-FR_52: Résultats de préparation, d'exécution et de comparaison des machines V

Document principal FR_52

Auteur du projet : Víctor Estrada Díaz · Einstein–VED · UNIHOLOG.

Mise à jour locale : révision des 18–19 octobre 2026. Ce document décrit l'état actuel des programmes et de leurs rapports, et non des évaluations antérieures.

1. Règles de préparation et de lecture

Il faut assurer la cohérence : l'état intérieur de V doit être garanti inconnu et toutes les entrées et sorties de qbit doivent être inconnues et indépendantes, sauf celles définies comme 0 ou 1. Tout est préparé pour que, à partir de l'inconnu, l'effondrement se produise lors de la mesure en $t = t_0$.

On propage d'abord les valeurs expressément fixées et leurs conséquences logiques. On conserve ensuite les relations entre signaux inconnus sans leur attribuer de résultats anticipés. Connaître la relation ne revient pas à connaître les valeurs d'une exécution préparée.

Dans ce protocole, on n'exige ni de Q ni de V de lire ou de déterminer sa phase interne avant de mesurer. Les transformations de signaux sont des opérations internes, et non des lectures de l'observateur. Les tests préalables d'une formule avec des valeurs d'essai ne sont pas des observations de la machine déjà préparée.

L'exécution de V ne reçoit pas de résultats de Q. Les rapports de V sont d'abord enregistrés ; la comparaison avec la référence Q s'exécute ensuite, au moyen d'un autre programme et uniquement pour les cas dont la mesure est enregistrée.

V est ici implémentée au moyen d'un logiciel conventionnel déterministe. On n'utilise ni V_ref, ni registre global d'amplitudes, ni énumération de trajectoires, ni échantillonneur de résultats Q pour produire sa sortie.

Composition interne précisée par l'auteur

Une machine V peut contenir des modules qui sont d'autres machines V :

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

Les relations sont préparées sans connaître les variables non fixées. Lorsque les entrées du module sont fixées à t_i, ces mêmes valeurs se propagent en conservant les dépendances entre modules. Les valeurs déjà déclarées comme 0 ou 1 et leurs conséquences se propagent d'abord.

Dans le protocole à déclenchement unique en $t = t_0$, la propagation interne répond à ce déclenchement sans introduire d'observations externes antérieures. Cette précision résout le critère de composition demandé pour V2 ; aucune préparation différente de celle définie par l'auteur n'est demandée. L'implémentation et sa vérification restent documentées séparément. V1 reste établie ; V2 et V3 sont exécutées et vérifiées séparément.

Vf : test indépendant et précondition de V2

Règle de construction de V : la statistique ne fait pas partie du circuit. Aucun état ni aucune sortie n'est produit par des distributions, estimateurs, histogrammes ou tirages aléatoires. Les prédictions et comparaisons éventuelles sont externes et ne conditionnent pas l'exécution. L'ignorance préalable n'est pas remplacée par un échantillonneur. Cette règle figure aussi dans les instructions locales de construction du projet.

Le module Vf reçoit des entrées Q_vbits connectées et leurs relations, et non un tuple de bits déjà fixés par l'adaptateur. Pendant la préparation, il ne consulte pas leurs valeurs et n'évalue aucun signal. Vf fixe ses entrées dans `colapsar(t0)`, avec le même déclenchement extérieur ; les valeurs se propagent seulement ensuite. Il n'ajoute ni horloge, ni tirage aléatoire, ni mesure indépendante. Une sortie achevée est conservée sans répéter l'effondrement.

La précondition de Vf comprend six tests propres, indépendants de V2. Ils vérifient la préparation, le blocage des lectures, l'instant commun, la sortie conservée et plusieurs largeurs du module. Les huit tests de connexion font l'objet d'un rapport distinct. Ces tests ne sont pas additionnés pour constituer une certification de V2.

Vf reçoit des Q_vbits directs ou liés et les fixe uniquement dans son déclenchement. Il évalue une fois le graphe connecté depuis cette capture commune, conserve la valeur de chaque nœud et applique le câblage Fourier aux valeurs obtenues sur ses ports. Une source partagée conserve la même valeur : aucun nouveau résultat n'est choisi dans chaque module. Les paires de signaux arrivent au comparateur avec les références du même événement. Il s'agit d'une propagation déterministe de valeurs capturées, et non, par définition, d'une évolution quantique conjointe.

Avant de composer V2, le rapport de Vf est vérifié par rapport au code actuel. V2 utilise ensuite le contrat d'entrée, de transformation et de sortie du module. L'hypothèse plus large d'équivalence avec Fourier Q reste distincte de ce protocole vérifié : elle n'est pas démontrée par le seul fait que les entrées soient inconnues avant le déclenchement. Cette équivalence n'est pas non plus une précondition pour construire ou exécuter V.

V2 : connexion et diagnostic de la composition avec Vf

La réalisation composée relie les relations modulaires aux Q_vbits de Vf et conserve le comparateur de sortie. Le déclenchement commun atteint Vf avant toute consultation des références. Elle conserve le plan installé de V2, sans registre global d'amplitudes ni résultat Q introduit dans l'exécution.

Les résultats V contiennent les deux exécutions complètes avec horloge réelle et le profil déclaré. Le profil configure des références déterministes, et non un mécanisme statistique ou une condition statistique d'exécution. Les deux cas indiquent `MEDIDO`, avec les sources capturées, le graphe, les valeurs internes et

la sortie. `test_v2.py` utilise désormais cette réalisation par défaut ; l'ancien blocage pour dépendances partagées n'est pas conservé. La comparaison ultérieure est exécutée après l'enregistrement des sorties, sans les réinjecter dans V. Ses résultats, y compris les désaccords, restent dans le rapport. Construction, exécution et comparaison sont distinctes : `MEDIDO` ne démontre pas à lui seul l'équivalence avec Q2.

2. Comparateur connecté au latch

Avec l'autorisation de l'auteur, la lecture de la section 12.2 a été connectée :

$$w = \frac{|s_0|^2}{|s_0|^2 + |s_1|^2}, \qquad r(t) = \text{frac}\left(\frac{\omega t + \eta}{2\pi}\right),$$

$$b(t_0) = \begin{cases} 0, & r(t_0) < w, \\ 1, & r(t_0) \geq w. \end{cases}$$

Le comparateur temporel implémente cette règle sans générer de nombres aléatoires. Deux signaux nuls invalident la lecture ; aucun résultat arbitraire ne leur est attribué. L'égalité avec le seuil appartient à la branche qui fournit 1.

La référence logicielle a pour configuration par défaut $\omega = 2\pi$ radians par seconde et $\eta = 0$. C'est une référence périodique déterministe. La configuration et la référence effectivement capturée figurent dans le rapport. Le seuil est calculé avec la formule déclarée : il n'est pas estimé à partir de fréquences et la sortie n'est pas tirée au hasard. La discussion probabiliste de la section 12.2 concerne des prédictions externes ; elle n'est ni un composant ni une précondition d'exécution de V.

CircuitoSenalesV prépare et exécute les transformations de signaux sans renvoyer leurs valeurs à l'appelant. H, phase, RX et RZ utilisent les formules de operaciones_senales.py. Les connexions CNOT et de phase entre unités sont exécutées dans les cas que le composant peut réaliser sans perdre les relations conjointes. Les phases diagonales en fin de circuit sont conservées comme relations de branche, sans construire de registre d'amplitudes conjointes. Cette règle ne s'applique pas si une H ou RX mélange ensuite les branches. Une connexion générale qui n'est pas encore implémentée entraîne un arrêt explicite : elle n'est pas remplacée par deux unités indépendantes.

Dans V1, la cible de l'oracle conserve la relation de signaux correspondant à la valeur propre d'inversion −1. La CNOT applique ce signe au canal de contrôle correspondant, en conservant la cible. C'est une application algébrique de la connexion, et non une mesure intermédiaire ni une classification de l'oracle fournie comme réponse.

Le latch déclenche une capture conjointe. L'instant t_0 provient de l'horloge monotone de cette exécution, et non d'une étiquette temporelle fournie pour choisir le résultat. Le comparateur fixe les bits des sources et ces mêmes valeurs sont propagées à travers les relations binaires finales jusqu'à déterminer les sorties. On n'effectue pas de nouvelle lecture à chaque nœud et on ne choisit pas un bit indépendant pour chaque sortie. Les relations déjà fixées par les entrées sont simplifiées au préalable.

Les relations finales sont préparées avec propagacion_conocidos.py. `resolver_captura` accepte uniquement un tuple immuable de bits capturés ; il rejette une source différée sans la lire. Il rejette également une capture qui contredit une valeur expressément fixée. La résolution ne consulte pas Q, ne consulte pas à nouveau l'horloge et n'accède pas aux signaux.

Cette implémentation applique la propagation aux relations binaires finales. Elle ne déplace pas la capture de l'autre côté d'une H intermédiaire pour contourner un couplage de signaux restant à implémenter. Le composant générique prépare son préfixe de signaux une fois ; l'exécuteur différé de V3 reporte aussi ce calcul au déclenchement. Les nœuds des relations préparées sont résolus. Ce travail est inclus dans le temps de mesure et n'est pas déclaré à coût constant.

La capture conserve les bits de source, les valeurs propagées des nœuds et les bits de sortie. L'audit enregistre également les composantes réelles et imaginaires des signaux du préfixe qui alimente ces relations, le seuil et la référence au même instant logique. Les interfaces de lecture et d'audit restent bloquées avant le déclenchement. Une nouvelle consultation renvoie l'enregistrement existant et ne répète pas la mesure. Une capture logicielle cohérente n'est pas présentée comme une mesure de simultanéité physique du matériel.

Le composant actuel ne réutilise pas une référence commune comme si elle justifiait plusieurs lectures inconnues indépendantes. S'il a besoin de plus d'une lecture non fixée, il accepte des références expressément connectées par port. Il refuse qu'une seule instance soit présentée comme plusieurs sources. L'existence de références distinctes ne certifie pas leur indépendance. Sans ces connexions, il indique celles qui manquent. Cette limitation appartient à ce composant ; elle n'est pas ajoutée comme prémisse universelle de V.

3. Données, programmes et résultats actuels

Machine	Programme	Entrées	Rapport
V1	test_v1.py	data_v1.json	resultados_v1.json
V2	test_v2.py	data_v2.json	resultados_v2.json
V3	test_v3.py	data_v3.json	resultados_v3.json

Les trois programmes utilisent pruebas_v.py. V2 utilise par défaut la composition avec Vf et son profil de lecture déclaré. V1 conserve son circuit ; V3 utilise l'exécuteur différé et le même comparateur. Chaque cas prépare une nouvelle machine et demande une unique mesure. Les données contiennent les entrées du cas, et non les réponses de Q. Les rapports conservent les entrées, les empreintes SHA-256, les temps de préparation et de mesure, l'instant enregistré et la capture, lorsque celle-ci est achevée.

V1 : résultat établi pour la tâche de Deutsch

Le résultat est établi par l'argument exact de la section 7 et le rapport vérifiable, en complément des quatre mesures enregistrées ci-dessous.

Ordre des bits : `[q0, q1]`, q0 étant le moins significatif. La comparaison du protocole de Deutsch utilise le premier bit, q0.

Entrée de l'oracle	Table d'entrée	q0 obtenu	q0 prévu par la référence Q
Constante 0	`[0,0]`	0	0

Entrée de l'oracle	Table d'entrée	q0 obtenu	q0 prévu par la référence Q
Constante 1	`[1,1]`	0	0
Identité	`[0,1]`	1	1
Inverseur	`[1,0]`	1	1

Les quatre premiers bits enregistrés concordent avec la référence Q du protocole. Le calcul Q ultérieur attribue à chacun une probabilité de 0.9999999999999996, équivalente à 1 dans les limites de la précision numérique utilisée. Les valeurs de q1 et la capture complète de chaque exécution figurent dans le rapport JSON ; elles peuvent changer si l'on répète le déclenchement à un autre instant.

L'égalité de ces quatre premiers bits n'est pas utilisée pour affirmer une distribution conjointe complète ni une indépendance entre préparations. La sortie q1 et la qualité statistique de la référence nécessitent leurs propres vérifications. Aucun avantage temporel par rapport à du matériel Q n'a non plus été mesuré.

V2 et V3 : règles auditées et exécutions par défaut

L'audit des règles enregistre **14 tests réussis** de V2 et V3. La préparation ne consulte pas les sources et ne fixe pas les sorties inconnues. Une seule horloge externe enregistre le déclenchement et tous les modules utilisent ce même instant. La propagation ne relit pas les sources ; une lecture ultérieure renvoie la capture enregistrée et un échec n'autorise pas une nouvelle tentative sur la même préparation.

V2 conserve son graphe partagé et Vf fixe les entrées Q_vbits connectées dans son propre déclenchement. Dans V3 différée, la préparation conserve uniquement constantes et connexions : le calcul des signaux, la résolution et la lecture sont exécutés une fois dans cet événement. Le profil V3 déclare les références périodiques déterministes des trois ports. Aucune sortie n'est produite par probabilités, tirages aléatoires ou statistiques ; le seuil du comparateur est une opération déterministe déjà déclarée.

Les tests vérifient le protocole d'accès du logiciel et les connexions, pas l'isolation physique contre l'inspection ni l'indépendance statistique de sources déterministes. Cacher une variable Python ne démontre pas ces propriétés. La comparaison Q reste externe et ultérieure.

Machine et entrée	État	Point d'arrêt
V2, $N = 15$, base 2	`MEDIDO`	Composition modulaire, Vf et sortie achevée
V2, $N = 21$, base 2	`MEDIDO`	Composition modulaire, Vf et sortie achevée
V3, trois unités, H-CZ-RZ	`MEDIDO`	Préparation différée, références déclarées et capture commune ; phases terminales conservées

V2 et V3 enregistrent leurs bits et permettent la comparaison externe ultérieure. V3 n'utilise plus par défaut le composant sans références connectées. L'essai de phases décrit ci-dessous conserve sa portée propre ; il n'est pas confondu avec l'exécution par défaut ni ajouté comme une preuve globale.

V2 : révision des blocs et de la taille des connexions

verificar_v2.py a été exécuté sans consulter Q. Le rapport est verificacion_v2.json. Les données originales et les connexions déclarées de V2 sont conservées.

Entrée	Unités	Multiplications contrôlées	Portes du plan complet	Captures de blocs avec entrées fixées	Chaînes modulaires vérifiées
$N = 15$, base 2	12	8	56	256	6
$N = 21$, base 2	15	10	80	640	6

Il s'agit de **896 captures de composants et de 12 chaînes de multiplications**, et non de 908 mesures de V2 complète. Les deux contrôles, tous les petits mots de travail de ces tests et la conservation des mots hors du module ont été vérifiés. Ces tables d'essai ne font pas partie du fonctionnement interne de V. La chaîne calcule la relation déclarée $y \mapsto a^x y \bmod N$ sans recevoir de résultat Q.

Avec $b = \ bit_length(N)$, V2 prépare $3b$ unités, $2b$ blocs modulaires et une Fourier inverse sur $2b$ unités. Son plan compilé contient exactement :

$$g(b) = 2b^2 + 6b\,.$$

Ce décompte a été vérifié pour sept tailles, entre 2 et 64 bits du module. C'est une borne quadratique du nombre de connexions, et non une preuve du temps total d'exécution d'une réalisation encore incomplète. L'arithmétique des blocs modulaires a également un coût dépendant de la longueur de ses opérandes.

L'adaptateur générique de signaux, qui n'est plus la réalisation par défaut de V2, s'arrête à la première multiplication contrôlée : opération d'indice 8 pour $N = 15$ et d'indice 10 pour $N = 21$, en comptant à partir de 0. Le rapport conserve désormais l'indice, les ports, le multiplicateur et le module exacts, ainsi que le temps de préparation consommé. Ce qui manque dans ce composant, c'est la connexion de la relation conjointe de signaux du contrôle non lu et du travail avec la Fourier ultérieure. **Il ne manque pas de latch, et il n'est pas demandé de connaître la phase avant de mesurer.** Ce couplage n'a pas été remplacé par une mesure anticipée, une recherche de la période ou un registre global d'amplitudes.

V2 : composition modulaire avec contrôles non encore lus

verificar_puente_v2.py compose les relations modulaires existantes et les SWAP d'entrée de la Fourier, sans attribuer de valeurs aux contrôles. Le rapport du pont conserve le graphe des connexions : 20 nœuds pour N=15 et 72 pour N=21, sans énumérer les états conjoints. Les valeurs d'essai servent uniquement aux vérifications du composant, pas à lire une V2 préparée.

Pour N=15, on déduit que six H de la Fourier agissent sur des facteurs libres et fixent leur sortie à zéro ; leurs phases contrôlées avec un port nul sont sans effet. La première H nécessitant les signaux liés au travail a l'indice 47 dans le plan complet. Pour N=21, elle a l'indice 25. Les indices commencent à zéro. Ce

sont des déductions des connexions, pas des mesures anticipées de la phase ou des contrôles.

Cette vérification analyse l'adaptateur de signaux, et non l'exécution de la nouvelle composition avec Vf. Ses indices et déductions ne sont pas présentés comme sorties de V2. Les captures actuelles figurent dans les rapports d'exécution cités plus haut. L'exécution différée de V3 a ses propres tests.

V3 : phases finales connectées et capture d'essai

verificar_v3.py a été exécuté. Le rapport verificacion_v3.json conserve quatre branches avec références d'essai, une capture avec horloge réelle et les empreintes de son code et de ses données originales.

La CZ finale n'est pas remplacée par deux phases locales. La relation conjointe de phase est conservée ; pour les bits capturés du protocole, elle est :

$$d(b_0, b_1, b_2) = \exp(i\pi b_0 b_1 + i\frac{\pi}{2}(b_2 - \frac{1}{2})).$$

La justification pour une phase terminale diagonale dans la base de lecture est :

$$|d(x)| = 1, \qquad |d(x)a_x|^2 = |a_x|^2 .$$

Cette égalité algébrique ne nécessite pas d'énumérer les valeurs de x dans le programme. La phase et ses ports restent préparés comme une relation ; le rapport conserve son facteur pour la branche capturée. **Cette égalité n'est pas utilisée pour effacer une phase avant une H ou RX ultérieure.** Le test de régression vérifie expressément cette limite.

Les quatre branches vérifiées avec des entrées de comparateur d'essai sont `(0,0,0)`, `(0,1,0)`, `(1,0,0)` et `(1,1,0)`. On vérifie également que la phase est évaluée sur les sorties propagées, et non sur des bits de source qu'une CNOT aurait pu changer. L'angle d'une CP terminale est également conservé. Aucune référence ni l'horloge ne sont consultées avant le déclenchement ; les relectures ne produisent pas de seconde capture.

L'essai avec horloge réelle utilise deux références périodiques déclarées :

$$\omega_0 = 2\pi, \qquad \omega_1 = 2\pi\sqrt{2}, \qquad \eta_0 = \eta_1 = 0 .$$

La capture concrète, son t_0 et les valeurs de référence figurent dans le rapport. Cette configuration est un **test de connexion et de lecture**, et non une déclaration d'indépendance ou d'uniformité démontrées. La configuration n'est pas changée silencieusement : l'exécuteur par défaut actuel et son profil sont identifiés plus haut. Une seule capture ne suffit pas à déclarer toute la préparation de cohérence satisfaite. Q est consulté uniquement après l'enregistrement des rapports V.

Pour la famille de H locales suivies exclusivement de phases CZ terminales, 2, 8, 32 et 128 unités ont été testées. Deux signaux par unité et une relation par phase sont enregistrés, et non 2^n états. À précision fixe, la préparation et la résolution de cette famille coûtent $O(n + g)$ en temps et en mémoire, avec g phases et leurs connexions. Le test utilise des références d'essai identifiées ; il ne démontre pas la qualité statistique d'une source physique et n'étend pas la borne aux circuits arbitraires avec interférence.

4. Comparaison ultérieure avec Q

comparar_resultados.py s'exécute après l'enregistrement des trois rapports. Il vérifie que chacun correspond à son fichier d'entrée et au code actuel. Il rejette les résultats dont les empreintes de code sont obsolètes, au lieu de les réutiliser comme actuels. Il conserve l'empreinte du rapport V utilisé. Le résultat se trouve dans comparacion_maquinas.json.

Bilan des exécutions par défaut : **7 cas V, 7 mesures achevées, aucune connexion en attente et 7 comparaisons ultérieures**. Il existe en outre deux comparaisons d'essais séparés : H-CNOT et V3 avec références déclarées non validées comme indépendantes. Ces essais ne sont pas comptés comme des exécutions par défaut achevées. La comparaison de V1 est effectuée avec la référence Q locale existante dans `DEMUESTRA/comparacion/q1_v1.py`, sans la modifier ni l'exécuter à l'intérieur de V. C'est un calcul de référence Qiskit, et non une mesure d'une machine Q physique. Son empreinte est conservée.

Les modules de référence sont chargés uniquement pour les cas V mesurés. La référence ne peut pas fournir de réponse au processus V, qui est déjà terminé et a enregistré son rapport avant l'exécution de ce vérificateur.

Circuit existant H-CNOT : propagation conjointe

Le programme portable existant MaquinasV/par_h_cnot/v.py a également été exécuté, sans changer ses connexions H et CNOT ni le remplacer par un détecteur. Avec les deux entrées préparées à 0, la relation finale est `salida_1 = salida_0` : une seule valeur non fixée détermine les deux sorties.

L'exécution avec horloge réelle enregistre sa sortie concrète dans le rapport, toujours avec les deux bits égaux. Deux tests préalables avec horloge contrôlée ont vérifié aussi bien `(0,0)` que `(1,1)`. Les valeurs de source et de sortie, les nœuds et le même t_0 figurent dans resultados_propagacion_colapso.json. La lecture répétée renvoie cette capture et n'exécute pas de nouvel effondrement.

Après l'enregistrement de la capture V, la référence du QASM correspondant a été calculée. La référence Q locale attribue une probabilité d'environ 0.5 à chacune de ces deux sorties. L'enregistrement de cette comparaison supplémentaire figure dans `circuitos_adicionales` du JSON de comparaison. Une unique capture n'est pas présentée comme une preuve d'une distribution ni d'une expérience de Bell.

5. Vérifications locales des composants

Exécution directe avec des résultats connus

ejecutar_tareas_conocidas_v.py exécute le logiciel conventionnel V installé, sans moteur Q ni simulation Q. Les données externes déclarent les références connues avant l'essai ; seules les entrées opérationnelles arrivent à V. Ce ne sont pas des mesures de matériel Q. Chaque cas comporte une capture, sans choisir l'instant ni répéter jusqu'au succès.

Le rapport enregistre quatre cas V3 concordants : H-H renvoie `[0,0,0]`, H-RZ(π)-H renvoie `[1,0,0]`, RX(π) sur le troisième port renvoie `[0,0,1]`, et la lecture H-H-CZ-RZ appartient à ses quatre sorties admises. Les trois premiers vérifient une sortie unique ; le quatrième ne démontre pas l'égalité des distributions.

Dans V2, les nouvelles lectures pour N=15 et N=21, base 2, valent toutes deux 28 et ne donnent pas de facteurs avec le même post-traitement classique que Q2. Seule cette fonction classique est chargée, pas le module Q. Une lecture Q2 peut également être inexploitable ; deux captures ne démontrent ni différence générale ni équivalence. L'évaluation des captures précédentes reste identifiée séparément : la lecture 740 pour N=21 retrouve bien 3 et 7.

La famille V3 H-H par unité renvoie zéro pour 2, 8, 32 et 128 unités. Ses deux portes et deux canaux par unité admettent un parcours linéaire à précision fixe ; les temps sont dans le JSON. Cela ne démontre pas l'ordre de V3 arbitraire. Le nombre quadratique de portes du plan V2 ne démontre pas à lui seul l'ordre total de factorisation. Aucun temps de simulation Q n'est utilisé et aucun temps de matériel Q n'est disponible ici pour établir une égalité d'ordres.

Composant	Vérification exécutée	Rapport
Comparateur et latch	9 règles de comparaison, 5 rejets d'entrées invalides, 8 captures avec horloge d'essai et 1 avec horloge réelle	resultados_comparador_temporal.json
Portes binaires conventionnelles	38 cas mesurés concordant avec leurs tables logiques, avec entrées fixées	resultados_puertas_v.json
Propagation de la connaissance	102 motifs, 6 relations composées et 5 déclarations invalides rejetées	resultados_propagacion_conocidos.json
Relations de signaux	10 identités algébriques avec variables sans valeur attribuée et 3 régressions numériques de H-H	resultados_relaciones_senales.json
Propagation d'une capture	2 cas composés, 32 cas de la relation modulaire, 2 captures avec horloge d'essai et le programme H-CNOT avec horloge réelle	resultados_propagacion_colapso.json
Composants de V2	896 captures de blocs avec entrées fixées, 12 chaînes et décompte des connexions pour 7 tailles	verificacion_v2.json
Phases terminales de V3	4 branches d'essai, CP, phase après CNOT, 4 tailles jusqu'à 128 unités et une capture avec horloge réelle et références déclarées	verificacion_v3.json

Composant	Vérification exécutée	Rapport
Règles V2/V3	14 tests : préparation sans lecture, déclenchement commun, propagation, capture unique et noyaux sans tirages aléatoires ni statistiques	auditoria_reglas_v2_v3.json

Ces vérifications sont différentes et ne s'additionnent pas comme si elles étaient toutes des mesures de machines complètes. Les horloges d'essai sont identifiées comme telles et ne sont pas présentées comme des exécutions avec horloge physique de déclenchement. La propagation vérifie des identités telles que `0 AND A = 0` et `A XOR no-A = 1` sans attribuer de valeur à A. Les formules symboliques ne sont pas présentées comme des lectures de la machine préparée.

6. Reproduction locale

Depuis `Traductores_V/Pruebas_V` :

```
python test_v1.py
python test_v2.py
python test_v3.py
python test_propagacion_colapso.py
python verificar_v2.py
python verificar_v3.py
python auditar_reglas_v2_v3.py
python comparar_resultados.py
```

Les chemins par défaut sont résolus à partir du répertoire du programme. Les essais renvoient le code 0 lorsque toutes les mesures sont achevées et le code 2 si l'une reste en attente. Les erreurs de programmation interceptées apparaissent sous `ERROR_EJECUCION`, séparément des connexions en attente. Les rapports propres aux programmes sont régénérés ; les fichiers d'entrée ne sont pas modifiés.

L'exécution V utilise la bibliothèque standard de Python. La référence Q s'installe avec requirements.txt et les tests algébriques optionnels avec requirements_relaciones.txt.

```
python test_comparador_temporal.py
python test_puertas_v.py
python test_propagacion_conocidos.py
python test_relaciones_senales.py
python test_propagacion_colapso.py
```

Si ce dossier est déplacé hors de l'arborescence locale, il faut fournir également les modules Q de référence pour la comparaison ultérieure. V n'en a pas besoin pour s'exécuter. Aucune publication sur le web ni aucune opération Git n'a été réalisée.

La révision complète, ordonnée et avec régénération des rapports s'exécute avec ejecutar_revision_v.py. Elle vérifie V1, examine V2 avant de passer à V3 et exécute Q uniquement à la fin. Le contrôle local vérifie les originaux, DEMUESTRA et les traductions existantes. Le rapport de révision distingue les tests techniques corrects des machines complètes encore en attente. Un code de sortie accepté pour un point en attente ne devient pas une certification de ce point en attente.

7. Borne de travail de V1 : sans analyse interne d'états exponentiels

Le V1 installé a été audité au moyen de analizar_complejidad_v1.py. Le rapport vérifiable se trouve dans analisis_complejidad_v1.json, avec les empreintes du code et des résultats examinés. Cette analyse n'exécute pas Q et n'effectue pas de nouvelles mesures de V.

Ressources des quatre exécutions

Cas V1	Unités	Portes du plan	H	NOT/CNOT de l'oracle	Signaux complexes locaux	Comparateurs	Nœuds finaux
Constante 0	2	3	3	0	4	2	3
Constante 1	2	4	3	1	4	2	3
Identité	2	4	3	1	4	2	3
Inverseur	2	5	3	2	4	2	3

Les quatre signaux sont deux canaux pour chacune des deux unités, et non les entrées d'un vecteur global indexé par les états d'une base conjointe. Le programme n'énumère pas de configurations ni de trajectoires pour sélectionner une réponse. Il ne répète pas non plus la capture jusqu'à ce qu'elle concorde avec un résultat de Q. Il y a un seul déclenchement par préparation.

Argument de complexité

La borne ne repose pas uniquement sur l'absence d'une table exponentielle. Les opérations exécutées par le code sont inspectées :

1. L'oracle de l'essai est compilé à partir de ses deux bits d'entrée en un maximum d'une NOT et d'une CNOT. Il ne reçoit pas de classification calculée par Q.
2. La préparation parcourt un plan de trois à cinq portes. Chaque H, NOT ou CNOT utilisée par ce V1 effectue un nombre borné d'opérations sur ses signaux locaux ; elle n'appelle pas d'évaluateur d'états conjoints.

3. Le déclenchement obtient une référence temporelle, applique deux comparateurs et résout trois nœuds finaux. Il n'attend pas de phase favorable, ne cherche pas de sortie et ne mesure pas à nouveau les sources pendant la propagation.
4. Les listes et registres de travail ont des tailles bornées par ces unités, portes et nœuds. L'algèbre symbolique des tests ne s'exécute pas dans ce chemin numérique de V1.

Avec une arithmétique à précision fixe, le travail du noyau par exécution admet une borne de la forme :

$$T_{V1} \leq C_0 + 5C_{\text{puerta}} + 2C_{\text{comparador}} + 3C_{\text{nodo}},$$

et sa mémoire de travail une borne de la forme :

$$M_{V1} \leq D_0 + 4D_{\text{señal}} + 5D_{\text{conexion}} + 3D_{\text{nodo}}.$$

Les termes constants incluent la préparation des structures et l'enregistrement d'une capture. Ils n'incluent pas la simulation Q ultérieure, qui est un autre processus de vérification et ne produit pas la réponse de V.

Par conséquent, pour **ce V1 à deux unités et le protocole de Deutsch implémenté** :

$$T_{V1} = O(1), \qquad M_{V1} = O(1)$$

par exécution. Pour R répétitions du même protocole, le travail est $O(R)$; conserver tous les rapports nécessite une mémoire supplémentaire $O(R)$. La borne concerne les opérations algorithmiques ; ce n'est pas une garantie de latence du système d'exploitation ni une égalité en secondes avec un dispositif Q.

Conclusion : le V1 exécuté n'incorpore ni recherche ni analyse d'états internes de taille exponentielle. Il produit sa sortie au moyen de ses opérations de signaux, de son comparateur et de ses relations, et non en simulant Q1. Ce résultat concerne le programme et le protocole analysés ; il n'est pas utilisé pour affirmer le passage à l'échelle de V2/V3 ni d'une extension à un nombre variable d'unités que ce V1 n'implémente pas.

Pour répéter l'analyse après avoir généré `resultados_v1.json` :

```
python analizar_complejidad_v1.py
```

Résultat établi de V1

Pour la préparation et le circuit déclarés, la paire finale du contrôle est :

$$s_0 = \frac{(-1)^{f(0)} + (-1)^{f(1)}}{2}, \qquad s_1 = \frac{(-1)^{f(0)} - (-1)^{f(1)}}{2}.$$

Si les valeurs de la fonction coïncident, le second canal est nul ; si elles diffèrent, le premier est nul. Ainsi, pour toute référence de lecture admise par le comparateur, le bit qui résout la tâche satisfait :

$$q_0 = f(0) \oplus f(1).$$

La même réponse au problème de Deutsch est établie pour V1 et Q1 idéal pour les quatre fonctions binaires possibles, avec une capture par préparation et le coût borné ci-dessus. Elle ne dépend pas du choix d'un instant favorable. Cette conclusion concerne le bit qui résout la tâche ; elle ne caractérise pas toutes les distributions ni le matériel Q.

asentar_v1.py vérifie l'argument en arithmétique exacte et par 16 essais de déclenchement, ainsi que les empreintes des quatre captures avec horloge réelle et leur comparaison Q1 ultérieure. Le résultat établi conserve ces preuves. V2 et V3 ont leurs propres exécutions et audits ; leurs règles vérifiées ne remplacent pas l'évaluation des tâches résolues.

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**, conformément à la déclaration éthique de l'auteur ; utilisation commerciale autorisée uniquement si elle respecte ses conditions. Sont interdits : guerres, violations des droits humains, dommages environnementaux, exploitation, surveillance de masse et discrimination. Principes déclarés : respect de la dignité humaine, absence d'utilisation lucrative sans autorisation et interdiction des utilisations nuisibles.
Licence de l'auteur · CC BY-NC-SA 4.0 · HL3.