-ES_52: Traductores y compiladores de V

Autor: Víctor Estrada Díaz. Marco: Einstein-VED - UNIHOLOG.
Fecha: 10 de octubre de 2026.

Referencias del proyecto: Documento base · Láminas de construcción y comparación Q-V · Presentación a la comunidad · Índice completo de recursos.

Licencia Dual - Ética y Uso

- **Texto (CC BY-NC-SA 4.0)**: compartir, adaptar, no comercial, atribución y compartir igual.
- **Código (HL3)**: uso comercial permitido solo si cumple principios éticos. Prohibidos: guerras, violación de derechos humanos, daño ambiental, explotación, vigilancia masiva y discriminación.

Principios éticos declarados por el autor: respeto a la dignidad humana, no uso lucrativo sin permiso y prohibición de usos dañinos. Declaración · CC BY-NC-SA 4.0 · HL3.

Finalidad y reglas

Esta herramienta traduce un circuito Q conocido a un plan de construcción V. No consulta respuestas de Q ni calcula previamente un resultado para ocultarlo hasta la medida. Conserva puertas, controles,

conexiones y orden. No introduce un registro de amplitudes ni recorre posibles resultados.

V es la máquina determinista del autor, realizable con electrónica o software convencional. El desconocimiento antes de la medida no añade un mecanismo aleatorio a su dinámica.

En V, las salidas permanecen desconocidas antes de la medida. La lectura conjunta se desencadena en $t = t_0$, registrado por el reloj del controlador. La evolución de las puertas corresponde a la realización conectada, no a la llamada de medida. No se observa esa evolución para obtener valores binarios intermedios.

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

Una reconsulta devuelve la lectura registrada y no repite el disparo. Un error del detector consume esa preparación. No se reemplaza por otro intento ni por una salida inventada.

Composición de módulos internos

El plan de construcción debe conservar una máquina que contiene otras máquinas V, con sus relaciones preparadas y sus variables aún no fijadas:

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

Las entradas del módulo quedan fijadas en su instante t_i y sus valores se propagan por las conexiones. El traductor no debe sustituir una dependencia compartida por salidas independientes ni exigir los valores internos antes de preparar el módulo. Los valores declarados como 0 o 1 y sus consecuencias pueden propagarse primero.

Si el protocolo tiene un único disparo en $t = t_0$, la fijación y propagación internas responden a ese disparo, sin añadir lecturas externas anteriores. Preparar relaciones, ejecutar la realización y observar sus resultados son operaciones distintas. El disparo sí puede desencadenar la propagación interna de la realización: separar las interfaces no la prohíbe. La regla del autor está desarrollada en el apartado 20.3 del documento principal. El estado ejecutable se mantiene en los resultados actuales.

Organización portátil

```
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
```

Cada archivo `MaquinasQ/*.q` contiene OpenQASM **2.0**, aunque su extensión sea `.q`. El nombre sin extensión identifica la máquina. Se incluye un ejemplo de dos unidades con H y CNOT; no una realización física de esas puertas.

Los lanzadores calculan las rutas desde su propia ubicación. Los tres documentos se enlazan mediante rutas relativas. Al trasladar **todo ES_52** se conserva esa navegación. La carpeta Traductores_V también funciona por separado, pero sus enlaces al documento base requieren el resto del proyecto.

La portabilidad de rutas no instala herramientas ni hardware: el destino debe disponer de Python y Qiskit, y, según el lenguaje, SWI-Prolog o un compilador C++ compatible con la interfaz de GCC/Clang y C++17.

Instalación

Con Python disponible, instalar las dependencias del paquete:

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

Puede seleccionarse un intérprete mediante `V_PYTHON`, SWI-Prolog mediante `V_SWIPL` y un compilador mediante `V_CXX`. Estas variables contienen la ruta al ejecutable, sin argumentos adicionales. Por defecto

se buscan las herramientas instaladas. No se descargan ni instalan automáticamente.

Los comandos siguientes se ejecutan desde Traductores_V. Desde otra carpeta se utiliza la ruta al lanzador; las máquinas se buscan en su MaquinasQ, no en el directorio de trabajo del usuario.

Windows: PowerShell o 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
```

El mismo uso es posible con `v.bat`, por ejemplo:

```
v.bat traducir par_h_cnot
```

La política de ejecución de PowerShell puede impedir abrir el `.ps1`. No se modifica dicha política: puede utilizarse el `.bat` o llamar directamente a `python gestor_v.py` con los mismos argumentos.

Linux y otros entornos con shell

```
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
```

Traducción y compilación

La traducción genera las cinco salidas en `MaquinasV/<nombre>/`. No sobrescribe una traducción existente: se debe conservarla con otro nombre o retirar explícitamente los archivos anteriores para regenerarla. El JSON contiene el plan, el mapa de medida y la huella de la entrada, no estados internos ni una distribución de respuestas.

La acción `compilar` comprueba la sintaxis Python, carga el módulo Prolog o comprueba la unidad C++ sin enlazar. No prepara ni dispara una realización V. La aceptación de un compilador no es un resultado experimental.

Se admiten H, X, CNOT, Toffoli, SWAP, CZ, Rx, Rz, P/U1, S, S inversa, T, T inversa, Z y CU1/CP cuando el analizador admite sus declaraciones. Las puertas definidas en QASM se expanden con sus conexiones. No se eliminan fases de sus definiciones de forma silenciosa.

Se requiere exactamente una medida terminal por unidad, con destinos clásicos distintos. Se rechazan medidas intermedias o parciales, reset, control dependiente de lecturas, operaciones posteriores a una medida y puertas sin traducción definida. OpenQASM 3, U/U2/U3, Y y Ry no se admiten en esta primera versión. Estos rechazos describen límites de la herramienta, no una refutación de V.

Ejecutar una realización conectada

La ejecución exige aportar una realización. Sin ella, el lanzador se detiene con un mensaje explícito; no usa un simulador ni datos de ensayo como sustituto. La realización es código confiable que el usuario decide ejecutar y debe respetar las restricciones internas de V.

Python

El archivo de realización define `crear_realizacion()`. Devuelve un objeto con `preparar(plan)`, que configura sin devolver lecturas, y `disparar(t0)`, que entrega un bit medido por unidad, en orden de índice de unidad.

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

El informe incluye la lectura, su mapa de bits clásicos y el instante de disparo. Los destinos clásicos no medidos se dejan sin lectura (`null`), no se inventa un resultado para ellos.

Prolog

El archivo de realización implementa los predicados multifile `qasm_v:realizar_preparacion/1` y `qasm_v:realizar_disparo/2`. El controlador registra el reloj civil al disparar; no lo confunde con una calibración del instante de conversión del dispositivo.

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

C++

El archivo de realización incluye `v.hpp`, implementa la interfaz `qasm_v::Realizacion` y define la fábrica:

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

El gestor genera un programa de conexión, lo compila junto a esa realización y ejecuta el binario. No introduce otra dinámica de puertas. La realización incluye sus propios recursos y dependencias; esta primera versión no configura bibliotecas externas de hardware.

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

Evaluación y alcance

La carpeta pertenece al proyecto 52 y es autónoma para traducir y conectar realizaciones. DEMUESTRA permanece fuera como gestor común: utiliza `documentos_proyecto.json` para identificar el documento principal y los anexos, generar sus derivados y comprobar sus rutas. No mezcla el contenido del 42, del 45 ni sus evaluaciones con este proyecto.

Desde DEMUESTRA se actualizan los documentos locales con:

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

Esta acción genera HTML, PDF y un informe JSON de fuentes y derivados; no publica, no modifica evaluaciones Lean/Prolog ni emite certificados. El programa `v.pl` es una salida del traductor V, no el certificado Prolog del documento científico. Sus funciones y sus evaluaciones son distintas.

El controlador etiqueta una lectura como `REALIZACION_NO_VERIFICADA`: no atribuye certificación por el mero hecho de recibir bits. El momento registrado es el de la invocación del disparo, no una garantía de simultaneidad física calibrada. Los recursos y la dinámica internos de la realización deben revisarse por separado.

La comparación con Q exige el mismo protocolo y documentar las condiciones, salidas, repeticiones, recursos y tiempos correspondientes de ambas. Traducir conexiones, compilar código y medir una realización son etapas distintas. Ninguna limitación de este gestor se convierte en una conclusión negativa sobre V, y ninguna prueba de sintaxis se presenta como equivalencia.

El analizador empleado es el de OpenQASM 2 de Qiskit.