How computers must think

Abstract: We evaluate the two requirements for how computers must think: four-valued logic and the loop with sentinel structure. The logic allows for discovery of the correct mapping of the structure as a theorem.

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.)

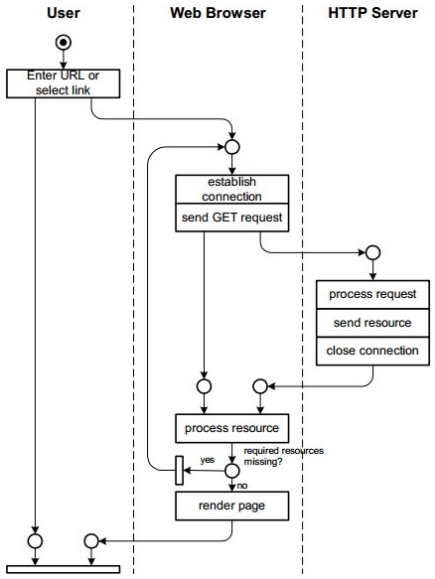
LET ~ Not, ¬ ; + Or, ∨, ∪, ⊔ ; - Not Or; & And, ∧, ∩, ⊓, ·, ∘ , ⊗ ; \ Not And;
> Imply, greater than, ⟶, ⇒ , ↦, ≻, ⊃, ↠ ; < Not Imply, less than, ∈, ≺, ⊂, ⊮, ⊭, ↞, ≲ ;
= Equivalent, ≡, :=, ⇔, ↔, ≙, ≈, ≃ ; @ Not Equivalent, ≠, ⊕;
% possibility, for one or some, ∃, ∃!, ◊, M; # necessity, for every or all, ∀, □, 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.

For computers to think, the two requirements are a four-valued logic and tautology of the programming structure for the loop with sentinel. The quaternary logic is needed to differentiate tautology from truthity, and contradiction from falsity, as shown farther below when evaluating the loop and sentinel in this table.

Loop	Do-loop	Do-loop while	Do-loop until	Do-while loop	Do-until loop	For-next loop
Sentinel	exit	while	until	while	until	next

[See next page.]

A single looping example suffices. (From the FMC behavior diagram as a Petri net at the Apache modelling project (2008).)



(2.1.1.1)

Remark 2.1.1: For our purpose, the return node from below r can point directly to p.

LET p establish_connection,
q process_request,
r process_resource,
s render_page.

(p>q)>(r>(p+s)) ; TTTT **F**T**F**T TTTT TTTT (2.1.1.2)
(p>q)>(~(r>p)>s) ; TTTT **F**T**F**T TTTT TTTT (2.1.1.3)

Remark 2.1.1.1: It is the four-valued logic that exposes the mappings of Eqs. 2.1.1.2 and 2.1.1.3 as *not* tautologous.

The consequent of Eq. 2.1.1.2 reads "process_resource result implies fall through to render_page or loop back to establish_connection".

The consequent of Eq. 2.1.1.3 reads "if process_resource result does not imply loop back to establish_connection then fall through to render_page".

To coerce 2.1.1.1 into a tautology we inject the literal of the antecedent into the consequent as follows.

The consequent of 2.1.1.2 is to read "process_resource result implies if establish_connection then process_request as loop back or fall through to render_page". (2.1.2.1)

(p>q)>(r>((p>q)+s)) ; TTTT TTTT TTTT TTTT (2.1.2.2)

The consequent of 2.1.1.3 is to read "if process_resource result does not imply if establish_connection then process_request as loop back, then fall through to render_page". (2.1.3.1)

(p>q)>(~(r>(p>q))>s) ; TTTT TTTT TTTT TTTT (2.1.3.2)