Refutation of program verification by reduction

Abstract: The formula for program verification by reduction is *not* tautologous, thereby refuting such an approach and forming a *non* tautologous fragment of the universal logic VŁ4.

We assume the method and apparatus of Meth8/VŁ4 with Tautology as the designated proof value, **F** as contradiction, N as truthity (non-contingency), and C as falsity (contingency). The 16-valued truth table is row-major and horizontal, or repeating fragments of 128-tables, sometimes with table counts, for more variables. (See ersatz-systems.com.)

LET ~ Not, ¬ ; + Or, ∨ , ∪ , ⊔ ; - Not Or; & And, ∧ , ∩, ⊓ , · ; \ Not And;
> Imply, greater than, → , ⇒ , ↦, > , ⊃ , ↠ ; < Not Imply, less than, ∈ , < , ⊂ , ⊬ , ⊭ , ↞ , ≲ ;
= Equivalent, ≡, :=, ⇔, ↔, ≙, ≈, ≃ ; @ Not Equivalent, ≠;
% possibility, for one or some, ∃ , ◊, M; # necessity, for every or all, ∀ , □, L;
(z=z) T as tautology, ⊤, ordinal 3; (z@z) **F** as contradiction, Ø, Null, ⊥ , zero;
(%z>#z) N as non-contingency, Δ, ordinal 1; (%z<#z) C as contingency, ▽, ordinal 2;
~(y < x) (x ≤ y), (x ⊆ y), (x ⊑ y); (A=B) (A~B).
Note for clarity, we usually distribute quantifiers onto each designated variable.

From: Barthe, G.; Eilers, R.; Georgiou, P.; Gleiss, B.; Kovács, K.; Maffei, M. (2019). Verifying relational properties using trace logic. arxiv.org/pdf/1906.09899.pdf
arxiv.org/pdf/1906.09899.pdf

1 Introduction

Program verification generally focuses on proving that all executions of a program lie within a specified set of executions, that is, properties are seen as sets of traces. However, this approach is not general enough to capture various fundamental properties, such as non-interference ... and robustness … . These notions are naturally modeled as relational properties, that is as properties over sets of pairs of traces. Relational properties are special instances of hyperproperties [15], which are formally defined as sets of sets of traces. Verification of relational properties can be achieved in different ways. One approach is by reduction to program verification:

> given a program P and a hyperproperty φ, construct a program Q and a property ψ, such that: (i) Q verifies ψ and (ii) Q verifies ψ implies P verifies φ. … (1.1)

LET p, q, r, s: P, Q, φ, ψ.

((p&r)>(q&s))>((q>s)&((q>s)>(p>r))) ;
TFFF TT**F**T T**F**T**F** TTTT (1.2)

Remark 1.2: Eq. 1.2 is *not* tautologous, thereby refuting such an approach by reduction to program verification.