd}\}$, $H' = \{P(n), n\leqslant k, n \text{ even}\}$. Consider $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$. Then $\mathscr{T}\sim\mathscr{T}'$, but $H\not\vdash H'$, and $H'\not\vdash H$.

Properties of sharpness equivalence will be studied more precisely in the next section. The following theorem shows that sharpness ordering is relevant with respect to validity and bias. Informally, validity is an increasing property, and lack of bias a decreasing one.

4.6. Monotony Theorem. *Let $\mathscr{C}$ be a testing context, and let $\mathscr{T}$ and $\mathscr{T}'$ be two batteries of tests for $\mathscr{C}$. Suppose $\mathscr{T}$ is logically (resp. formally) asymptotically sharper than $\mathscr{T}'$.*

If $\mathscr{T}'$ is logically (resp. formally) asymptotically valid, so is $\mathscr{T}$. If $\mathscr{T}$ is logically (resp. formally) unbiased, so is $\mathscr{T}'$.

Proof. We consider only the formal case (the logical one can be proved in a very similar way). Take $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$, $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$, $\mathscr{T}' = \langle H', (T'_n)_{n\in\mathbb{N}}\rangle$.

Assume that $\mathcal{T}'$ is valid, and examine $H \bigsqcup_n T_n$; $H \bigsqcup_n T_n \vdash H' \sqcup T'_i$ and this is true for all $i \in \mathbb{N}$; thus $H \bigsqcup_n T_n \vdash H' \sqcup_n T'_n$, and $H' \bigsqcup_n T'_n \vdash A$ by hypothesis. Thus $H \bigsqcup_n T_n \vdash A$.

Assume that $\mathcal{T}$ is unbiased and examine $H' \sqcup A$. Given any $n \in \mathbb{N}$, $H' \sqcup A \vdash H \sqcup A$, and $H \sqcup A \vdash T_n$ by hypothesis. So $H' \sqcup A \vdash T_n$; but $H' \sqcup A \vdash H$ obviously, and thus $H' \sqcup A \vdash H \sqcup T_n$, and this is true for all $n \in \mathbb{N}$. Thus $H' \sqcup A \vdash H \bigsqcup_n T_n$, and, as we have shown previously, $H \bigsqcup_n T_n \vdash H' \bigsqcup_n T'_n$. Thus $H' \sqcup A \vdash H' \bigsqcup_n T'_n$, and, as a consequence $H' \sqcup A \vdash T'_n$ for all $n \in \mathbb{N}$. □

4.7. (Counter-)examples. (1) Consider Example 4.5(4); $\mathcal{T}' \geqslant \mathcal{T}$, $\mathcal{T}$ is unbiased but $\mathcal{T}'$ is biased.

(2) Take $L = \{P\}$, $S = \mathbb{N}$, $A = \{\forall x\, P(x)\}$, $H = \emptyset$, $H' = \{P(0) \wedge \cdots \wedge P(k) \to \forall x\, P(x)\}$ and $T_n = \{P(0) \wedge \cdots \wedge P(n)\}$. Consider $\mathcal{T} = \langle H, (T_n)_{n \in \mathbb{N}} \rangle$ and $\mathcal{T}' = \langle H', (T_n)_{n \in \mathbb{N}} \rangle$. Then $\mathcal{T}' \geqslant \mathcal{T}$, $\mathcal{T}'$ is valid and $\mathcal{T}$ is not.

4.3. Asymptotic equivalence

We now study the properties of the equivalences induced by the asymptotic sharpness. In fact, those preorders, especially the formal one, are rather poor, and are mainly valuable because of their induced equivalences. The following theorem makes precise the relationships between sharpness equivalence and acceptability.

4.8. Stability Theorem. *Let $\mathscr{C}$ be a testing context, and $\mathcal{T}$, $\mathcal{T}'$ be two batteries of tests for this context.*

If $\mathcal{T}$ is logically (resp. formally) acceptable and $\mathcal{T}'$ is logically (resp. formally) asymptotically equivalent to $\mathcal{T}$, then $\mathcal{T}'$ is logically (resp. formally) acceptable.

In other words, acceptability is preserved by (asymptotic) equivalence. An interesting fact is that, in practical cases, two acceptable batteries of tests are actually equivalent.

4.9. Equivalence Theorem. *Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A \rangle$ be a testing context, and $\mathcal{T} = \langle H, (T_n)_{n \in \mathbb{N}} \rangle$ and $\mathcal{T}' = \langle H', (T'_n)_{n \in \mathbb{N}} \rangle$ be two batteries of tests for $\mathscr{C}$. Suppose that there exists a **finite sub-theory** A_0 of A such that $A_0 \sqcup H \vdash\!\dashv A_0 \sqcup H'$ (finiteness condition).*

Then, if $\mathcal{T}$ and $\mathcal{T}'$ are formally *acceptable they are formally asymptotically equivalent.*

Proof. Notice at first that the finiteness condition implies that

$$A \sqcup H \vdash\!\dashv A_0 \sqcup H \vdash\!\dashv A_0 \sqcup H' \vdash\!\dashv A \sqcup H'.$$

By acceptability, $H \bigsqcup_n T_n \vdash\!\dashv H \sqcup A$ and $H' \bigsqcup_n T'_n \vdash\!\dashv H' \sqcup A$. Then we have

$$H \bigsqcup_n T_n \vdash H \sqcup A \vdash H \sqcup A_0 \vdash A_0.$$

By compactness we get j such that $H \sqcup T_j \vdash A_0$. Then

$$H \sqcup T_j \vdash H \sqcup A_0 \vdash H' \sqcup A_0 \vdash H' \bigsqcup_n T'_n.$$

Thus for all $i \in \mathbb{N}$, there exists $j \in \mathbb{N}$ such that $H \sqcup T_j \vdash H' \sqcup T'_i$. From the first remark, $H' \sqcup A \vdash H \sqcup A$. Thus $\mathcal{T} \geqslant \mathcal{T}'$, and, by symmetry, $\mathcal{T} \sim \mathcal{T}'$. □

Intuitively, the finiteness condition means that $\mathcal{T}$ and $\mathcal{T}'$ are not too different from one another. The theorem then states that two acceptable batteries of tests are either equivalent, or *very* different.

Consider such an example. Take $L = \{P\}$, $S = \mathbb{N}$, $A = \{P(n), n \in \mathbb{N}\}$ (A must of course be infinite!). Let $\mathcal{T} = \langle H, (T_n)_{n \in \mathbb{N}} \rangle$ with

$$H = \emptyset, \qquad T_n = \{P(0) \wedge \cdots \wedge P(n)\},$$

and $\mathcal{T}' = \langle H', (T'_n)_{n \in \mathbb{N}} \rangle$, with $H' = A$, $T'_n = \emptyset$. Then $\mathcal{T}$ and $\mathcal{T}'$ are formally acceptable, $\mathcal{T}' \geqslant \mathcal{T}$ but not $\mathcal{T} \geqslant \mathcal{T}'$. Notice that $H \sqcup A \vdash\!\dashv H' \sqcup A$. It can be seen that no *finite* $A_0 \subseteq A$ such that $H \sqcup A_0 \vdash\!\dashv H' \sqcup A_0$ exists.

Another example follows. Take $S = \mathbb{N}$, $L = \{P\}$, $A = \{P(n), n \in \mathbb{N}\}$, $T_n = T'_n = \{P(0) \wedge \cdots \wedge P(n)\}$; take $H = \{P(0) \wedge \cdots \wedge P(k) \to \forall x\, P(x)\}$ and $H' = \{P(0) \wedge \cdots \wedge P(k) \to P(n), n \in \mathbb{N}\}$. However, in most practical cases the finiteness condition holds.

Let $\mathscr{C}$ be a testing context, and $\mathcal{T}$ a battery of tests for $\mathscr{C}$. $\mathcal{T}$ is (formally) finitely acceptable if $\mathcal{T}$ is formally acceptable and (formally) finitely valid (cf. Section 3.2).

4.10. Lemma. *Let $\mathscr{C}$ be a testing context, and let $\mathcal{T}$ and $\mathcal{T}'$ be two batteries of tests for $\mathscr{C}$. Suppose that $\mathcal{T}$ and $\mathcal{T}'$ are (formally) finitely acceptable and are comparable with the formal asymptotic sharpness preorder.*

Then the finiteness condition holds.

Proof. We use the above notation.

Let us examine $\mathcal{T}$. By finite acceptability we get k such that $H \sqcup T_k \vdash A$. By acceptability then,

$$H \bigsqcup_n T_n \vdash\!\dashv H \sqcup A \vdash\!\dashv H \sqcup T_k.$$

By compactness we get a finite subtheory of A, say B, such that

$$H \bigsqcup_n T_n \vdash\!\dashv H \sqcup A \vdash\!\dashv H \sqcup T_k \vdash\!\dashv H \sqcup B.$$

With $\mathcal{T}'$, we get in the same way k' and B'. Suppose now $\mathcal{T} \geqslant \mathcal{T}'$. From the first condition, it can be easily shown that $H \bigsqcup_n T_n \vdash H' \bigsqcup_n T'_n$ and thus $H \sqcup B \vdash H' \sqcup B'$. From the second one we get $H' \sqcup B' \vdash H \sqcup B$. Take now $A_0 = B \sqcup B'$, which is actually a finite subtheory of A: $H \sqcup A_0 \vdash\!\dashv H' \sqcup A_0$. □

4.11. Corollary. *Let $\mathcal{T}$ and $\mathcal{T}'$ be two (formally) finitely acceptable batteries of tests for a testing context $\mathscr{C}$.*

If $\mathcal{T}$ are $\mathcal{T}'$ are comparable with the formal asymptotic sharpness preorder, then they are equivalent.

In practice, the theory being tested is often finite, and by the Compactness Theorem 3.5, Theorem 4.9 applies.

Let us conclude this section by showing that the potential dual statement of the Equivalence Theorem 4.9 is not true in general. For this purpose, take $S=\mathbb{N}$, $L=\{P, Q\}$, $A=\{\forall x\, P(x)\}$; let $(\mathscr{S})$ be the family of those $L(S)$-structures that validate $H=\{(\forall x\, Q(x))\leftrightarrow(\forall x\, P(x))\}$, and such that P and Q have calculable meanings; take

$$T_n=\{P(0)\wedge\cdots\wedge P(n)\},\qquad T'_n=\{Q(0)\wedge\cdots\wedge Q(n)\}.$$

Then $\mathscr{T}=\langle H,(T_n)_{n\in\mathbb{N}}\rangle$ and $\mathscr{T}'=\langle H',(T'_n)_{n\in\mathbb{N}}\rangle$ are both logically acceptable, but not comparable.

5. Construction, optimization and quality assessment

Previous sections were devoted to studying, from a theoretical viewpoint, the properties of testing contexts and batteries of tests. They have shown that the behaviour of those abstract objects may be quite directly related to the common intuition about program testing.

This section describes the application of this theoretical model to three important practical problems: the effective construction of a suitable battery of tests for a given problem, the optimization of a given battery of tests according to some given criterion, and the global quality assessment of a testing process.

5.1. Towards an effective construction method

As shown by the testing process diagram (see Section 1.4) the construction phase consists of starting with a testing context $\mathscr{C}_0$, and defining a new testing context $\mathscr{C}_1$, which is a conservative restriction of $\mathscr{C}_0$, together with a formally acceptable battery of tests $\mathscr{T}$ for $\mathscr{C}_1$:

$$\mathscr{C}_0 \xrightarrow[\text{construction}]{} \mathscr{T} \text{ acceptable for } \mathscr{C}_1,\ \mathscr{C}_1\subseteq\mathscr{C}_0.$$

In practice, we must only deal with 'likely' conservative restrictions. *The new hypotheses we postulate* (regularity or uniformity hypotheses) *must be sensible with respect to the concrete problem.*

Consider the following example. Let $\mathscr{C}_0=\langle L_0, S, (\mathscr{S}_0), A\rangle$ be an initial context with formal hypotheses H_0. $\mathscr{C}_0$ is an H_0-context. Suppose that A is $\{\forall x\,\phi(x)\}$ where ϕ is an $L_0(S)$-formula, without any quantifier, and without any variable but x, such that any non-logical symbol of ϕ has a calculable meaning for every structure of $(\mathscr{S}_0)$. Suppose that S is countable, say $S=\mathbb{N}$. In order to construct $\mathscr{T}$ and $\mathscr{C}_1$, we may use two kinds of hypotheses, depending on the practical context.

(1) Take $H_1=\{\phi(0)\wedge\cdots\wedge\phi(k)\rightarrow\forall x\,\phi(x)\}$, restrict $\mathscr{C}_0$ with H_1 to get an $H_0\sqcup H_1$-context $\mathscr{C}_1$ (notice that $H_0\sqcup A\vdash H_1$ for any choice of k!).

Then $\mathscr{T}=\langle H_0\sqcup H_1,(T_n)_{n\in\mathbb{N}}\rangle$ with $T_n=\{\phi(0),\ldots,\phi(n)\}$ solves the problem. Note that the choice of k does not need to be explicit (exhaustive testing strategy).

(2) Add a new constant a to L_0, and consider $H_1 = \{\phi(a) \to \forall x\, \phi(x)\}$; restrict $\mathscr{C}_0$ to get $\mathscr{C}_1$ as above. Again, $H_0 \sqcup A \vdash H_1$. Now $T_n = \{\phi(a)\}$ solves the problem (random sampling testing).

We thus know how to effectively construct $\mathscr{T}$ and $\mathscr{C}_1$ for the basic case $A = \{\forall x\, \phi x\}$ in many practical problems (other special sensible hypotheses could however be used). We now show that the syntactical form of A may be modified.

5.1. Construction Lemma. *Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ and $\mathscr{T}' = \langle L, S, (\mathscr{S}), A'\rangle$ be two testing contexts with the same family of $L(S)$-structures $(\mathscr{S})$. Let $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ be a battery of tests for $\mathscr{C}$. Assume that $A \sqcup H \dashv\vdash A' \sqcup H$.*

Then $\mathscr{T}$ is an acceptable battery of tests for $\mathscr{C}'$ if and only if it is so for $\mathscr{C}$.

Proof. $\mathscr{T}$ is obviously a battery of tests for $\mathscr{C}'$ (this does not depend on A).

$\mathscr{T}$ is valid for $\mathscr{C}'$: $H \bigsqcup_n T_n \vdash A$ (validity for $\mathscr{C}$), $H \sqcup_n T_n \vdash H$, $H \bigsqcup_n T_n \vdash H \sqcup A \vdash H \sqcup A'$ and thus $H \bigsqcup_n T_n \vdash A'$.

$\mathscr{T}$ is unbiased for $\mathscr{C}'$: for any n, $H \sqcup A \vdash T_n$ (lack of bias for $\mathscr{C}$), and $H \sqcup A' \vdash H \sqcup A \vdash T_n$.

The converse is proved by symmetry. □

Now, the following Decomposition Theorem transforms the problem of constructing a battery of tests for a complex theory A into (possibly) infinitely many basic problems by splitting A into pieces.

5.2. Decomposition Theorem. *Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A\rangle$ be a testing context, with A a **countable** union $A = \bigsqcup_{i\in\mathbb{N}} A^{(i)}$.*

For each $i \in \mathbb{N}$, let $\mathscr{C}^{(i)}$ be the testing context $\langle L, S, (\mathscr{S}), A^{(i)}\rangle$, and let $\mathscr{T}^{(i)} = \langle H^{(i)}, (T_n^{(i)})_{n\in\mathbb{N}}\rangle$ be a battery of tests for $\mathscr{C}^{(i)}$. Let $H = \bigsqcup_{i\in\mathbb{N}} H^{(i)}$,

$$T_n = \bigsqcup_{i\le n} T_n^{(i)} = T_n^{(0)} \sqcup \cdots \sqcup T_n^{(n)}.$$

Then $\mathscr{T} = \langle H, (T_n)_{n\in\mathbb{N}}\rangle$ is a battery of tests for $\mathscr{C}$. If each $\mathscr{T}^{(i)}$ is logically acceptable for $\mathscr{C}^{(i)}$, so is $\mathscr{T}$ for $\mathscr{C}$. If each $\mathscr{T}^{(i)}$ is formally acceptable for $\mathscr{C}$, so is $\mathscr{T}$ for $\mathscr{C}$.

Proof. (1) *$\mathscr{T}$ is a battery of tests for $\mathscr{C}$.* Fix n. Each $T_n^{(i)}$ being a test for $\mathscr{C}^{(i)}$ and T_n being the union of finitely many T_n^i, T_n is actually a test for $\mathscr{C}$ ($\mathscr{C}$ and $\mathscr{C}^{(i)}$ are built on the *same* $(\mathscr{S})$!).

Each $\mathscr{C}^{(i)}$ is a testing context. For each $\mathscr{S} \in (\mathscr{S})$, $\mathscr{S} \models H^{(i)}$, for each index i; so $\mathscr{S} \models \bigsqcup_i H^{(i)}$ and $\mathscr{S} \models H$. For the same reason, for each index i, $H^{(i)} \sqcup T_{n+1}^{(i)} \vdash T_n^{(i)}$ for each n. Thus

$$\bigsqcup_{i\le n+1} (H^{(i)} \sqcup T_{n+1}^{(i)}) \vdash \bigsqcup_{i\le n+1} T_n^{(i)},$$

$$\left(\left(\bigsqcup_{i\le n+1} H^{(i)}\right) \sqcup T_{n+1}\right) \vdash (T_n^{(n+1)} \sqcup T_n),$$

$$(H \sqcup T_{n+1}) \vdash T_n^{(n+1)} \sqcup T_n,$$

$H \sqcup T_{n+1} \vdash T_n$, which holds for all n.

We now focus on the formal case; the logical one can be handled in the same way.

(2) *$\mathcal{T}$ is valid for $\mathscr{C}$.* Consider $H \bigsqcup_n T_n$. This can be rewritten as

$$\bigsqcup_i H^{(i)} \bigsqcup_n \left(\bigsqcup_{i \leqslant n} T_n^{(i)} \right),$$

$$\bigsqcup_i \left(H^{(i)} \bigsqcup_{n \geqslant i} T_n^{(i)} \right). \qquad (*)$$

But, by projective reliability, we have

$$H^{(i)} \sqcup T_i^{(i)} \vdash \bigsqcup_{n \leqslant i} T_n^{(i)} \quad \text{for all } i.$$

So (*) proves actually

$$\bigsqcup_i (H^{(i)} \bigsqcup_n T_n^{(i)}),$$

and by the validity of $\mathcal{T}^{(i)}$ for $\mathscr{C}^{(i)}$, this proves $\bigsqcup_i A^{(i)}$, that is A.

(3) *$\mathcal{T}$ is unbiased for $\mathscr{C}$.* Consider $H \sqcup A$. This may be rewritten

$$\bigsqcup_i (H^{(i)} \sqcup A^{(i)}).$$

By lack of bias of $\mathcal{T}^{(i)}$ for $\mathscr{C}^{(i)}$, this proves, for any n, $\bigsqcup_i T_n^{(i)}$ and, as a special case, $\bigsqcup_{i \leqslant n} T_n^{(i)}$, that is T_n. □

Observe that the diagonalization method used to build T_n does not work if there are *uncountably* many $A^{(i)}$.

Given the Construction Lemma 5.1, and the Decomposition Theorem 5.2, we can now describe an effective and general method to achieve the construction phase.

Remember firstly that, following the discussion in Section 3, we only have to deal with purely universal 'calculable' formulas. Consider therefore an initial context $\mathscr{C} = \langle L, S, (\mathcal{S}), A \rangle$, with $A = \{\forall x\, \forall y\, \phi(x, y)\}$ with ϕ a formula as previously. Suppose $S = \mathbb{N}$.

(1) Firstly, restrict $\mathscr{C}$ with a uniformity hypothesis H_1 (or some other suitable likely hypothesis) for the formula $\forall x[\forall y\, \phi(x, y)]$. We then obtain $\mathscr{C}_1 = \langle L, S, (\mathcal{S}_1), A \rangle$ and $H_1 = \{\forall y\, \phi(0, y) \wedge \cdots \wedge \forall y\, \phi(k, y) \rightarrow \forall x\, \forall y\, \phi(x, y)\}$. Under H_1, A is now equivalent to

$$A_1 = \{\forall y\, \phi(i, y),\ i \in \mathbb{N}\}.$$

(2) By the Construction Lemma 5.1, it suffices now to consider the context $\mathscr{C}_2$ obtained by replacing A with A_1 in $\mathscr{C}_1$. $\mathscr{C}_2 = \langle L, S, (\mathcal{S}_1), A_1 \rangle$.

(3) The Decomposition Theorem 5.2 may now be applied. It suffices to consider the (infinitely many) contexts $\mathscr{C}_2^{(i)} = \langle L, S, (\mathcal{S}_1), A_1^{(i)} \rangle$, with

$$A_1^{(i)} = \{\forall y\, \phi(i, y)\}.$$

(4) The example described earlier may now be applied. Notice that the Decomposition Theorem 5.2 can be applied at point (3) precisely because S is countable. With an uncountable S, we might not have been able to conclude anything. The method can obviously be extended to a more complex A. The following definition states some *sufficient* practical conditions on the testing contexts that can be handled in this way.

5.3. Definition. Let $\mathscr{C} = \langle L, S, (\mathscr{S}), A \rangle$ be a testing context. $\mathscr{C}$ is *testable* if S is at most countable, and A is a theory with at most countably many non-logical axioms (countable theory), each of them being a purely universal formula each of whose non-logical symbols has a calculable meaning, whatever structure of $(\mathscr{S})$ is considered.

Notice that if L has at most countably many non-logical symbols (countable language), one can always (by the Construction Lemma 5.1) reduce A to a countable theory ($L(S)$ being countable).

5.2. Construction vs. optimization

We can effectively construct an acceptable battery of tests for a (possibly restricted) given testing context. However, it is often the case that this battery of tests is not satisfactory, for some practical and/or theoretical reasons. We are thus naturally led to the optimization problem with respect to some given criterion.

Consider firstly the formal sharpness criterion. Given a context $\mathscr{C}$ and a battery of test $\mathscr{T}$, which is acceptable for $\mathscr{C}$, we try to optimize $\mathscr{T}$ to get a strictly sharper acceptable battery of tests. The Equivalence Theorem 4.9 applies (at least if the theory to be tested is finite—this is actually often the case). It states that any other acceptable battery of tests is in fact equivalent to $\mathscr{T}$, and thus can be easily characterized.

We must therefore turn to some *more precise* criterion which results from some external considerations. Those considerations are not, in general, expressible in our formalism. They often deal with testing cost or profitability. Nevertheless, we can, to a certain extent, provide a method for assessing their legitimacy.

The idea is to consider the likelihood, the legitimacy, of the hypotheses which are involved in sharpness equivalence. Consider the H_0-initial context $\mathscr{C}_0 = \langle L, S, (\mathscr{S}_0), A \rangle$. H_0 can be considered as the set of fundamental, *initial hypotheses* about the implementation, *whose validity may not be in doubt.*

Now, suppose we have already constructed (probably using the above method) an $H_0 \sqcup H_1$-context $\mathscr{C}_1 = \langle L, S, (\mathscr{S}_1), A \rangle$, which is a conservative restriction of $\mathscr{C}_0$ (in most cases, $H_0 \sqcup A \vdash H_1$), and an acceptable battery of tests $\mathscr{T} = \langle H_0 \sqcup H_1, (T_n)_{n \in \mathbb{N}} \rangle$ for $\mathscr{C}_1$. H_1 is a set of technical *construction hypotheses*, probably uniformity and regularity hypotheses, which have been *postulated* about the implementation for theoretical reasons. They should have been carefully chosen as likely hypotheses, but in any case they may be considered as *less legitimate than the initial hypotheses.*

Consider now a battery of tests $\mathscr{T}' = \langle H_0 \sqcup H_1, (T'_n)_{n \in \mathbb{N}} \rangle$, which is an optimization of $\mathscr{T}$ according to some external and inexpressible extra criterion. $\mathscr{T}'$ is equivalent to $\mathscr{T}$, and we thus have

$$H_0 \sqcup H_1 \bigsqcup_n T_n \vdash H_0 \sqcup H_1 \bigsqcup_n T'_n.$$

This means that, given the initial hypotheses and the construction hypotheses, $\bigsqcup_n T_n$ and $\bigsqcup_n T'_n$ are equivalent. We say that this optimization is of *level* 0 if equivalence still holds under the initial hypotheses. Otherwise, it is said to be of *level* 1.

A level 0 optimization can be considered as perfectly safe and legitimate. It can be seen that a typical optimization of level 0 is to increase the redundancy of the battery of tests, by adding some new experiments, already implied by the old ones, to the tests. Such an optimization will probably increase the cost of the testing process, but also the quality of the conclusion.

A level 1 optimization must be considered not to be as safe as a level 0 one. A typical example is *to decrease the redundancy of the battery of tests*, and thus the cost of the testing process. Consider for example a theory $A = \{\forall x\, \phi(x)\}$ to be tested under a construction hypothesis $H_1 = \{\exists x\, \phi(x) \to \forall x\, \phi(x)\}$; a test $\{\phi(a), \phi(b)\}$ will be, at level 1, optimized into $\{\phi(a)\}$.

These two kinds of optimization are quite usual in practice and are often left implicit. In fact, the word 'optimization' tends to be used when some extra hypotheses are involved, leading to some simplifications. In our formalism, we can express this by restricting the context once more with a new hypotheses theory H_2:

$$H_0\text{-initial context} \xrightarrow[\substack{\text{construction} \\ H_1}]{} H_0 \sqcup H_1\text{-context } \mathscr{C}_1$$

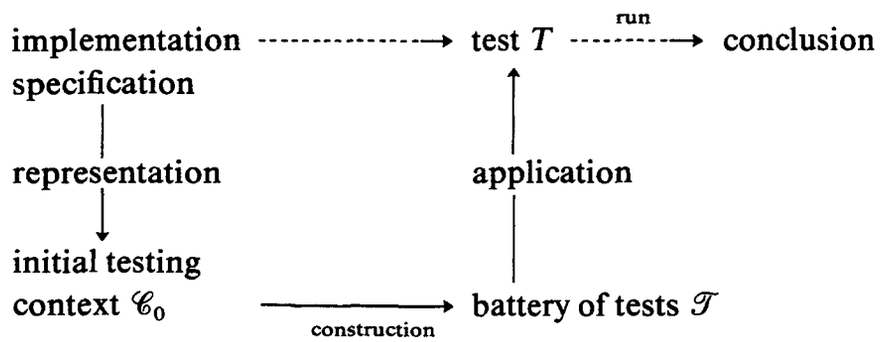
$$\xrightarrow[\substack{\text{optimization} \\ H_2}]{} H_0 \sqcup H_1 \sqcup H_2\text{-context } \mathscr{C}_2$$

(remember that conservative restriction is transitive). H_2 is called the *optimization hypotheses* theory. Such an optimization is said to be of *level* 2. Such extra hypotheses must be considered a priori to be illegitimate and not useful. Their validity should be examined very carefully, as should the profitability of such an optimization. An optimization of level 2 leads in general to some considerable alteration of the battery of tests.

5.3. Quality assessment of a testing process

It is now necessary to face the question of assessing the quality of a testing process. Even though we are not able at present to answer this question in a satisfactory fashion we will give here some insight into this very difficult but fundamental problem.

We can now proceed with the discussion of the testing process diagram first sketched in Section 1.4:



From the implementation features and the specification to be tested, we get an abstract version of the problem as an H_0-testing context $\mathscr{C}_0 = \langle L_0, S, (\mathscr{S}_0), A\rangle$. We then construct an acceptable battery of tests $\mathscr{T} = \langle H_0 \sqcup H_1, (T_n)_{n\in\mathbb{N}}\rangle$ for a (possibly) conservatively restricted $H_0 \sqcup H_1$-testing context $\mathscr{C}_1 = \langle L_1, S, (\mathscr{S}_1), A\rangle$. In the application phase, we choose a test T_p from $(T_n)_{n\in\mathbb{N}}$ and run it. If T_p is passed, the implementation is declared to be *correct* with respect to the given specification; if T_p fails, it is declared to be *incorrect.*

We suppose that the representation phase is *consistent* in that the concrete problem is *intuitively* 'correctly' translated into an abstract formulation. As a special case, we suppose that there is an (unknown) structure $\mathscr{S}_r$ which *actually* represents the implementation functional behaviour.

Suppose now that *the selected test T_p fails.* Suppose that the implementation is nevertheless correct. Then $\mathscr{S}_r \models A$. Because of the conservative restriction, there exists $\mathscr{S}'_r \in (\mathscr{S}_1)$ such that $\mathscr{S}'_r|_{L_0(S)} = \mathscr{S}_r$. Then by an earlier remark, $\mathscr{S}'_r \models A|_{L_1(S)}$ and, by hypothesis, $\mathscr{S}'_r \models (H_0 \sqcup H_1)$. But, by lack of bias, $A|_{L_1(S)} \sqcup H_0 \sqcup H_1 \vdash T_p$ (in fact for any p!). So $\mathscr{S}'_r \models T_p$ and we get a contradiction. We then can conclude that *the implementation is certainly incorrect,* whatever the index p chosen. This should be related to the discussion of Section 1.1 about testing anisotropy.

The above proof shows that if the implementation is correct, then all the tests T_n will be passed. Conversely, suppose that *the selected test T_p is passed,* and that the implementation is not correct. In general we cannot conclude anything. We need some extra hypotheses, called *coupling hypotheses.*

First coupling hypothesis: Everything happens exactly as if $\mathscr{S}_r$ is conserved by the restriction: one can find an $\mathscr{S}'_r \in (\mathscr{S}_1)$ such that $\mathscr{S}'_r|_{L_0(S)}$ behaves like $\mathscr{S}_r$.

Second coupling hypothesis: Everything happens exactly as if all the tests T_n, $n \in \mathbb{N}$ were passed—instead of T_p only.

Assume these coupling hypotheses. Because of the validity of $\mathscr{T}$, $H_0 \sqcup H_1 \bigsqcup_n T_n \vdash A$. Thus $\mathscr{S}'_r \models A$ (in fact $A|_{L_1(S)}$) and, by an earlier remark, $\mathscr{S}_r \models A$, which leads to a contradiction. The implementation is thus correct.

We have shown that the quality of the conclusion which our testing process leads to is actually related to (and only to) these two hypotheses. We may thus identify

them and define the quality of a testing process to be the validity level of those hypotheses.

It should be emphasized that these two hypotheses are essentially of the same nature. $(\mathscr{S}_1)$ may be viewed as a sampling of $(\mathscr{S}_0)$, the fairness of which is determined by the likelihood of the postulated construction hypotheses H_1. On the other hand, if T_p is passed, then, by reliability, all T_n with $n \leqslant p$ would have been passed. $(T_n)_{n \leqslant p}$ may also be viewed as a sampling of $(T_n)_{n \in \mathbb{N}}$, the fairness of which is determined by the index p. Thus, in both cases, we state that some results which are formally true on a fair sampling of a domain may be considered to be true for any object of the domain. This is no more than a special kind of regularity or uniformity hypothesis! Such kinds of coupling hypotheses have been used extensively in mutation testing [5].

Thus we can now (re-)define testing process quality as the fairness of those samplings. Remember that restricting the context from $\mathscr{C}_0$ to $\mathscr{C}_1$ may be regarded as increasing one's knowledge (in fact, by postulating some likely knowledge) about the implementation. On the other hand, the criterion for choosing T_p out of $(T_n)_{n \in \mathbb{N}}$ is mainly running cost. Testing process quality is thus directly related, through fairness, to

(1) the quality of information about the tested implementation,

(2) test running costs.

It must be verified that perfect information and infinite cost actually lead to a perfectly safe testing process conclusion. It is sufficient to look at an incorrect implementation. The initial context $\mathscr{C}_0$ looks like $\langle L, S, \{\mathscr{S}_r\}, A\rangle$: the family of structures has collapsed to a singleton. Suppose, for the sake of clarity, that $A = \{\forall x\, P(x)\}$ and $S = \mathbb{N}$. There exists an integer k such that $\mathscr{S}_r \nvdash P(k)$, so that $\mathscr{S}_r \models \{P(0) \wedge \cdots \wedge P(k) \rightarrow \forall x\, P(x)\}$ (perfect information). We restrict the context $\mathscr{C}_0$ with this hypothesis to $\mathscr{C}_1$ and take $T_n = \{P(0) \wedge \cdots \wedge P(n)\}$. We get an acceptable battery of tests for $\mathscr{C}_1$, and we may pick out T_k (infinite cost), which obviously fails.

The above discussion shows that, under perfect conditions, testing an implementation *proves* its correctness. Proving appears here as a special case of testing, namely as an extrapolation of testing to infinity. In practice we are thus led to consider program testing as a finite but highly valuable approximation of program proving, which is itself, in most cases, quite untractable. This is the very specificity of testing to give a *continuous* graduation of correctness assessments instead of the correct/incorrect (often humanly undecidable) alternatives of proving.

6. Conclusion

We have presented an abstract formalism for the investigation of program testing. This formalism extends Goodenough and Gerhart's one. Its main goal is to put into evidence the complementarity between program testing and program proving.

A specificity of testing is to provide a continuous graduation of correctness assessments instead of the correct/incorrect alternatives of proving. In our formal-

ism, program testing appears as a particular case of program proving. To test a program, one first proves the acceptability of a battery of tests $\langle H, (T_n)_{n\in\mathbb{N}}\rangle$. Then one proves the validity of some test T_p by running (evaluating) it. Acceptability guarantees continuity: the higher p is, the closer testing and proving are. In other words, extrapolating testing to infinity yields actually proving.

Testing is also essentially an incremental process. To test a program is to increase one's confidence and one's knowledge about it. Earlier theories of testing concentrated on the notion of a test criterion. We rather consider the testing process as a whole. This allows us to give account of such dynamic features. Incremental test construction and optimization can then easily be modelled. Also, we can localize precisely the 'likely' feature of testing, and model other actions as formal manipulations of well-defined mathematical objects.

Our theory of testing shows that all specifications are not naturally testable. They should at least be purely universally quantified. Algebraic data type specifications, for entirely different reasons, are precisely of that kind. Moreover, the concept of hierarchical (algebraic) specification leads naturally to uniformity hypotheses for lower-level sorts and regularity hypotheses for the sort of interest. Assuming that algebras are finitely generated with respect to hierarchy allows to design the battery of tests at an abstract level of denotation. The size of the denotation of a data can be considered as a measure of its computational complexity. Our theory provides therefore a testing strategy for hierarchical algebraic specifications. A semi-automatic tool for testing algebraic specification is currently under development on this basis (see [4] for details and experimental results). PROLOG is used to generate specifically terms satisfying some constraints. Observe that PROLOG specifications are also purely universal. The incremental resolution strategy of PROLOG yields regularity hypotheses. Also, solutions involving free variables correspond precisely to uniformity subdomains. Purely universally quantified specifications, but the use of regularity and uniformity hypotheses, appear therefore as a basic requirement for testability.

Acknowledgment

This paper is mainly based on the author's doctoral thesis [2] which was submitted to the following jury: B. Robinet, chairman, M.C. Gaudel, M. Nivat, J.P. Jouannaud, H. Gallaire, J.C. Rault and K. Culik, II. I owe a special debt to M.C. Gaudel for her help.

A previous version of this paper [3] was written while the author was visiting DAIMI at Aarhus University, Denmark, supported by an INRIA grant.

References

[1] W.R. Adrion, M.A. Branstad and J.C. Cherniavsky, Validation, verification and testing of computer software, *ACM Comput. Surveys* **14** (2) (1982) 159–192.

[2] L. Bougé, Modélisation de la notion de test de programmes; application à la production de jeux de test, Thèse de 3ème cycle, Université Paris 6, 1982.
[3] L. Bougé, A proposition for a theory of testing: an abstract approach to the testing process, Rept. No. PB-160, DAIMI, Aarhus University, Denmark, 1983.
[4] L. Bougé, N. Choquet, L. Fribourg and M.-C. Gaudel, Application of PROLOG to test sets generation from algebraic specifications, *Proc. 2nd Internat. Joint Conf. on Theory and Practice of Software Development*, Lecture Notes in Computer Science **186** (Springer, Berlin, 1985) 261–275.
[5] T.A. Budd, Mutation analysis: ideas, examples, problems and prospects, in: B. Chandrasekaran and S. Raddichi, eds., *Computer Program Testing* (North-Holland, Amsterdam, 1981) 129–148.
[6] M.S. Deutsch, *Software Verification and Validation*; *Realistic Project Approaches* (Prentice-Hall, Englewood Cliffs, NJ, 1982).
[7] M. Geller, Test data as an aid in proving program correctness, *Comm. ACM* **21** (5) (1978).
[8] S.L. Gerhart, Program validation, in: T. Anderson and B. Randers, eds., *Computing Systems Reliability* (Cambridge Univ. Press, 1979) 66–108.
[9] J.B. Goodenough and S.L. Gerhart, Toward a theory of test data selection, *SIGPLAN Notices* **10** (6) (1975) 495–510.
10] J.S. Gourlay, A mathematical framework for the investigation of testing, *IEEE Trans. Software Engrg.* **SE-9** (21) (1983) 686–709.
11] W.E. Howden, Completeness criteria for testing elementary program functions, *Proc. 5th Internat. Conf. Software Engrg.*, San Diego, CA (1981) 235–243.
12] E. Miller and W.E. Howden, Tutorial: Software testing and validation techniques, IEEE Catalog No. EH0 138–8, 1978.
13] J.R. Shoenfield, *Mathematical Logic* (Addison-Wesley, Reading, MA, 1967).
14] E.J. Weyuker, The oracle assumption of program testing, *Proc. 13th Hawaii Internat. Conf. Syst. Sciences* 1 (1980) 44–49.
15] E.J. Weyuker and T.J. Ostrand, Theories of program testing and the application to revealing subdomains, *IEEE Trans. Software Engrg.* **SE-6** (3) (1980) 236–246.
16] L.J. White, E.I. Cohen and S.J. Zeil, A domain strategy for computer program testing, in: B. Chandrasekaran and S. Raddichi, eds., *Computer Program Testing* (North-Holland, Amsterdam, 1981) 103–113.