A formal approach to specification-based black-box testing*

María Victoria Cengarle
Institut für Informatik
Ludwig-Maximilians-Universität München
cengarle@informatik.uni-muenchen.de

Armando Martín Haeberer
Oblog Software S.A.
haeberer@oblog.pt

1 Introduction

This paper introduces an initial account of a formal methodology for specification-based black-box verification testing of software artefacts against their specifications, as well as for validation testing of specifications against the so-called *application concept* [14].

When testing software process artefacts we have three actors. The first is a *posit* we make on the real world, whether it be a *software artefact* reified from a specification or a *software artefact to be*, i.e., the application concept, the *hypothetical posit* we imagined and whose behaviour the specification should capture. In both cases we obtain *evidence* from such a posit—if it is a real software artefact by executing it; if it is a hypothetical posit by producing instances of its hypothetical behaviour. The second actor is the specification, which is a *theory* supposedly explaining the behaviour of the posit. Actually, when testing the relation between a posit and its specification what we test is the 'correctness' of such an explanation. In case the posit is a hypothetical one, we talk about *validation testing*, i.e., the testing activity aims at answering the question 'are we constructing the correct thing?'. In the case the posit is a software artefact, we talk about *verification testing* and the testing activity aims at answering the question 'are we constructing this thing correctly?'. Finally, the third actor is the property we are testing, which is a *hypothesis* we make about the posit and which should be tested using the whole specification as a background theory. In other words, this hypothesis will be true if

*The research reported in this paper was developed with the support of the DAAD (German Academic Exchange Service), the CNPq (Brazilian National Research Council), the Ludwig-Maximilians-Universität München, the EPSRC (Engineering and Physical Sciences Research Council, UK), the Imperial College of Science, Technology and Medicine, London, and PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro, Brazil).

(and hopefully 'only if') the specification correctly 'explains' the hypothetical posit (validation), or if the software artefact is 'correct' with respect to the specification (verification).

This setting resembles very closely the one of testing of *scientific theories*, i.e., of testing the 'correctness' of the explanation a particular scientific theory supports about certain *phenomena*. As soon as we investigate the relation between the two settings, its resemblance is compelling (see [12, 5]). The specification corresponds to the scientific theory, whilst the posit the specification describes corresponds to the phenomenon the scientific theory explains.

Let us denote by T^* the specification (or background theory),[1] H the hypothesis under test, and $\underline{E}$ the evidence produced by the posit. The problem of relating the evidence emerging from a phenomenon with the theory explaining it, i.e., the problem of logically explaining how some evidence, which is a piece of observation, can refute or confirm a hypothesis on the basis of a theory[2] explaining such a phenomenon (both stated in a theoretical language), was one of the major issues of the *Philosophy of Science* (*Epistemology*) of the Twentieth Century.

In Fig. **1** the two major strategies to relate theory and evidence are depicted. The most popular one is illustrated on the right-hand side of the figure. There, from the theory $T^* \cup \{H\}$ a prediction E_P about the evidence $\underline{E}$ is derived using the logic underlying both the so-called *theoretical* and *observational* segments of $T^* \cup \{H\}$. (We will succinctly discuss these segments below.) Then, the *experiment* consists in comparing the predicted evidence E_P with the one produced by the posit, i.e., $\underline{E}$. The experiment is *successful* if the prediction holds, *unsuccessful* otherwise. This strategy is called in the epistemological jargon the *hypothetico-deductive* strategy (in the sequel abbreviated to HD); after proposing a hypothesis about the posit, and in the presence of a background theory $T^* = T^*$, a prediction of an evidence is deduced from the two together and then compared with the actual evidence. As we show below, certain conditions should hold for this strategy to be sound.

The left-hand side of Fig. **1** pictures an alternative strategy, which is also intuitive. This strategy is called the *bootstrap* strategy in the epistemological jargon. Again, the purpose is to test a hypothesis H about a posit on the basis of a background theory, this time $T^* = (T^* \cup \{H\})^*$.[3] From an evidence $\underline{E}$ produced by the posit and by means of a set of functions $\{f_{T^*_x}\}$ derived from T^* (using its underlying logic), obtain a valuation α for the variables $\{x\}$ of H. Then, the *experiment* consists in determining if

[1] Theory presentations are denoted by T, T, etc.; theories obtained from those presentations (i.e., presentations closed under inference) are denoted by T^*, T^*, etc.

[2] In Epistemology literature it is usual to read "on the background of a theory" instead.

[3] By abuse of notation we also write $T^* = T^* \cup \{H\}$.

the valuation α makes H valid, in which case the experiment is *successful*, otherwise it is *unsuccessful*. As in the case of HD, certain conditions on the derivation of the set $\{f_{T^*_x}\}$ must hold for the strategy to be sound. The bootstrap strategy (as well as HD) is based on Carnap's ideas; in Glymour's words, *Whenever our evidence is stated in terms narrower than those of our theory, [one of Carnap's ideas] contains a stratagem for making the connection between evidence and theory: use [the background theory] to deduce, from the [evidence], instances of [the hypothesis].*

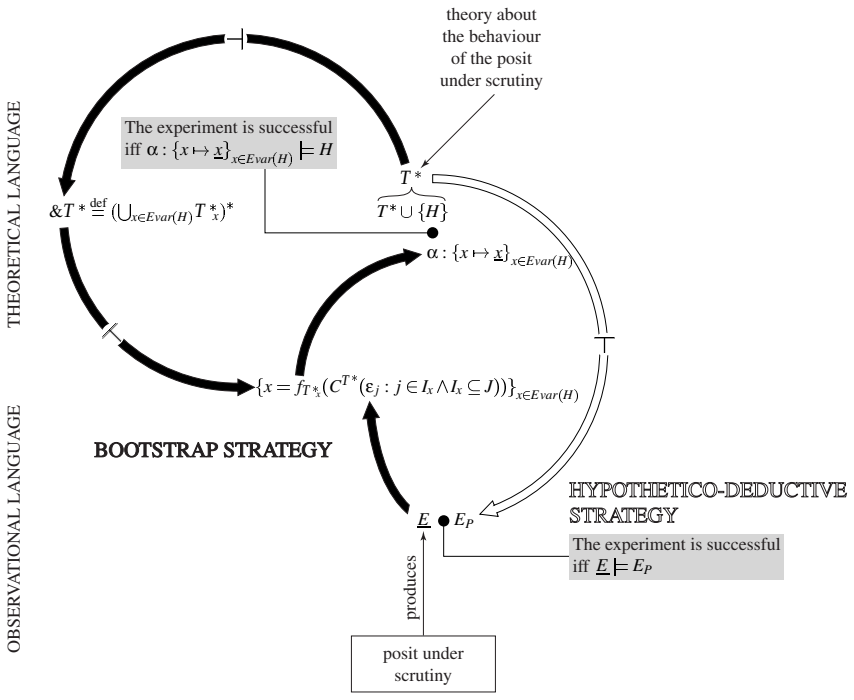

Figure 1: Testing strategies

According to which kind of testing we are performing, after applying any of the above strategies, *modus tollens* dictates the course to follow. In the case of verification testing of a software artefact against its specification, if the experiment was unsuccessful, then the software artefact must be revised. In the case of validation testing of a specification against a hypothetical posit, if the experiment was unsuccessful, then the specification must be revised. In both cases, if the experiment was successful, the only information we have is exactly that. Transforming this information into a confirmation means either performing infinitely many experiments or introducing some kind of *uniformity* hypothesis about the domain of the evidence enabling a finite partition of it into *uniform subdomains* such that it is sufficient to test one representative of each one of these subdomains. This paper will not deal with this last problem; its purpose is to introduce the bootstrap strategy as an appropriate one for specification-based black-box testing, and to show why HD is woefully inadequate.

In this paper we denote a theory by T^* to emphasise the fact that a theory is the closure of

a (usually finite, modulo axiom schemata) theory presentation (or axiomatisation) T^L by means of the inference rules of the underlying logic L. Notice that we have here another potentially infinite dimension for testing. Whatever strategy we use, we should test infinitely many hypotheses to cover the whole theory. However, if we have a finite theory presentation (modulo axiom schemata), we can use it instead of the whole theory, i.e., use its axioms as hypotheses.

Notice that evidence E is stated in a different language than theories and hypotheses. For expressing the former, a restricted language denoting observables and with a restricted logic suffices. For instance, the property 'this brick is red' denotes a directly observable fact, i.e., the redness of this particular brick. Moreover, the sentence 'all the bricks I am talking about are red' is a generalised (finite) conjunction of atomic sentences. In other words, the universal quantifier of the logic accompanying the language of observables must be finite, as we cannot observe infinitely many properties. The notion of direct observation can be relaxed to those things observed via 'accepted' instruments, a microscope when observing cells, or an *oracle* when observing software artefact behaviours.

In contrast, the language in which theories and hypotheses are stated must be rich enough to capture concepts, be they observable or not. Moreover, the accompanying logic must provide infinite quantifiers, modalities, etc. For instance, if the domain of the quantifiers were not infinite, a scientific theory would be transformed into an empirical generalisation.

The existence of these two sublanguages and their accompanying logics with different expressive and deductive power, is the root of the problem of testing alluded to above. In Glymour's words [11], *[...] how can evidence stated in one language confirm hypotheses stated in a language that outstrips the first? The hypotheses of the broader language cannot be confirmed by their instances, for the evidence, if framed in the narrower tongue, provides none. Consistency with the evidence is insufficient, for an infinity of incompatible hypotheses may obviously be consistent with the evidence [...].*

All these problems were deeply studied by the so-called *logical-empiricist philosophers*, in particular by the members of the *Vienna Circle*. Rudolf Carnap formally introduced the observational-theoretical dichotomy by means of a theory known today as *The Statement View of Scientific Theories* (in short The Statement View) in the 38 years between 1928 [2] and 1966 [3].

In the context of the Statement View, whenever we have an empirically interpreted theory T^* we have two disjoint subtheories of it, a theoretical one, whose presentation is $T_T = (\Sigma_T^T, Ax_T^T)$, and a purely observational one, whose presentation is $T_O = (\Sigma_O^T, Ax_O^T)$, related by a set of correspondence rules C^T, which provides the only empirical interpretation of T_T. Therefore, we have two languages

generated by the vocabulary of T^*, the theoretical one (which we will call L_T^T) and the observational one (which we will call L_O^T). Observable facts are stated in L_O^T . We will refer to such observable facts as *evidence*. The requirement that the set of correspondence rules provide the only empirical interpretation of the theoretical subtheory preserves the safeness of observational consequences. That is, some theoretical symbols are given empirical interpretation, by the correspondence rules, in terms of the observational symbols. The theoretical symbols whose meaning is not definitionally given by the correspondence rules are not interpreted further. Their properties are given by relating them with the already interpreted theoretical symbols by means of the deduction mechanisms provided by the underlying logic. This means that there are more theoretical symbols than observational ones. Therefore, it is easy to see that L_T^T outstrips L_O^T in expressive power. This difference is due, on the one hand, to the above mentioned 'difference in size' of the corresponding vocabularies, and on the other, to the logic underlying the theoretical and the observational segments of the theory.

Outline In Sect. 2 we present the hypothetico-deductive strategy in detail and discuss its flaws. In Sect. 3 we introduce in detail the bootstrap strategy for deterministic evidence and, in Sect. 4, we apply it to the verification testing of a toy deterministic program. In Sect. 5 the bootstrap strategy is adapted to non-deterministic evidence, whilst in Sect. 6 its application to the verification testing of a toy non-deterministic program is presented.

2 The hypothetico-deductive strategy

Let us recall Fig. **1**. In the HD strategy the hypothesis H and the theory T^* are used to derive a prediction E_P, and then some actual evidence $\underline{E}$ is used to determine whether or not the prediction is true, i.e., if $\underline{E} \models E_P$. The hypothesis H is stated in the theoretical language; the background theory T^* is an interpreted theory containing its theoretical part, its observational part, and its correspondence rules; finally, the prediction must obviously be stated in the observational language. In order to derive the prediction from the union of the theory and the hypothesis, the theoretical terms appearing in the latter must have a direct (by means of some correspondence rules) or indirect (by means of inference and some correspondence rules) empirical interpretation.

A HD schema is a triple $\langle T^*, H, E_P \rangle$ where T^*, H, and E_P are as in our scenario above, and the following three conditions hold:

(i) $T^* \cup \{H\}$ is consistent

(ii) $H, T^* \vdash E_P$

(iii) $T^* \nvdash E_P$

Then, as we said before, $\underline{E} \models E_P$ does not refute H w.r.t. T^*, whilst $\underline{E} \not\models E_P$ refutes H w.r.t. T^*.

Let us analyse these conditions. The first one is obvious; if $T^* \cup \{H\}$ were inconsistent, then any prediction could be derived from it. The second condition is the essence of the HD strategy. Finally, the third condition prevents us from affirming that H is tested in the case that T^* suffices for predicting E_P. Notice that we are using T^* instead of T^*: the reason is the necessity of stating condition (iii). Therefore, when using the HD strategy, we should consider $T^* \cup \{H\}$ as the theory explaining (making an appropriate prediction E_P) for $\underline{E}$, but H can only be confirmed w.r.t. some T^*, i.e., w.r.t. a subtheory of T^*.

The outstanding problems of the HD strategy are the following. First, $\underline{E}$ can never refute or confirm (for the precise meaning of confirmation recall the discussion of page 3 in the introduction) w.r.t. T^* any consequence of T^* itself; notice, however, that, if H is a consequence of T^*, then it can be confirmed with respect to $H \rightarrow T^*$, i.e., w.r.t. a subtheory of the original theory T^* such that conjoining it with the hypothesis is logically equivalent to T^* (see [10]). Second, if E_P is not a tautology, $\underline{E} \models E_P$, and L is any consistent sentence such that $E_P \nvdash L$, then L is confirmed by $\underline{E}$ w.r.t. a true theory (namely $L \rightarrow E_P$). Third, if H is confirmed by $\underline{E}$ w.r.t. T^*, then so is $H \wedge K$, where K is any sentence whatsoever that is consistent with H and T^*. (Recall the property known as *reinforcement of the antecedent* of classical propositional logic.) The first difficulty might be tolerated were it not for the other two; together, they make the HD account untenable.

There were different attempts to save the HD strategy. Let us consider, for instance, Merrill's attempt to overcome the third difficulty (see [15]). Additionally to the above listed three conditions, we should corroborate the fact that there do *not* exist sentences K, L, and M such that:

(iv) $\vdash H \leftrightarrow K \wedge L$ (viii) $L \nvdash H$ (viii) $\nvdash M \leftrightarrow L$

(v) $K \nvdash H$ (ix) $M \nvdash H$ (ix) $K, M, T^* \vdash E_P$

(x) $T^* \cup \{K, M\}$ is consistent

which in English means that if H is a conjunction, then no one of its conjuncts L (or any M equivalent to L) suffices for deriving E_P.

Unfortunately, Glymour showed (see [10]) that this addition leads to circular reasoning, i.e., $E_P \vdash H$. The proof can be sketched as follows. Assume that T^*, H, and E_P satisfy (i), (ii), and (iii) above.

Suppose that $\vdash (H \leftrightarrow (T^* \leftarrow E_P))$, then $(T^* \leftarrow E_P) \vdash H$ by the deduction theorem and, given that $E_P \vdash (T^* \leftarrow E_P)$, then $E_P \vdash H$. Suppose now that $\nvdash (H \leftrightarrow (T^* \leftarrow E_P))$, then we let K be $(T^* \leftarrow E_P)$, L be $((T^* \leftarrow E_P) \leftarrow H)$, and M be any tautology. In this case, and using the deduction theorem and modus ponens, as well as reasoning by *reductio ad absurdum*, it can be shown that K, L, and M satisfy (iv)–(x). Thus if E_P is a prediction by means of the hypothetico-deductive strategy, then necessarily $\vdash (H \leftrightarrow (T^* \leftarrow E_P))$ as in the first case above, which means that $E_P \vdash H$.

Moreover, Glymor has proved in [10] that another suggested attempt to save the HD strategy by adding additional constraints does not help. Even a late attempt to lend credibility to hypothetico-deductivism by using relevance logic (see [16]) instead of classical logic was not successful, since relevance logic itself was not yet well accepted.

Because of the failure of the HD strategy, specification-based black-box testing methods based on it (or on rudimentary versions of it), for instance the method proposed in [7, 8, 9] (for its criticism on the basis of this setting see [5]), have inherent and insurmountable problems. This problem led us to the consideration of the bootstrap strategy as an alternative basis for a methodology for specification-based black-box verification testing of software artefacts and for validation testing of specifications against the application concept.

3 The bootstrap strategy for deterministic evidence

We present here the bootstrap testing strategy for the case of theories over existential (in)equational logic EEQ^+ (called simply EEQ if inequalities are not involved), i.e., the logic underlying systems of n (in)equations with m unknowns. For the sake of simplicity, we assume that there is no α-conversion.

Let $\mathfrak{A} = \langle \boldsymbol{A}, \leq \rangle$ be an algebra with a natural order. Let T^* be a theory, let H be an equation (or inequality), and let both be mutually consistent. Let $E = \{\varepsilon_j : j \in J\}$ be a set of variables, and let $\underline{E} = \{\underline{\varepsilon_j} : \underline{\varepsilon_j} \in \boldsymbol{A} \text{ and } j \in J\}$ be a set of values for the variables in E (these variables and their values belong to the observational language), which once lifted by the set of correspondence rules C^{T^*} are consistent with H and T^*.

Bootstrap testing schemata are three place relations $\langle T^*, H, \underline{E} \rangle$, where T^* is a theory, H the hypothesis to be tested with respect to this theory, and $\underline{E}$ the evidence which can refute H with respect to T^*. In general, to be considered a bootstrap testing schema, $\langle T^*, H, \underline{E} \rangle$ must satisfy a set of conditions such as (i) to (iv) below. Of the bootstrap schemata we introduce only one. All of them coincide in

satisfying conditions (i), (ii) and (iv) below; the difference between them resides in the requirement stated by each one's condition (iii).

We begin by stating the Schema I for the deterministic case. In order to do so, we need to introduce some concepts and notation. A *subtheory* of a theory T^* is a theory ${T_1}^*$ such that $T^* \vdash {T_1}^*$; two theories are *equivalent* if each one is a subtheory of the other. A variable is *essential* to an (in)equation K if it occurs in every (in)equation equivalent to K; the set of essential variables of K is denoted by $Evar(K)$. (Recall that we do not have α-conversion.)

The Schema I of bootstrap testing is defined as follows. Given $\langle T^*, H, \underline{E}\rangle$, for each $x \in Evar(H)$ let ${T^*}_x$ be a subtheory of T^* such that:

(i) ${T^*}_x$ determines (the value of) x as a function of a set of variables indexed by I_x,
which is a subset of the evidence $E = \{\varepsilon_j : j \in J\}$,
denoted by $x = f_{{T^*}_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$.[4]

(ii) The set of values for the variables in $Evar(H)$
given by $\underline{x} = f_{{T^*}_x}(C^{T^*}(\underline{\varepsilon_j} : j \in I_x \wedge I_x \subseteq J))$
satisfies H.

(iii) There is no (in)equation K with $Evar(K) \subset Evar(H)$
such that $H, \&T^* \vdash K$ and $K, \&T^* \vdash H$.[5]

(iv) For all $x \in Evar(H)$, there is no (in)equation K with $Evar(K) \subset Evar({T^*}_x)$
such that $\vdash H \wedge {T^*}_x \leftrightarrow \{K\}$.

If conditions (i) to (iv) above are met, $\underline{E}$ is said to provide a *positive test* of H with respect to T^*.

The motivation for the conditions above is as follows:

Condition (i) The requirement that a value be determined for each quantity occurring essentially in H reflects a common prejudice against theories containing quantities that cannot be determined from the evidence. Given $\langle T^*, H, \underline{E}\rangle$, when values for the basic quantities occurring in H have not been determined from the evidence $\underline{E}$ using some theory T^*, then $\underline{E}$ and the relevant fragment of T^* do not of themselves provide reason to believe that those basic quantities are related as H claims them to be.

[4] We denote by $f_{{T^*}_x}$ the function (determined by subtheory ${T^*}_x$) which assigns a value to the essential variable x as a function of the $\{\varepsilon_j : j \in I_x \wedge I_x \subseteq J\}$ translated by the set of correspondence rules C^{T^*}.

[5] $\&T^* \stackrel{\text{def}}{=} (\bigcup_{x \in Evar(H)} {T^*}_x)^*$. Notice that if ${T^*}_x$ is presented by an axiomatisation T_x, then $\&T^* \stackrel{\text{def}}{=} (\bigcup_{x \in Evar(H)} T_x)^*$. In this latter case we denote by $\&T$ the axiomatisation $\bigcup_{x \in Evar(H)} T_x$.

Condition (ii) Obvious.

Condition (iii) Suppose there exists an (in)equation K such that $Evar(K) \subset Evar(H)$, $H, \&T \vdash K$ and $K, \&T \vdash H$. Let $y \in Evar(H)$, $y \notin Evar(K)$. This means that y is essential to H, but not to H in conjunction with $\&T$. In other words, y could take any value, independent of the evidence $\{\underline{\varepsilon_j} : j \in I_y \wedge I_y \subseteq J\}$. Therefore, the evidence $\underline{E} = \{\underline{\varepsilon_j} : \underline{\varepsilon_j} \in \boldsymbol{A} \text{ and } j \in J\}$ and the method $\{x = f_{T_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))\}_{x \in Evar(H)}$ of computing quantities in $Evar(H)$ from the evidence would fail to test the constraints H imposes on y. Thus, a significant component of what H says would go untested.

Condition (iv) Consider the theory presentation $T : \{x = y, c = d\}$ and the hypothesis $H : x = y$ with $E = \{x, y, c, d\}$.[6] For simplicity, and because the theoretico-observational distinction is not relevant for this discussion, let us suppose that the correspondence rules are, in this case, identity functions, therefore, we identify observational and theoretical variables. The set $Evar(H)$ is $\{x, y\}$, and a positive test of H w.r.t. T is any set $\underline{E} = \{\underline{x}, \underline{y}, \underline{c}, \underline{d} \mid \underline{x} = \underline{y}\}$, because, applying conditions (i) to (iv), (i.e., the schema above), $T_x : x = x$ and $T_y : y = y$. This means that whatever values c and d take, the hypothesis $H : x = y$ will not be refuted with respect to theory T provided the evidence satisfies $\underline{x} = \underline{y}$. Notice that a $T'_x : x = y + (c - d)$ is rejected by condition (iv) because there exists $K : y = y + c - d$ with $Evar(K) = \{y, c, d\}$ included in $Evar(T'_x) = \{x, y, c, d\}$ such that $\vdash H \wedge T'_x \leftrightarrow K$. If we eliminate condition (iv) and, therefore, accept $T'_x : x = y + (c - d)$, then the evidence $\underline{E} = \{\underline{x}, \underline{y}, \underline{c}, \underline{d} \mid \underline{x} = \underline{y} \wedge \underline{c} \neq \underline{d}\}$ will refute $H : x = y$ although T does not establish any link between variables x and y, on the one hand, and c and d, on the other.

4 Bootstrap testing of a deterministic program

In this section we apply the Schema I of bootstrap testing presented above to an example. First, let us recall that the way of declaring our intention of what is observable and what is not, is by stating appropriate correspondence rules relating the evidence with the theory (that is, observables can be single quantities and not necessarily a whole sort).

Let SP^{EEQ} be the specification (or theory presentation) over the logic *EEQ* of a program with input $\{x_1, x_2, x_3, x_4\}$ and output x_{10} given at the top of Fig. **2**, where Ax^{BA} is the set of axioms of

[6] We denote by $L : \{\varphi_1, \ldots, \varphi_n\}$ a system named L consisting of the formulae $\varphi_1, \ldots, \varphi_n$. This is abbreviated to $L : \varphi$ if the system consists of just one formula.

Boolean algebra. Suppose we want to verify the integrated circuit at the bottom of Fig. **2** against the specification SP^{EEQ}. The evidence is obviously constituted by sets $\underline{E} = \{\underline{\varepsilon_1}, \underline{\varepsilon_2}, \underline{\varepsilon_3}, \underline{\varepsilon_4}, \underline{\varepsilon_5}\}$ of values for variables ε_1, ε_2, ε_3, ε_4, and ε_5.

Assume that the values of ε_1, ε_2, ε_3, ε_4, and ε_5 in CH0106 are related, respectively, to the variables x_1, x_2, x_3, x_4, and x_{10} in SP^{EEQ} as detailed in Fig. **2**. Notice that the values $\underline{\varepsilon_i}$ for ε_i $(i = 1,2,3,4,5)$ can be either 0V or 5V (where V stands for *volts*), whilst variables x_1, x_2, x_3, x_4, and x_{10} in SP^{EEQ} can only take values **0** and **1**.

The set C^{SP^*} of correspondence rules is therefore constituted by five rules:

- four correspondence rules $C_1^{SP^*}$, $C_2^{SP^*}$, $C_3^{SP^*}$, and $C_4^{SP^*}$, describing the procedure for introducing into the pins labelled ε_1, ε_2, ε_3, and ε_4 of the integrated circuit CH0106, an appropriate signal (0V or 5V) corresponding to a particular valuation (**0** or **1**) for variables x_1, x_2, x_3, and x_4, as well as
- a correspondence rule $C_5^{SP^*}$ describing the measurement procedure to be applied to the output pin labelled ε_5 of CH0106 for assigning to variable x_{10} its corresponding value (i.e., **0** or **1**).

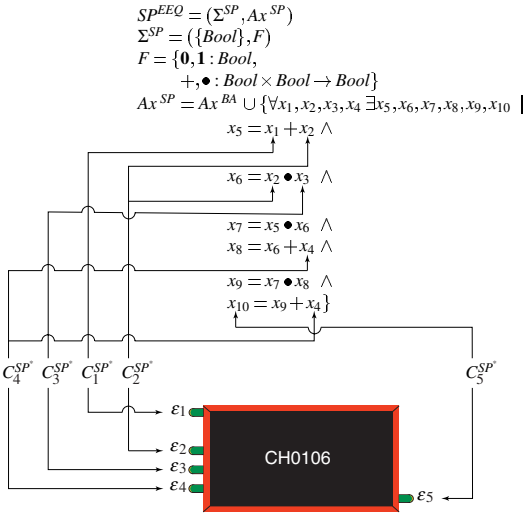

Figure 2: A testing setting

In order to verify the circuit against the specification, we have to derive a set of experiments. Each one of these experiments tests the circuit against a particular hypothesis H on the basis of the (theory generated by the) specification (theory presentation) SP^{EEQ}. The set of chosen hypotheses must cover SP^*. Therefore, a natural choice for the hypotheses are the axioms of the presentation SP^{EEQ}. Let us test, for instance, $H : x_7 = x_5 \bullet x_6$.

At this point let us emphasise that, as is the case in this example, we can do this, *even when the essential variables of the hypothesis are not related to any symbol of the observational vocabulary by any correspondence rule.*

In the general case, what the test can do is to refute our theory or, at best, not to refute it. In this particular case, since the observable variables range over the Booleans, i.e., the set $\{\mathbf{0}, \mathbf{1}\}$, there is a

finite set of possibilities for the evidence, i.e., each one of the combinations of valuations of ε_1, ε_2, ε_3, ε_4, and ε_5 with the set $\{0V, 5V\}$. Therefore, we can exhaustively test H with respect to SP^{EEQ}. Thus, in this particular case, we can confirm H.

Given that $Evar(H) = \{x_5, x_6, x_7\}$, three subtheories $SP^*_{x_5}$, $SP^*_{x_6}$, and $SP^*_{x_7}$ of SP^* should be derived for determining the values for the variables in $Evar(H)$ as functions of $C^{SP^*}(\underline{E})$, where $C^{SP^*}(\underline{E})$ is the evidence $\underline{E}$ 'seen' through the set of correspondence rules C^{SP^*}.

First attempt

For instance, we can present the subtheories as follows:

$$SP_{x_5} : x_5 = x_1 + x_2$$
$$SP_{x_6} : x_6 = x_2 \bullet x_3$$
$$SP_{x_7} : x_7 = (x_1 + x_2) \bullet x_2 \bullet x_3$$

Therefore, we have

$$H, \&SP : \begin{cases} x_7 = x_5 \bullet x_6 \\ x_5 = x_1 + x_2 \\ x_6 = x_2 \bullet x_3 \\ x_7 = (x_1 + x_2) \bullet x_2 \bullet x_3 \end{cases}$$

Notice that $\&SP \vdash H$, and therefore condition (iii) is violated because there exists K, namely $K : x_7 = x_7$, with $Evar(K) = \{x_7\} \subset Evar(H) = \{x_5, x_6, x_7\}$ and both $H, \&SP \vdash K$ and $K, \&SP \vdash H$ (in fact, K could be any tautology with fewer variables than H).

```
program P1
var W1, W2, W3, W4:Bool
function +BBH(x,y:Bool):HBool
function +HBH(x:HBool;y:Bool):HBool
function +HBB(x:HBool;y:Bool):Bool
function •HHH(x:HBool;y:HBool):HBool
function •BBH(x:Bool;y:Bool):HBool
function a(x,y,z,t:Bool):HBool
return x +BBH y
function b(x,y,z,t:Bool):HBool
return y •BBH z
function c(x,y,z,t:Bool):HBool
return a(x,y,z,t) •HHH b(x,y,z,t)
function d(x,y,z,t:Bool):HBool
return b(x,y,z,t) +HBH t
function e(x,y,z,t:Bool):HBool
return c(x,y,z,t) •HHH d(x,y,z,t)
function z(x,y,z,t:Bool):Bool
return e(x,y,z,t) +HBB t
begin
   input(W1, W2, W3, W4)
   output(z(W1,W2,W3,W4))
end
```

Figure 3: A possible implementation

What has violated condition (iii)? We have used a misleading way of calculating a value for the essential variable x_7, i.e., a biased value calculated by using only the 'input evidence' $\underline{E} \setminus \{\underline{\varepsilon_5}\}$ instead of a value calculated involving the measurement $C^{SP^*}_{1 \le i \le 5}(\underline{E})$ of the whole 'input/output' evidence $\underline{E}$. What was wrong in this first attempt? The choice of $\&SP^*$.

Second attempt

So, we must obtain a more appropriate set of subtheories $SP^*_{x_i}$. Notice that, from SP^{EEQ}, we can derive

$$
\begin{aligned}
x_{10} &= x_9 + x_4 \\
&= (x_7 \bullet x_8) + x_4 \\
&= (x_7 \bullet (x_6 + x_4)) + x_4 \\
&= (x_7 \bullet ((x_2 \bullet x_3) + x_4)) + x_4 \\
&= (x_7 + x_4) \bullet ((x_2 \bullet x_3) + x_4 + x_4) \\
&= (x_7 + x_4) \bullet ((x_2 \bullet x_3) + x_4) \\
&= (x_7 \bullet x_2 \bullet x_3) + x_4
\end{aligned}
$$

multiplying both sides by $\overline{x_4}$,

$$
\begin{aligned}
x_{10} \bullet \overline{x_4} &= (x_7 \bullet x_2 \bullet x_3 \bullet \overline{x_4}) + (\overline{x_4} \bullet \overline{x_4}) \\
&= x_7 \bullet x_2 \bullet x_3 \bullet \overline{x_4}
\end{aligned}
$$

which functionally determines a value for x_7 iff $x_2 \bullet x_3 \bullet \overline{x_4} = \mathbf{1}$, i.e., iff $x_2 = \mathbf{1} = x_3$ and $x_4 = \mathbf{0}$. Therefore, we present

$$SP_{x_7} : x_{10} = x_7$$

From SP_{x_7}, H, and SP^{EEQ}, one can derive

$$
\begin{aligned}
x_{10} &= x_5 \bullet x_6 \\
&= (x_1 + x_2) \bullet x_2 \bullet x_3 \\
&= x_2 \bullet x_3
\end{aligned}
$$

which is the so-called *representative* of H.

Notice that

$$
H, \& SP : \begin{cases} x_7 = x_5 \bullet x_6 \\ x_5 = x_1 + x_2 \\ x_6 = x_2 \bullet x_3 \\ x_{10} = x_7 \end{cases}
$$

and there is no K that could violate condition (iii).

Hence, given the restriction $\underline{x_2} = \mathbf{1} = \underline{x_3}$ and $\underline{x_4} = \mathbf{0}$ to the valuation of variables x_1, x_2, x_3, x_4, and x_{10} imposed by the subtheories $SP^*_{x_5}$, $SP^*_{x_6}$, and $SP^*_{x_7}$, the set of test cases is reduced to the two instances obtained by setting x_1 to $\mathbf{0}$ and observing if the value of x_{10} is $\mathbf{0}$ or $\mathbf{1}$, and repeating the same observation after setting x_1 to $\mathbf{1}$. In order to do this, we should use the procedures described in the correspondence rules $C_1^{SP^*}$, $C_2^{SP^*}$, $C_3^{SP^*}$, and $C_4^{SP^*}$ for applying the correct signals to the input pins ε_1,

ε_2, ε_3, and ε_4; and the procedure described in $C_5^{SP^*}$ for measuring the output value in the pin ε_5. Notice that the evidence will provide a positive test of H with respect to SP^* only in the cases in which the value of x_{10} is **1**, i.e., when ε_5 is measured to be $5V$.

```
program P2
var W1,W2,W3,W4:Bool;count:Int
begin
  input(W1,W2,W3,W4)
  if W4 then output(true)
  else count:=0
    if W2 then count:=count+1 endif
    if W3 then count:=count+1 endif
    if count>1 then output(true)
    else output(false)
    endif
  endif
end
```

Figure 4: An alternative implementation

A possible implementation of SP^* is the functional program `P1` in Fig. **3**. Notice, however, that 'an intelligent' programmer or an optimising transformation system (for instance) could have produced the imperative program `P2` in Fig. **4**. In this alternative implementation there are no Boolean functions at all; nevertheless hypothesis $H : x_7 = x_5 \bullet x_6$ can still be tested because SP^{EEQ} also explains the behaviour of `P2`. This example shows the power of the bootstrap strategy in performing black-box testing taking into account only the specification structure, the property under test, and the input/output relation of the program implementing the specification. In [5] we show that `P2` would be rejected by the approach of [9], simply because the `P2` does not implement every axiom of the specification.

5 The bootstrap strategy for non-deterministic evidence

Bootstrap schemata for deterministic evidence provide means for determining whether or not some evidence provides a positive test of a hypothesis with respect to a theory, where a hypothesis is either an equation or an inequality. However, even the latter option of the hypothesis being an inequality does not fully account for the application of bootstrap testing to non-deterministic sets of evidence (as for example the one generated by a non-deterministic program), since the subtheories T^*_x must functionally determine a valuation for $x \in Evar(H)$. (In the simplest case, they explicitly describe functions.) For making bootstrap testing fully applicable to the case of non-deterministic sets of evidence, we need to generalise the setting by allowing the subtheories to (non-vacuously) include inequalities so as to allow the variables to become set valued. (In the discussion below, given a function $f : D \to I$ and given $E \subseteq D$, we let $f(E)$ denote the set $\{f(d) : d \in E\}$.)

Now suppose that, in a setting such as that for schemata for deterministic evidence above, for each $j \in J$, the evidence ε_j takes its values from a set $\underline{\Delta_j}$. Then, for each $x \in Evar(H)$ we have five

possibilities, namely:

1. If $x = f_{T^*_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$,
   then we let $\underline{x} = f_{T^*_x}(C^{T^*}(\underline{\Delta_j} : j \in I_x \wedge I_x \subseteq J))$.

2. If $x > f_{T^*_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$,
   then we let $\underline{x} = \{a :$ if $b \in f_{T^*_x}(C^{T^*}(\underline{\Delta_j} : j \in I_x \wedge I_x \subseteq J))$, then $a > b\}$.
   (An alternative is to make $\underline{x} = \{a :$ there exists $b \in f_{T^*_x}(C^{T^*}(\underline{\Delta_j} : j \in I_x \wedge I_x \subseteq J))$ s.t. $a > b\}$.)

3. If $x < f_{T^*_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$,
   then we valuate x analogously to the preceding case.

4. If $x \leq f_{T^*_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$,
   then the value of $\underline{x}$ is calculated as the union of the set values given by cases 1 and 3.
   In case $x \geq f_{T^*_x}(C^{T^*}(\varepsilon_j : j \in I_x \wedge I_x \subseteq J))$, we proceed analogously.

5. If the value of $x \in Evar(H)$ is determined by a collection of inequalities, then the (set-value) of x is the intersection of the sets given by those inequalities separately.

Now, evidence $\underline{E} = \{\underline{\Delta_j} : j \in J\}$ provides a positive test of a hypothesis H with respect to a theory T^* according to the bootstrap schemata for deterministic evidence, if on the one hand, for each $x \in Evar(H)$ there exists a value in $\underline{x}$ such that these values satisfy H in the usual way, and on the other hand, the other conditions of the schema in use are met.

The reader might be disturbed by the requirement that a single value of $\underline{x}$ for each $x \in Evar(H)$ satisfying H suffices. This seems to mean that a non-deterministic program is considered correct when at least the value produced in one of its executions satisfies H. What about the values $\underline{x}$ produced in other executions not satisfying H, should they be accepted? However, we must recall that the definition of the bootstrap schemata requires that $\underline{E}$ be consistent with H and T^*, and therefore, both pre- and postconditions of the program must be satisfied for the evidence $\underline{E}$ to be able to provide a positive test of H w.r.t. T . This requirement on the evidence $\underline{E}$ stands for an universal quantification overriding the troublesome existential one above for all input/output (observable) variables. Therefore, the weak existential condition above applies only to internal (non-observable) variables.

6 Bootstrap testing of a non-deterministic program

$$
\begin{array}{l}
CONV^{EEQ^+} = (\Sigma^{CONV}, Ax^{CONV}) \\
\Sigma^{CONV} = (\{\mathbb{Q}\}, F) \\
F = \{+, -, \cdot, / : \mathbb{Q}\times\mathbb{Q}\to\mathbb{Q}\} \\
Ax^{CONV} = \{\forall m_1, n_1, m_2, n_2\ \exists m_3, n_3, x, y \mid \\
\qquad\qquad 0 < m_1 < m_2 \quad \wedge \\
\qquad\qquad 0 < n_1 < n_3 \leq \frac{m_2 n_1 - m_1 n_2 + m_3(n_2 - n_1)}{m_2 - m_1} \quad \wedge \\
\qquad\qquad n_2 < n_1 \quad \wedge \\
\qquad\qquad m_3 < 0 \quad \wedge \\
\qquad\qquad y \leq m_1 x + n_1 \quad \wedge \\
\qquad\qquad y \geq m_2 x + n_2 \quad \wedge \\
\qquad\qquad y \leq m_3 x + n_3 \quad \wedge \\
\qquad\qquad x > 0 \quad \wedge \\
\qquad\qquad y > 0 \quad \}
\end{array}
$$

Figure 5: The specification $CONV^{EEQ^+}$

Let us now apply the variant of Schema I for theories containing inequalities and set-valued variables to the verification of a non-deterministic program.

Consider the specification $CONV^{EEQ^+}$ over the logic EEQ^+ given in Fig. **5**. (We assume that the universally quantified variables are sorted by the correspondence rules in such a way that their constraints, i.e., those conditions involving only those variables and perhaps constants, are satisfied.)

A geometrical interpretation of this specification is the one depicted in Fig. **6**, where the surface filled with the parallel vertical line pattern represents the convex polygon defined by the inequalities $y \leq m_1 x + n_1$, $y \geq m_2 x + n_2$, $y \leq m_3 x + n_3$, $x > 0$, and $y > 0$. (That is, the polygon enclosed by the lines $L_1 : y = m_1 x + n_1$, $L_2 : y = m_2 x + n_2$, $L_3 : y = m_3 x + n_3$, and the coordinate axes.) Notice that the convex polygon in Fig. **6** satisfies the conditions $0 < m_1 < m_2$ and $m_3 < 0$ in $CONV^{EEQ^+}$. Notice as well that the point $\langle x_0, y_0 \rangle$ exists because $m_1 \neq m_2$, and that therefore, $k = \frac{m_2 n_1 - m_1 n_2 + m_3(n_2 - n_1)}{m_2 - m_1}$ also exists. Finally, notice that the convex polygon in Fig. **6** also satisfies conditions $0 < n_1 < n_3 \leq k$ and $n_2 < n_1$ in $CONV^{EEQ^+}$.

As is the case with any specification, $CONV^{EEQ^+}$ can specify many programs, in particular the non-deterministic program `Pnondet` whose input/output diagram is the one depicted in Fig. **7**. We will consider that the correspondence rules are identity functions, therefore for the sake of simplicity we will use the set $\{m_1, n_1, m_2, n_2, x, y\}$ as the evidence, instead of using the actual evidence $E = \{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5, \varepsilon_6\}$ and the set of correspondence rules. Here, we assume that the universally quantified variables are sorted by the cor-

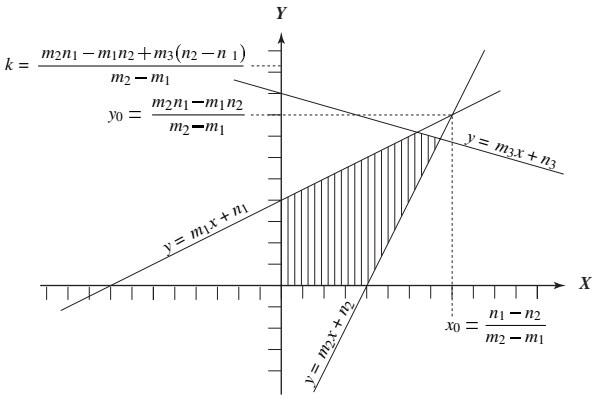

Figure 6: A geometrical interpretation of $CONV^*$

respondence rules in such a way that constraints imposed on them by $CONV^{EEQ^+}$ are satisfied. In other words the data $\{\underline{m_1}, \underline{n_1}, \underline{m_2}, \underline{n_2}\}$ with which program `Pnondet` is fed is such that $0 < \underline{m_1} < \underline{m_2} \wedge \underline{n_2} < \underline{n_1} \wedge 0 < \underline{n_1}$.

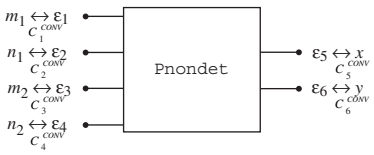

Figure 7: The input/output diagram of `Pnondet`

The informal description of the intended semantics of the program `Pnondet` is as follows: each time a set of values $\{\underline{m_1}, \underline{n_1}, \underline{m_2}, \underline{n_2}\}$ is fed into `Pnondet`,

1. it randomly chooses suitable values $\underline{m_3}$ and $\underline{n_3}$ for m_3 and n_3, respectively, satisfying the conditions imposed on them by $CONV^{EEQ^+}$;

2. then, it chooses, also randomly, any pair $\langle \underline{x}, \underline{y} \rangle$ of coordinates for a point lying inside the convex polygon defined by the constraints $y \leq \underline{m_1}x + \underline{n_1}$, $y \geq \underline{m_2}x + \underline{n_2}$, $y \leq \underline{m_3}x + \underline{n_3}$, $x > 0$, and $y > 0$ also imposed by $CONV^{EEQ^+}$;

3. finally, it outputs the values $\underline{x}$ and $\underline{y}$.

Thus, `Pnondet` has an obviously nondeterministic behaviour, since, after randomly choosing values for the coefficients of the straight line $y = m_3x + n_3$, (thus defining a particular convex polygon), it produces as output an also randomly chosen pair $\langle \underline{x}, \underline{y} \rangle$ of coordinates defining a point lying inside this convex polygon.

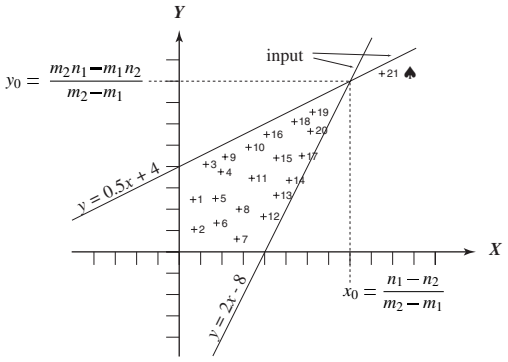

Figure 8: Output generated by 21 executions of `Pnondet`

Now, let us suppose we fed `Pnondet` 21 consecutive times with values $\underline{m_1} = 0.5$, $\underline{n_1} = 4$, $\underline{m_2} = 2$, and $\underline{n_2} = -8$. As we said above, each time (i.e., in each execution), `Pnondet` will first randomly choose values $\underline{m_3}$ and $\underline{n_3}$, thus defining a particular convex polygon.

In each one of these 21 executions, `Pnondet` produces an output pair of values $\langle \underline{x}, \underline{y} \rangle$ defining the points depicted in Fig. **8** (see also in the table in Fig. **9**, columns m_1, n_1, m_2, n_2, x, and y, which exhibit the input/output relation defined by the 21 executions of `Pnondet` in question).

The purpose of this example is to test (using the variant of Schema I for non-deterministic programs) whether or not the behaviour of `Pnondet` satisfies its intended semantics (and, therefore, if it is correct with respect to $CONV^{EEQ^+}$). Thus, what we should do now is to test each one of the axioms of $CONV^{EEQ^+}$ (on the basis of $CONV^*$), using the input/output relation produced by the 21 executions of `Pnondet` as evidence. To do this, we need to derive, for each one of the essential variables z of each one of these axioms, an appropriate $CONV^*_z$. However, it will suffice as an example to analyse only the testing of $H_1 : m_3 < 0$ and $H_2 : n_3 \leq k$.

$H_1 : m_3 < 0$ $H_2 : n_3 \leq k$ (a)

	input				output						
	m_1	n_1	m_2	n_2	point number	x	y	$CONV_{m_3} : m_3 < m_1$	n_3 lowerbound calculated with: $CONV_{n_3}$ $y < n_3$	m_3 upperbound calculated with: $CONV_{m_3} : m_3 < 0$	k least upperbound calculated using $m_3 = 0$
	0.5	4	2	-8	1	0.65	2.45	0.5	2.45	0	8
	0.5	4	2	-8	2	0.70	1.05	0.5	1.05	0	8
	0.5	4	2	-8	3	1.25	4.10	0.5	4.10	0	8
	0.5	4	2	-8	4	1.95	3.75	0.5	3.75	0	8
	0.5	4	2	-8	5	1.70	2.50	0.5	2.50	0	8
	0.5	4	2	-8	6	1.75	1.35	0.5	1.35	0	8
	0.5	4	2	-8	7	2.70	0.60	0.5	0.60	0	8
	0.5	4	2	-8	8	2.80	2.00	0.5	2.00	0	8
	0.5	4	2	-8	9	2.15	4.45	0.5	4.45	0	8
	0.5	4	2	-8	10	3.25	4.90	0.5	4.90	0	8
	0.5	4	2	-8	11	3.40	3.45	0.5	3.45	0	8
	0.5	4	2	-8	12	3.95	1.65	0.5	1.65	0	8
	0.5	4	2	-8	13	4.55	2.65	0.5	2.65	0	8
	0.5	4	2	-8	14	5.15	3.35	0.5	3.35	0	8
	0.5	4	2	-8	15	4.55	4.40	0.5	4.40	0	8
	0.5	4	2	-8	16	4.10	5.50	0.5	5.50	0	8
	0.5	4	2	-8	17	5.75	4.50	0.5	4.50	0	8
	0.5	4	2	-8	18	5.40	6.10	0.5	6.10	0	8
	0.5	4	2	-8	19	6.25	6.55	0.5	6.55	0	8
	0.5	4	2	-8	20	6.15	5.65	0.5	5.65	0	8
♠	0.5	4	2	-8	21	9.50	8.35	0.5	8.35	0	8

(a) with $k = \frac{m_2 n_1 - m_1 n_2 + m_3(n_2 - n_1)}{m_2 - m_1}$

Figure 9: Data for the testing of hypotheses H_1 and H_2

So, let us begin with $H_1 : m_3 < 0$. It is obvious that it has just one essential variable, i.e., m_3. Therefore, we need to derive only a subtheory $CONV^*_{m_3}$. Notice that for this derivation, we can use all the axioms of $CONV^{EEQ^+}$ with the exception of $m_3 < 0$ itself, because we would otherwise violate condition (iii) of Schema I. To see why, suppose we allow the use of $m_3 < 0$ in the derivation of $CONV^*_{m_3}$, then it is obvious that $\&CONV \vdash H_1$. This leads to the violation of the said condition (iii) because for any ground tautology K in $CONV^*$, since K has no variables and hence $Evar(K) \subset Evar(H_1) = \{m_3\}$, obviously $\&CONV, K \vdash H_1$ and $\&CONV, H_1 \vdash K$ holds.

Then, let us use the inequalities

$$n_1 < n_3 \leq \frac{m_2 n_1 - m_1 n_2 + m_3(n_2 - n_1)}{m_2 - m_1}$$

From them we obtain that

$$\begin{aligned} n_1 &< \tfrac{m_2n_1 - m_1n_2 + m_3(n_2 - n_1)}{m_2 - m_1} \\ (m_2 - m_1)n_1 &< m_2n_1 - m_1n_2 + m_3(n_2 - n_1) \text{ since } m_2 - m_1 > 0 \\ m_2n_1 - m_1n_1 &< m_2n_1 - m_1n_2 + m_3(n_2 - n_1) \\ m_1n_2 - m_1n_1 &< m_3(n_2 - n_1) \\ m_1(n_2 - n_1) &< m_3(n_2 - n_1) \\ m_1 &> m_3 \text{ since } n_2 - n_1 < 0 \end{aligned}$$

that is,

$$CONV_{m_3} : m_3 < m_1$$

which is a presentation of $CONV^*_{m_3}$.

In the table of Fig. **9** we have one column under the key $H_1 : m_3 < 0$. Notice that $CONV_{m_3}$ is of the type referred to in case 3 (on page 14). Therefore $\underline{m_3} = (-\infty, 0.5]$ and the $H_1 : m_3 < 0$ is satisfied since there exists a value in $\underline{m_3}$ smaller than 0.

Let us now test the hypothesis $H_2 : n_3 \leq k = \frac{m_2n_1 - m_1n_2 + m_3(n_2 - n_1)}{m_2 - m_1}$ on the basis of $CONV^*$. The set $Evar(H_2)$ of essential variables of H_2 is $\{m_1, n_1, m_2, n_2, m_3, n_3\}$. Given that m_1, n_1, m_2, n_2 are part of the evidence, their corresponding subtheories can functionally determine them. So, we set

$$\begin{aligned} &CONV_{m_1} : m_1 = m_1 \\ &CONV_{n_1} : n_1 = n_1 \\ &CONV_{m_2} : m_2 = m_2, \text{ and} \\ &CONV_{n_2} : n_2 = n_2 \end{aligned}$$

From $y \leq m_3x + n_3$, and given that, according to $CONV$, n_3 as well as x are positive and m_3 negative, we can deduce that $y < n_3$. The subtheories for m_3 and n_3 are then:

$$\begin{aligned} &CONV_{m_3} : m_3 < 0, \text{ and} \\ &CONV_{n_3} : y < n_3 \end{aligned}$$

Recall that a subtheory T^*_z must constrain the value of z using only the evidence. Therefore, in the case of m_3, we cannot use the inequality $y \leq m_3x + n_3$, since we cannot eliminate n_3, even using other parts of $CONV$.

In the table of Fig. **9** we have three columns, now under the key $H_2 : n_3 \leq k$ giving bounds for the values of n_3, of m_3, and of k.

Notice that $CONV_{n_3}$ is of the type referred to in case 2 (on page 14). Therefore $\underline{n_3} = (8.35, \infty)$ and $H_2 : n_3 \leq k$ is not satisfied since there exists a value in $\underline{m_3}$ that makes k smaller than 8.35. (See column 'k lower bound using $m_3 = 0$,' calculated using the least upper bound 0 for m_3.) This is due to the point number 21 (labelled with ♠ both in Fig. **8** and in the table in Fig. **9**), which is obviously outside any possible convex polygon.

Therefore, `Pnondet` is not correct with respect to $CONV^*$.

If we analyse the table of Fig. **9**, we can consider the upperbound calculated for m_3 in the test of $H_1 : m_3 < 0$ to be too coarse, because for any point i in Fig. **8**, the least upperbound for m_3 –not even considering in its calculation the hypothesis H_1– will be 0. Since the point i will obviously be inside the convex polygon limited by L_1, L_2, the coordinate axes, and the line given by $y = y_i$, $m_3 = 0$ will be a finer upperbound. However, we are testing `Pnondet` against the whole theory. Thus, we should also consider that for this point i, we have a greatest lowerbound for n_3 namely $n_3 = y_i$. Therefore, the line limiting the minimum polygon in which the point i lies will be $L : y = mx + n$ with $m < m_1$ and $n > y_i$; this line L is $y = y_i$.

7 Conclusions

What we have presented is a very promising and powerful approach to specification-based verification and validation testing of software process artefacts (i.e., software artefacts as well as specifications). We have shown that the approach is useful for both deterministic and non-deterministic programs and specifications.

This approach was developed using the general epistemological background sketched in Sects. 1, 3, and 5. We would like to emphasise that in doing this, as theoretical computer scientists, we proceeded *more like epistemologists than as logicians or mathematicians* in conducting our quest.

On the same basis, in Sect. 2 we have also discussed the flaws of the best known alternative to the bootstrap strategy, namely the hypothetico-deductive strategy. These flaws prevent its use for sustaining testing methodologies.

Whether in the framework of verification testing, it is worth noting the similarity between the bootstrap core idea and the notion of implementation relation defined by refinement. Recall in Fig. **1** the valuation α induced by $\underline{E}$, that has to be such that $\alpha \models H$; in other words, the bootstrap strategy requires that evidence provide instances of the hypothesis under test. Refinement requires that the

relation defined by (the semantics of) a program be contained in the set of relations defined by (the semantics of) a specification; see [1]. Thanks to the deduction theorem, both just mean that evidence resp. program implies the specification.

As we said in the introduction, 'exhaustive' testing has two different dimensions. One is the 'coverage' by the test of the whole specification to be tested. The other is the 'exhaustion' of the test w.r.t. the (possibly infinite) domain of interpretation of the symbols in the specification. It seems that our approach takes care of the first of these dimensions. However, we have presented it just for theories over an existential (in)equational logic. Then, the bootstrap-strategy-based approach should be extended to deal with theories over other logics of interest (e.g., classical first-order, temporal, deontic, and dynamic logics) and with various kinds of semantics (for instance, transition systems). This is a must if we intend to turn the approach to a practical one. Also the bootstrap testing of structured specifications and systems must be considered. The second dimension (which is the problem addressed by Gaudel with her 'uniformity hypotheses') must be studied, for instance, in the light of Carnap's and Hintikka's results on inductive logic and the general methodology of (scientific) induction [4, 13].

Comparing the bootstrap strategy with real software testing developments is a crucial issue. This comparison should begin by an exhaustive analysis of the current testing methods in the light of the epistemological framework here presented. A starting point could be, for instance, the method reported in [6], whose similarity with this strategy was pointed out by C. Heitmeyer.

References

[1] M. Bidoit, M. V. Cengarle, and R. Hennicker. Proof systems for structured specifications and their refinements. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*. Springer-Verlag, 1999.

[2] R. Carnap. *Der logische Aufbau der Welt*. Weltkreis-Verlag, Berlin, 1928.

[3] R. Carnap. *Philosophical foundations of physics*. Basic books, New York, 1966.

[4] R. Carnap. *Logical Foundations of Probability*. Univ. of Chicago Press, Chicago–Illinois, 1977. 2nd rev. edition, 1962.

[5] M. V. Cengarle and A. M. Haeberer. Towards an epistemology-based methodology for verification and validation testing. Technical Report 0001, LMU München, Inst. für Informatik, Jan. 2000. 71 pages.

[6] A. Gargantini and C. Heitmeyer. Using model checking to generate tests from requirements specifications. In O. Nierstrasz and M. Lemoine, editors, *Proc. of ESEC/FSE'99*, volume 1687 of *LNCS*. Springer-Verlag, May 1999.

[7] M.-C. Gaudel. Testing can be formal, too. In P. D. Mosses, M. Nielsen, and M. I. Schwartzbach, editors, *Proc. of TAPSOFT'95*, volume 915 of *LNCS*, pages 82–96. Springer-Verlag, May 1995.

[8] M.-C. Gaudel and P. R. James. Testing data types and processes, an unifying theory. In *3rd ERCIM on FMICS'98*. CWI, May 1998.

[9] M.-C. Gaudel and P. R. James. Testing algebraic data types and processes: a unifying theory. *Formal Aspects of Computing*, 1999. To appear.

[10] C. Glymour. Hypothetico-deductivism is hopeless. *Philosophy of science*, 47:322–325, 1980.

[11] C. Glymour. *Theory and Evidence*. Princeton Univ. Press, New Jersey, 1980.

[12] A. M. Haeberer and T. S. E. Maibaum. The very idea of software development environments: A conceptual architecture for the arts environment. In B. Nuseibeh and D. Redmiles, editors, *Proc. of ASE'98*, pages 260–269. IEEE CS Press, 1998.

[13] J. Hintikka. Toward a theory of inductive generalization. In Y. Bar-Hillel, editor, *Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science*, pages 274–288, Amsterdam, 1962. Stanford Univ. Press.

[14] M. M. Lehman. *Program Evolution – Processes of Software Change*. Academic Press, New York, 1985. ISBN 0-12-442441-4.

[15] G. H. Merrill. Confirmation and prediction. *Philosophy of science*, 46:98–117, 1979.

[16] C. K. Waters. Relevance logic brings hope to hypothetico-deductivism. *Philosophy of science*, 54:453–464, 1987.