-EN_52: Translators and compilers for V

Author: Víctor Estrada Díaz. Framework: Einstein-VED - UNIHOLOG.
Date: 10 October 2026.

Project references: Base document · Q-V construction and comparison plates · Presentation to the community · Complete resource index.

Dual License - Ethics and Use

- **Text (CC BY-NC-SA 4.0)**: sharing, adaptation, noncommercial use, attribution and share alike.
- **Code (HL3)**: commercial use permitted only if it complies with ethical principles. Prohibited: wars, human rights violations, environmental harm, exploitation, mass surveillance and discrimination.

Ethical principles declared by the author: respect for human dignity, no use for profit without permission and prohibition of harmful uses. Statement · CC BY-NC-SA 4.0 · HL3.

Purpose and rules

This tool translates a known Q circuit into a construction plan for V. It does not query answers from Q or compute a result beforehand to hide it until measurement. It preserves gates, controls, connections and

order. It does not introduce an amplitude register or iterate over possible results.

V is the author's deterministic machine, realizable with conventional electronics or software. Lack of knowledge before measurement does not add a random mechanism to its dynamics.

In V, the outputs remain unknown before measurement. Joint readout is triggered at $t = t_0$, recorded by the controller's clock. The evolution of the gates belongs to the connected realization, not to the measurement call. That evolution is not observed to obtain intermediate binary values.

$$M_V(t_0) = (v_0, \ldots, v_{n-1}), \qquad v_i \in \{0, 1\}.$$

A repeat query returns the recorded readout and does not repeat the trigger. A detector error consumes that preparation. It is not replaced by another attempt or by an invented output.

Composition of internal modules

The construction plan must preserve a machine containing other V machines, with their relations prepared and variables not yet fixed:

$$V_t(V_1, \ldots, V_m), \qquad \mathbf{x}_i(t_i) \in \{0, 1\}^{r_i}.$$

Module inputs become fixed at their instant t_i and their values propagate through the connections. The translator must not replace a shared dependency with independent outputs or require internal values before preparing the module. Values declared as 0 or 1 and their consequences may propagate first.

If the protocol has a single trigger at $t = t_0$, internal fixing and propagation respond to that trigger without adding earlier external readouts. Preparing relations, executing the realization and observing its results are distinct operations. The trigger may initiate internal propagation in the realization: separating the interfaces does not forbid it. The author's rule is developed in section 20.3 of the main document. Executable status is maintained in the current results.

Portable organization

```
ES_52/
  ES_52_Maquina_V_UNIHOLOG.md
  ES_52_Laminas_Q_V.md
  img/
  Traductores_V/
    ES_52_Traductores_y_compiladores_de_V.md
    gestor_v.py
    traductor_qasm_v.py
    v_relacional.py
    operaciones_senales.py
    v1.py
    v2.py
    v3.py
    protocolo_q3.py
    ejecucion_v.py
    requirements.txt
    v.bat
    v.ps1
    v.sh
    MaquinasQ/
      par_h_cnot.q
    MaquinasV/
      par_h_cnot/
        v.py
```

```
        v.pl
        v.cpp
        v.hpp
        v.json
```

Each `MaquinasQ/*.q` file contains OpenQASM **2.0**, even though its extension is `.q`. The name without the extension identifies the machine. An example of two units with H and CNOT is included; not a physical realization of those gates.

The launchers calculate paths from their own location. The three documents are linked using relative paths. Moving **all of ES_52** preserves that navigation. The Traductores_V folder also works separately, but its links to the base document require the rest of the project.

Path portability does not install tools or hardware: the destination must have Python and Qiskit, and, depending on the language, SWI-Prolog or a C++ compiler compatible with the GCC/Clang interface and C++17.

Installation

With Python available, install the package dependencies:

```
python -m pip install -r requirements.txt
```

An interpreter can be selected using `V_PYTHON`, SWI-Prolog using `V_SWIPL` and a compiler using `V_CXX`. These variables contain the path to the executable, without additional arguments. By default,

installed tools are searched for. They are not downloaded or installed automatically.

The following commands are run from Traductores_V. From another folder, use the path to the launcher; machines are searched for in its MaquinasQ, not in the user's working directory.

Windows: PowerShell or BAT

```
.\v.ps1 listar
.\v.ps1 traducir par_h_cnot
.\v.ps1 compilar par_h_cnot --lenguaje python
.\v.ps1 compilar par_h_cnot --lenguaje prolog
.\v.ps1 compilar par_h_cnot --lenguaje cpp
```

The same usage is possible with `v.bat`, for example:

```
v.bat traducir par_h_cnot
```

The PowerShell execution policy may prevent opening the `.ps1`. That policy is not modified: the `.bat` can be used, or `python gestor_v.py` can be called directly with the same arguments.

Linux and other shell environments

```
sh ./v.sh listar
sh ./v.sh traducir par_h_cnot
sh ./v.sh compilar par_h_cnot --lenguaje python
sh ./v.sh compilar par_h_cnot --lenguaje prolog
sh ./v.sh compilar par_h_cnot --lenguaje cpp
```

Translation and compilation

Translation generates the five outputs in `MaquinasV/<nombre>/`. It does not overwrite an existing translation: it must be kept under another name, or the previous files must be explicitly removed to regenerate it. The JSON contains the plan, the measurement map and the input fingerprint, not internal states or a distribution of answers.

The `compilar` action checks Python syntax, loads the Prolog module or checks the C++ unit without linking. It does not prepare or trigger a V realization. Acceptance by a compiler is not an experimental result.

H, X, CNOT, Toffoli, SWAP, CZ, Rx, Rz, P/U1, S, inverse S, T, inverse T, Z and CU1/CP are supported when the parser accepts their declarations. Gates defined in QASM are expanded with their connections. Phases are not silently removed from their definitions.

Exactly one terminal measurement per unit is required, with distinct classical destinations. Intermediate or partial measurements, reset, control dependent on readouts, operations after a measurement and gates without a defined translation are rejected. OpenQASM 3, U/U2/U3, Y and Ry are not supported in this first version. These rejections describe limits of the tool, not a refutation of V.

Executing a connected realization

Execution requires a realization to be supplied. Without one, the launcher stops with an explicit message; it does not use a simulator or trial data as a substitute. The realization is trusted code that the user chooses to execute and must respect V's internal restrictions.

Python

The realization file defines `crear_realizacion()`. It returns an object with `preparar(plan)`, which configures without returning readouts, and `disparar(t0)`, which delivers one measured bit per unit, in unit index order.

```
.\v.ps1 ejecutar par_h_cnot --lenguaje python --realizacion
.\mi_realizacion.py
```

The report includes the readout, its classical bit map and the trigger instant. Unmeasured classical destinations are left without a readout (`null`); no result is invented for them.

Prolog

The realization file implements the multifile predicates `qasm_v:realizar_preparacion/1` and `qasm_v:realizar_disparo/2`. The controller records the wall clock when triggering; it does not confuse it with a calibration of the device's conversion instant.

```
.\v.ps1 ejecutar par_h_cnot --lenguaje prolog --realizacion
.\mi_realizacion.pl
```

C++

The realization file includes `v.hpp`, implements the `qasm_v::Realizacion` interface and defines the factory:

```
std::unique_ptr<qasm_v::Realizacion> crear_realizacion();
```

The manager generates a connection program, compiles it together with that realization and runs the binary. It does not introduce other gate dynamics. The realization includes its own resources and dependencies; this first version does not configure external hardware libraries.

```
.\v.ps1 ejecutar par_h_cnot --lenguaje cpp --realizacion
.\mi_realizacion.cpp
```

Evaluation and scope

The folder belongs to project 52 and is self-contained for translating and connecting realizations. DEMUESTRA remains outside as the shared manager: it uses `documentos_proyecto.json` to identify the main document and the annexes, generate their derivatives and check their paths. It does not mix the content of 42 or 45, or their evaluations, with this project.

From DEMUESTRA, local documents are updated with:

```
python publica.py --documentos-locales ES_52
```

This action generates HTML, PDF and a JSON report of sources and derivatives; it does not publish, modify Lean/Prolog evaluations or issue certificates. The `v.pl` program is an output of the V translator, not the Prolog certificate of the scientific document. Their functions and evaluations are distinct.

The controller labels a readout as `REALIZACION_NO_VERIFICADA`: it does not confer certification merely by receiving bits. The recorded time is that of the trigger invocation, not a guarantee of calibrated physical simultaneity. The realization's internal resources and dynamics must be reviewed separately.

Comparison with Q requires the same protocol and documentation of the corresponding conditions, outputs, repetitions, resources and times of both. Translating connections, compiling code and measuring a realization are distinct stages. No limitation of this manager becomes a negative conclusion about V, and no syntax test is presented as equivalence.

The parser used is the Qiskit OpenQASM 2 parser.