-FR_52: Traducteurs et compilateurs de V

Auteur : Víctor Estrada Díaz. Cadre : Einstein-VED - UNIHOLOG.
Date : 10 octobre 2026.

Références du projet : Document de base · Planches de construction et de comparaison Q-V · Présentation à la communauté · Index complet des ressources.

Double licence - Éthique et utilisation

- **Texte (CC BY-NC-SA 4.0)** : partage, adaptation, utilisation non commerciale, attribution et partage dans les mêmes conditions.
- **Code (HL3)** : utilisation commerciale autorisée uniquement si elle respecte les principes éthiques. Sont interdits : guerres, violations des droits humains, dommages à l'environnement, exploitation, surveillance de masse et discrimination.

Principes éthiques déclarés par l'auteur : respect de la dignité humaine, pas d'utilisation lucrative sans autorisation et interdiction des utilisations nuisibles. Déclaration · CC BY-NC-SA 4.0 · HL3.

Objectif et règles

Cet outil traduit un circuit Q connu en un plan de construction V. Il ne consulte pas les réponses de Q et ne calcule pas un résultat à l'avance pour le cacher jusqu'à la mesure. Il conserve les portes, les contrôles, les connexions et l'ordre. Il n'introduit pas de registre d'amplitudes et ne parcourt pas les résultats possibles.

V est la machine déterministe de l'auteur, réalisable avec de l'électronique ou du logiciel conventionnels. L'ignorance avant la mesure n'ajoute pas de mécanisme aléatoire à sa dynamique.

Dans V, les sorties restent inconnues avant la mesure. La lecture conjointe est déclenchée à $t = t_0$, enregistré par l'horloge du contrôleur. L'évolution des portes relève de la réalisation connectée, et non de l'appel de mesure. Cette évolution n'est pas observée pour obtenir des valeurs binaires intermédiaires.

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

Une nouvelle consultation renvoie la lecture enregistrée sans répéter le déclenchement. Une erreur du détecteur consomme cette préparation. Elle n'est remplacée ni par une autre tentative ni par une sortie inventée.

Composition des modules internes

Le plan de construction doit conserver une machine contenant d'autres machines V, avec leurs relations préparées et variables non encore fixées :

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

Les entrées du module sont fixées à leur instant t_i et leurs valeurs se propagent par les connexions. Le traducteur ne doit pas remplacer une dépendance partagée par des sorties indépendantes ni exiger les valeurs internes avant de préparer le module. Les valeurs déclarées comme 0 ou 1 et leurs conséquences peuvent se propager d'abord.

Si le protocole a un déclenchement unique en $t = t_0$, la fixation et la propagation internes répondent à ce déclenchement sans ajouter de lectures externes antérieures. Préparer les relations, exécuter la réalisation et observer ses résultats sont des opérations distinctes. Le déclenchement peut initier la propagation interne de la réalisation : séparer les interfaces ne l'interdit pas. La règle de l'auteur est développée dans la section 20.3 du document principal. L'état exécutable est maintenu dans les résultats actuels.

Organisation portable

```
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
```

Chaque fichier `MaquinasQ/*.q` contient de l'OpenQASM **2.0**, bien que son extension soit `.q`. Le nom sans extension identifie la machine. Un exemple de deux unités avec H et CNOT est inclus ; ce n'est pas une réalisation physique de ces portes.

Les lanceurs calculent les chemins à partir de leur propre emplacement. Les trois documents sont reliés par des chemins relatifs. Déplacer **l'ensemble de ES_52** préserve cette navigation. Le dossier Traductores_V fonctionne également séparément, mais ses liens vers le document de base nécessitent le reste du projet.

La portabilité des chemins n'installe ni outils ni matériel : la destination doit disposer de Python et de Qiskit, et, selon le langage, de SWI-Prolog ou d'un compilateur C++ compatible avec l'interface de GCC/Clang et C++17.

Installation

Avec Python disponible, installer les dépendances du paquet :

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

Un interpréteur peut être sélectionné au moyen de `V_PYTHON`, SWI-Prolog au moyen de `V_SWIPL` et un compilateur au moyen de `V_CXX`. Ces variables contiennent le chemin vers l'exécutable, sans arguments supplémentaires. Par défaut, on recherche les outils installés. Ils ne sont ni téléchargés ni installés automatiquement.

Les commandes suivantes sont exécutées depuis Traductores_V. Depuis un autre dossier, on utilise le chemin vers le lanceur ; les machines sont recherchées dans son MaquinasQ, et non dans le répertoire de travail de l'utilisateur.

Windows : PowerShell ou 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
```

La même utilisation est possible avec `v.bat`, par exemple :

```
v.bat traducir par_h_cnot
```

La politique d'exécution de PowerShell peut empêcher l'ouverture du `.ps1`. Cette politique n'est pas modifiée : on peut utiliser le `.bat` ou appeler directement `python gestor_v.py` avec les mêmes arguments.

Linux et autres environnements avec 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
```

Traduction et compilation

La traduction génère les cinq sorties dans `MaquinasV/<nombre>/`. Elle n'écrase pas une traduction existante : il faut la conserver sous un autre nom ou retirer explicitement les fichiers précédents pour la régénérer. Le JSON contient le plan, la correspondance des mesures et l'empreinte de l'entrée, et non des états internes ou une distribution de réponses.

L'action `compilar` vérifie la syntaxe Python, charge le module Prolog ou vérifie l'unité C++ sans édition de liens. Elle ne prépare ni ne déclenche une réalisation V. L'acceptation par un compilateur n'est pas un résultat expérimental.

H, X, CNOT, Toffoli, SWAP, CZ, Rx, Rz, P/U1, S, l'inverse de S, T, l'inverse de T, Z et CU1/CP sont pris en charge lorsque l'analyseur accepte leurs déclarations. Les portes définies en QASM sont développées avec leurs connexions. Les phases de leurs définitions ne sont pas supprimées silencieusement.

Exactement une mesure terminale par unité est exigée, avec des destinations classiques distinctes. Sont rejetés les mesures intermédiaires ou partielles, reset, le contrôle dépendant des lectures, les opérations postérieures à une mesure et les portes sans traduction définie. OpenQASM 3, U/U2/U3, Y et Ry ne sont pas pris en charge dans cette première version. Ces rejets décrivent des limites de l'outil, et non une réfutation de V.

Exécuter une réalisation connectée

L'exécution exige de fournir une réalisation. Sans elle, le lanceur s'arrête avec un message explicite ; il n'utilise ni simulateur ni données d'essai comme substitut. La réalisation est du code de confiance que l'utilisateur choisit d'exécuter et doit respecter les restrictions internes de V.

Python

Le fichier de réalisation définit `crear_realizacion()`. Il renvoie un objet avec `preparar(plan)`, qui configure sans renvoyer de lectures, et `disparar(t0)`, qui fournit un bit mesuré par unité, dans l'ordre des indices d'unité.

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

Le rapport comprend la lecture, sa correspondance avec les bits classiques et l'instant de déclenchement. Les destinations classiques non mesurées restent sans lecture (`null`) ; aucun résultat n'est inventé pour elles.

Prolog

Le fichier de réalisation implémente les prédicats multifile `qasm_v:realizar_preparacion/1` et `qasm_v:realizar_disparo/2`. Le contrôleur enregistre l'horloge civile lors du déclenchement ; il ne la confond pas avec un étalonnage de l'instant de conversion du dispositif.

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

C++

Le fichier de réalisation inclut `v.hpp`, implémente l'interface `qasm_v::Realizacion` et définit la fabrique :

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

Le gestionnaire génère un programme de connexion, le compile avec cette réalisation et exécute le binaire. Il n'introduit pas d'autre dynamique des portes. La réalisation inclut ses propres ressources et dépendances ; cette première version ne configure pas de bibliothèques matérielles externes.

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

Évaluation et portée

Le dossier appartient au projet 52 et est autonome pour traduire et connecter des réalisations. DEMUESTRA reste à l'extérieur comme gestionnaire commun : il utilise `documentos_proyecto.json` pour identifier le document principal et les annexes, générer leurs dérivés et vérifier leurs chemins. Il ne mélange pas le contenu du 42 ou du 45, ni leurs évaluations, avec ce projet.

Depuis DEMUESTRA, les documents locaux sont mis à jour avec :

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

Cette action génère du HTML, du PDF et un rapport JSON des sources et des dérivés ; elle ne publie pas, ne modifie pas les évaluations Lean/Prolog et n'émet pas de certificats. Le programme `v.pl` est une sortie du traducteur V, et non le certificat Prolog du document scientifique. Leurs fonctions et leurs évaluations sont distinctes.

Le contrôleur étiquette une lecture comme `REALIZACION_NO_VERIFICADA` : il n'attribue pas de certification du seul fait de recevoir des bits. L'instant enregistré est celui de l'invocation du déclenchement, et non une garantie de simultanéité physique étalonnée. Les ressources et la dynamique internes de la réalisation doivent être examinées séparément.

La comparaison avec Q exige le même protocole et la documentation des conditions, sorties, répétitions, ressources et temps correspondants des deux machines. Traduire des connexions, compiler du code et mesurer une réalisation sont des étapes distinctes. Aucune

limitation de ce gestionnaire ne se transforme en une conclusion négative sur V, et aucun test de syntaxe n'est présenté comme une équivalence.

L'analyseur employé est celui d' OpenQASM 2 de Qiskit.