Hoare-Like Triples and Kleene Algebras with Top and Tests
Towards a Holistic Perspective on Hoare Logic, Incorrectness Logic, and Beyond

Lena Verscht
Saarland University
lverscht@cs.uni-saarland.de

Benjamin Kaminski
Saarland University and University College London
kaminski@cs.uni-saarland.de

1 Introduction
“Program correctness and incorrectness are two sides of the same coin.”
— O’Hearn [9]

We argue that the object of discourse is not a coin and that it has at least three sides or rather dimensions:

1. correctness vs. incorrectness
2. totality vs. partiality
3. reachability vs. unreachability

We will explore how one can pigeonhole total and partial correctness, as well as incorrectness into this view and explore further program properties that emerge from exhausting all combinations of the above dimensions. We will furthermore explore how to express validity of those properties in the language of Kleene algebras with top and tests [11].

2 Canonical Exegeses of Triples
The central notion of Hoare logic [4] are triples of the form \{b\} p \{c\}, where p is a program, b is a precondition on initial states, and c is a postcondition on final states. To give meaning to a triple, it must be exegited when a triple is considered valid and when it is not. For Hoare logic, there are two such exegeses: validity for total and for partial correctness.

**Exegesis I: Partial Correctness.** \{b\} p \{c\} is valid for partial correctness iff all executions of p that start in b can only ever terminate in c. The execution is allowed to diverge. For example, validity for partial correctness of

\[
\{\text{wrong password}\}, p_{login} \quad \text{\{login failed\}}
\]

specifies that when the user enters a wrong password, \(p_{login}\) will either diverge or terminate with a failed login. In particular, successful login is impossible with the wrong password.

**Exegesis II: Total Correctness.** \{b\} p \{c\} is valid for total correctness iff all executions of p that start in b definitely terminate and they do so in c. Reusing the example above,

\[
\{\text{correct password}\}, p_{login} \quad \text{\{login successful\}}
\]

specifies that when the user enters the correct password, the login will definitely be successful.

It is noteworthy that both partial and total correctness are about coreachability of all initial states: \{b\} p \{c\} is valid if all initial states in b are (partially) mapped by p into (but not necessarily onto) the final states in c.

**Weakest (Liberal) Preconditions.** Proposed by Dijkstra [2], weakest (liberal) preconditions are a very versatile alternative to Hoare logic and led to numerous extensions [5, 6, 8, 12]. Given only a program p and a postcondition c, the weakest liberal precondition is the weakest predicate \(wp[p]\) (c), such that all executions of p that start in \(wp[p]\) (c) either diverge or terminate in c. One can (re)define partial correctness in terms of weakest liberal preconditions: \{b\} p \{c\} is valid for partial correctness iff \(b \implies wp[p]\) (c).

For total correctness, the weakest precondition is the weakest predicate \(wp[p]\) (c), such that all executions of p that start in \(wp[p]\) (c) definitely terminate and they do so in c. One can define total correctness in terms of weakest preconditions: \{b\} p \{c\} is valid for total correctness iff \(b \implies wp[p]\) (c).

**Exegesis III: Incorrectness.** Incorrectness triples were introduced by De Vries and Koutavas [1] and later independently reintroduced by O’Hearn [9]. \{b\} p \{c\} is valid for incorrectness iff all executions of p that terminate in c could have started in b. In other words: iff p maps (a subset of) b onto c. In that sense, incorrectness is really about reachability of all final states. If c is a set of error states, then validity of \{b\} p \{c\} for incorrectness indeed proves that executing p on b is not safe, because doing so can reach an error in c.

For example, validity for incorrectness of

\[
\{\text{correct password}\}, p_{login} \quad \text{[program crash]}
\]

specifies that \(p_{login}\) can crash on entering a correct password.

**Strongest Postconditions.** Given only a program p and a precondition b, the strongest postcondition [3] is the strongest
This immediately gives rise to two questions: What about sp\[p\] (b) such that any state in sp\[p\] (b) is reachable from b by executing p. We can redefine incorrectness in terms of strongest postconditions [12]: [b] p \[c\] is valid for incorrectness iff c \(\Rightarrow\) sp\[p\] (b).

sp has a Galois connection to wp, namely

\[ b \Rightarrow wp[p] (c) \quad \text{iff} \quad sp[p] (b) \Rightarrow c. \quad (\dagger) \]

Because of this, we can also define partial correctness in terms of strongest postconditions: \{b\} p \[c\] is valid for partial correctness iff sp\[p\] (b) \(\Rightarrow\) c. Notice that the difference between incorrectness and partial correctness is merely the direction of the implication between c and sp\[p\] (b).

3 Complete Exegeses of Triples

Given a triple of precondition b, program p, and postcondition c, we have so far seen three exegesis of such triples, all given in terms of wlp/wp/sp and implications:

- **Total Correctness**
  \[ b \Rightarrow wp[p] (c) \]

- **Partial Correctness**
  \[ b \Rightarrow wp[p] (c) \]
  \[ sp[p] (b) \Rightarrow c \]

- **Incorrectness**
  \[ c \Rightarrow sp[p] (b) \]

From inspecting this table, we can make out three degrees of freedom one has in exegeting triples:

1. The difference between total correctness and partial correctness is the usage of wp vs. wlp.
2. The difference between partial correctness and incorrectness is the direction of the implication.
3. The difference between total correctness and incorrectness is wp vs. sp.

This immediately gives rise to two questions: What about wp\[p\] (c) \(\Rightarrow\) b and wlp\[p\] (c) \(\Rightarrow\) b? And can we define strongest liberal postconditions?

**Strongest Liberal Postconditions.** Indeed, these can be sensibly defined [12]. The difference between wlp and wp is that the liberal variant additionally contains all states that diverge, i.e. those initial states for which there exists no final state in which the execution of p could terminate. Dually, slp\[p\] (b) contains also all states that cannot be reached at all by executing p, i.e. those final states for which there exists no initial state in which the execution of p could have started.

slp also has a Galois connection [12], but to wp, namely

\[ wp[p] (c) \Rightarrow b \quad \text{iff} \quad c \Rightarrow slp[p] (b). \quad (\ddagger) \]

Exhaustively combining wp, wlp, sp, and slp with all implication directions gives a completed canon of in total 8 different exegeses of triples, four of which reduce to only two via Galois connections, see Figure 1.

**Exegesis IV: Partial Incorrectness.** The difference between total and partial correctness is the use of wp vs. wlp. Consequently, when we replace sp by slp in the defining implication c \(\Rightarrow\) sp\[p\] (b) of incorrectness, we should arrive at the exegesis of partial incorrectness, as suggested (but not further explored) by Zhang and Kaminski [12]: [b]p\[c\] is valid for partial incorrectness iff c \(\Rightarrow\) slp\[p\] (b) (and by (\ddagger) equivalently iff wp\[p\] (c) \(\Rightarrow\) b), which means that all states in c are either unreachable (from anywhere) or reachable from b by executing p. For example, validity for partial incorrectness of

\[ \text{[wrong password]} \quad p_{login} \quad \text{[login successful]} \]

specifies that any state where the login was successful is either entirely unreachable, or reachable by entering a wrong password, thus constituting a potential bug.

**Exegesis V and VI.** There are two remaining triple exegeses, defined by wlp\[p\] (c) \(\Rightarrow\) b and slp\[p\] (b) \(\Rightarrow\) c, respectively. These have also been suggested by Zhang and Kaminski [12], but not further explored.

**Contrapositives.** Besides some actual equivalences via Galois connections, some triples are contrapositives of each other [12]. In fact, the entire left-hand-side (in some sense...
the correctness side) is the contraposition of the right-hand-side (the incorrectness side) in Figure 1. For example,
\[ b \implies \text{wp}[p](c) \iff \text{wp}[p](\neg c) \implies \neg b \]
and hence partial correctness is the contrapositive of partial incorrectness and so both describe in some sense the same class of specifications. We would claim, however, that it might be more intuitive for a working programmer to specify either a partial incorrectness triple or a partial correctness triple, depending on the property they have in mind.

4 Kleene Algebra with Top and Tests

Kleene algebra with tests (KAT), introduced by Kozen [7], is another alternative for specifying and reasoning about program properties. KAT terms are generalized regular expressions over a two-sorted alphabet comprising (i) programs \((p, q, \ldots)\) and (ii) tests \((b, c, \ldots)\). We interpret these symbols as relations: a program \(p\) relates (maps) initial states to final states by its execution. Initial states on which \(p\) diverges are not related with any final states. Similarly, unreachable final states are not related with any initial states. A test \(b\) maps initial states satisfying \(b\) to themselves, and those not satisfying \(b\) to no state. In this sense, tests act as filters.

Testing for false is denoted by 0 and gives the empty relation. In the lattice of relations (ordered by set inclusion), 0 is the least bottom element. In Kleene algebra with top and tests (TopKAT), we now additionally add a \(\top\) element to KAT, which codes for the universal relation relating every initial state to every final state [11].

Composing symbols means composing relations: For example, the term \(bpc\) intuitively means: first test for \(b\), then execute \(p\) (but only on states that satisfy \(b\)), then execute \(q\); then test for \(c\). Executions that fail to satisfy initially \(b\) or finally \(c\) are filtered out and not part of the resulting relation.

**Partial Correctness in TopKAT.** We can express partial correctness by the TopKAT equation \(\top bpc = \top bp\). Prepending the top element on both relations in the equation amounts to comparing their codomains, i.e. their final states. Hence, \(\top bpc = \top bp\) expresses that the set of final states in \(c\) in which \(p\) can terminate starting from \(b\) is exactly those final states in which \(p\) can terminate from \(b\) at all. No statement is made about initial states in \(b\) on which \(p\) diverges.

**Incorrectness in TopKAT.** It is not possible to express incorrectness in KAT, but this is possible in TopKAT [11], namely by \(\top bpc = \top c\). On the left-hand-side of the equation, we select all final states in \(c\) that were reachable by executing \(p\) on \(b\). On the right-hand-side, we select all final states in \(c\).

**Nondeterminism and Correctness in KAT.** So far, we have kept quiet about nondeterminism, but in Kleene algebras it is very natural to model nondeterministic programs. Considering total correctness of \(\{b\} p \{c\}\), one then has to decide:

(i) must all execution paths emerging from any state in \(b\) terminate in \(c\), or (ii) must there merely exist some path that does so? The former is called the demonic exegesis of nondeterminism while the latter is called angelic. On that note, Zhang and Kaminski [12] always interpret nondeterminism angelically for their nonliberal pre- and postconditions, and demonically for their liberal ones. Only this way, they obtain the Galois connections \((\top)\) and \((\top)\). However, the standard notion of total correctness is demonic and not angelic.

Unfortunately, it is known that demonic total correctness is inexpressible in KAT [10] because it lacks a way of reasoning about nontermination: Nonterminating executions are just not part of the resulting relation. For the angelic variant, however, \(bpc\top = b\top\) indeed expresses that from all initial states in \(b\), it is possible for \(p\) to terminate in \(c\), thus:

<table>
<thead>
<tr>
<th>{b} p {c} is valid for angelic total correctness</th>
</tr>
</thead>
<tbody>
<tr>
<td>(bpc\top = b\top)</td>
</tr>
</tbody>
</table>

This is, to the best of our knowledge, a novel result. Moreover, this result renders all triples depicted in Figure 1 expressible in TopKAT in a syntactically very similar manner:

| total correctness | \(bpc\top = b\top\) |
| partial correctness | \(\top bpc = \top bp\) |
| incorrectness      | \(\top bpc = \top c\) |

The opposite-hand-sides are expressible via contrapositives. For example, validity of \(\{b\} p \{c\}\) for partial incorrectness is expressible via \(bpc\top = p\top c\). And more equations fit into this pattern, for example \(\top bpc = \top pc\) and \(bpc\top = bp\top c\). Interestingly, these two express the angelic variants of partial correctness and incorrectness, respectively, and thus lead to further exegeses of triples.

5 Ongoing and Future Work

There are many open questions left to investigate. One direction is to investigate deeper the different exegeses of nondeterminism, which would lead to a total combination of 16 possible exegeses, of which we conjecture that still only four of them reduce to two via Galois connections, thus ending up with 14 possible exegeses. We would then like to investigate the impact of different assumptions about the program on this picture: For example, what happens if we assume that executions started in \(b\) always terminate? What happens if all states in \(c\) are reachable? What if the program is deterministic or reversible?

As for a different direction, none of the aforementioned triples expresses the existence of (at least) one path from \(b\) to \(c\). However, being able to more directly specify a bug like “it
is possible to enter a wrong password and yet obtain successful login” would be quite desirable. The following equations would express a property of this form:

\[ bpc \neq 0 \] and equivalently \[ \text{wp}[p] (c) \land b \neq false \]

In contrast to the existing triples, this property speaks neither about all initial states in \( b \) nor all final states in \( c \).

References


