Automatizált szoftverfejlesztés 3. vizsga (2025. január 23.)

1. feladat: Rövid kérdések — 15 pont

1.1 Teljesítménymérés során a VisualVM-et használjuk (egy népszerű teljesítménymérő eszköz a Java-hoz). Ha mind a Sampling, mind a Profiling módban futtatjuk a mérést, a Profiling módban azt láthatjuk, hogy a kód egyik része sokkal lassabb, mint a többi. Mit mondhatunk a kód ezen részéről?

1.2 Komplex állapotdiagram-modelleket szeretnénk használni egy korlátozott számítási kapacitással rendelkező beágyazott környezetben. Kódgenerátorokat kellene használnunk vagy interpretálva lenne jobb futtatnunk a kódot? Kérem a válaszát fejtse ki!

1.3 Kérem, adjon egy egyszerű példát egy minta-alapú statikus elemzési szabályra!

2. feladat: Gráf alapú modellezés — 25 pont

a) Készítsen egy **Refinery metamodellt** az alábbi specifikáció alapján:

Egy kvantumszámítástechnikai kísérletek és erőforrások kezelésére szolgáló rendszert szeretnénk modellezni. Egy kvantumszámítástechnikai laboratórium számítógépeket tartalmaz, amelyek lehetnek kvantumszámítógépek vagy klasszikus számítógépek. Minden kvantumszámítógép 2-16 kvantumbitből (qubitből) áll. A qubitek lehetnek szupravezető vagy csapdázott ion alapúak. Egy kísérlethez pontosan egy kvantumszámítógép szükséges, valamint legalább egy klasszikus számítógép a vezérléshez. Minden kísérlet egy vagy több kvantumáramkört futtat, amelyek 1-100 kvantumkapuból állnak. A kvantumkapuk lehetnek egy qubites kapuk (egyetlen bemeneti qubittel) vagy két qubites kapuk (egy bal és egy jobb oldali bemeneti qubittel). Minden kvantumáramkör létrehozhat mérési eredményeket. Az eredmény-adathalmazok lehetnek nyers mérési adatok vagy feldolgozott adatok. Minden feldolgozott adathalmaz pontosan egy nyers adathalmazból származik egy adott zajcsökkentő algoritmus alkalmazásával.

Kizárólag Refinery kódot írjon! NE írjon Java kódot vagy rajzoljon UML osztálydiagrammot.

Használja az alábbi fogalmakat: `Dataset` (adathalmaz), `appliedAlgorithm` (alkalmazott algoritmus), `circuits` (áramkörei), `leftInput` (bal bemenete), `input` (bemenete), `TrappedIonQubit` (csapdázott ion qubit), `SingleQubitGate` (egy qubites kapu), `results` (eredményei), `ProcessedData` (feldolgozott adat), `runsOn` (futtatója), `rightInput` (jobb bemenete), `gates` (kapujai), `TwoQubitGate` (két qubites kapu), `Experiment` (kísérlet), `ClassicalComputer` (klasszikus számítógép), `QuantumCircuit` (kvantumáramkör), `QuantumGate` (kvantumkapu), `QuantumComputer` (kvanumszámítógép), `Laboratory` (laboratórium), `RawData` (nyers adat), `qubits` (qubitjei), `Qubit` (qubit), `Computer` (számítógép), `computers` (számítógépei), `derivedFrom` (származik belőle), `SuperconductingQubit` (szupravezető qubit), `controlledBy` (vezérlője), `NoiseReductionAlgorithm` (zajcsökkentő algoritmus)

b) Rajzoljon **gráfmodellt** az alábbi adatok alapján:

A Stanford Kvantum Laboratóriumnak két számítógépe van: egy IBM-Q65 kvantumszámítógép három szupravezető qubittel (alfa, béta és gamma), valamint egy Dell PowerEdge R740, amely vezérlési feladatokra van konfigurálva. Az IBM-Q65-ön egy kvantum összefonódási kísérlet fut, amelyet a PowerEdge R740 vezérel. A kísérlet tartalmaz egy Bell-állapot előállítására szolgáló kvantumáramkört, amely két kvantumkaput tartalmaz: egy Hadamard-kaput, amely az alfa qubiten működik, és egy CNOT-kaput, amelynek bal és jobb oldali bemenetei rendre az alfa és béta qubitek. Az áramkör két adathalmazt állít elő: egy nyers adathalmazt, valamint egy feldolgozott adathalmazt, amely a nyers adathalmazból származik a Richardson-extrapolációs algoritmus alkalmazásával.

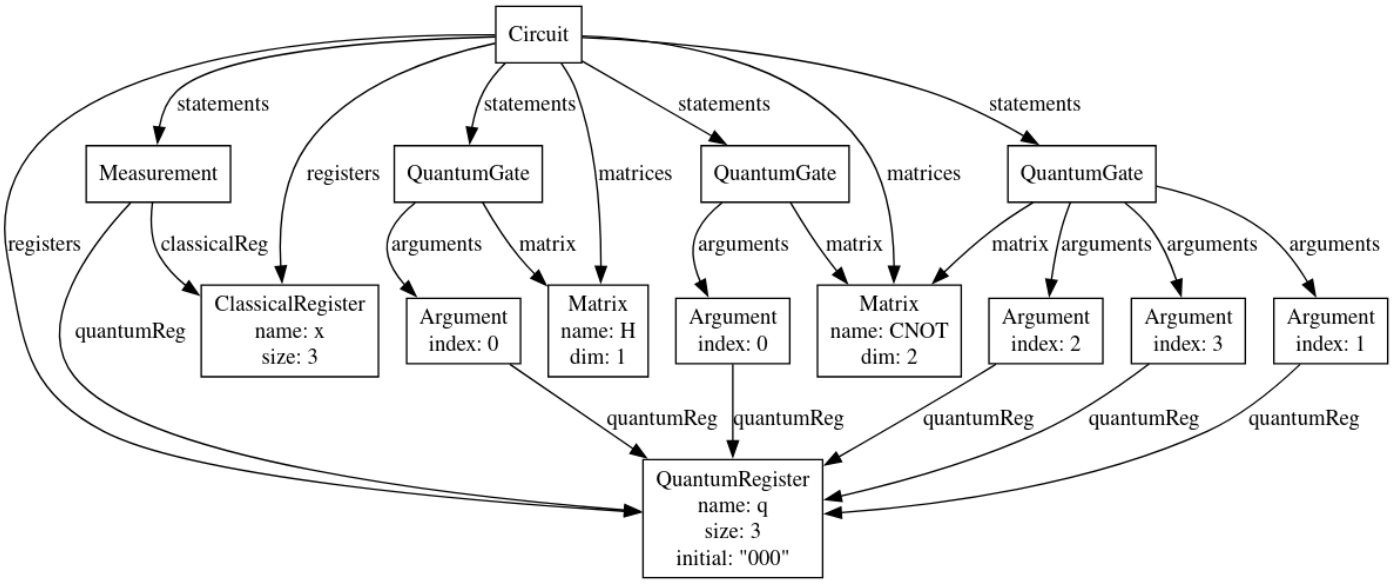
Kizárólag gráfmodellt rajzoljon! NE írjon Java vagy Refinery kódot.

c) Egy szakértő szerint a következő **jólformáltsági kényszer** érvényes: *egy kvantumáramkörben szereplő qubiteknek ahhoz a kvantumszámítógéphez kell tartozniuk, amelyen az áramkör fut.* A szakértő azonban gyanítja, hogy ezt a metamodell jelenlegi verziójával nem lehet kikényszeríteni. Ennek megerősítéshez rajzoljon egy olyan **gráfmodellt,** amely megfelel a metamodellnek, de megsérti a kényszert!

d) A szekértő szerint bizonyos kvantumkapuk akár egyszerre három qubiten is dolgozhatnak. **Javasoljon** egy módszert a metamodel módosítására, hogy az ilyen kvantumkapukat is kezelhessük.

3. feladat: Szöveges modellezés **30 pont**

Egy **szöveges szakterületspecifikus nyelvet** szeretnénk készíteni általános kvantumáramkörök leírásához. Az alábbi példa a kívánt **konkrét szintaxist** (szöveges leírást) és **absztrakt szintaxist** (példánygráfot) mutatja be a nyelvhez:




```
quantum reg q[3] = "000";

matrix H dim = 1;
H(q[0]);

matrix CNOT dim = 2;
CNOT(q[0]);
CNOT(q[1], q[2], q[3]);

reg x[3];
x = measure(q);
```

a) Készítsen egy **Langium nyelvtant** a nyelv elemzésére! Az alábbi deklarációk már rendelkezésére állnak:

```
grammar QuantumProgramming

hidden terminal WS: /\s+/;
terminal ID: /[_a-zA-Z][\w_]*/;
terminal INT: /\d+/;
terminal STRING: /"[^"]*"/;
```

Adja meg a nyelvtan hiányzó részét!

b) Készítsen **Jinja2 sablont** a kvantumáramkör statikus analízis jelentés generálására! A sablon bemenete az a `Circuit` objektum, amelyet az a) részben létrehozott nyelvtani elemzővel olvastunk be. Egy példa statikus analízis jelentés az összes elvárt hibajelzéssel az alábbiakban látható:

```
Too few arguments (got 1, expected 2) for matrix CNOT.
Too many arguments (got 3, expected 2) for matrix CNOT.
Index 3 of register q is out of range.
```

A könnyebb olvashatóság kedvéért a közvetlenül a példánymodellből származó szövegeket **fékövérrel** emeltük ki. Ügyeljen arra, hogy hibajelzést adjon minden olyan kapuhoz, melynek túl sok (`Too many`) vagy túl kevés (`Too few`) argumentuma van, illetve minden olyan argumentumhoz, mely túlcímzi (`out of range`) a megfelelő kvantumregisztert. Nem szükséges ellenőrizze a regiszterek inicializálását vagy a méréseket.

Az példánymodellben a `type` attribútum tartalmazza az objektumok típusát (pl. a `y.type == "Measurement"` használatával ellenőrizheti, hogy az `y` `Measurement` típusú-e). A kereszthivatkozások sztringként vannak tárolva, amelyek értékei megegyeznek a hivatkozott objektum nevével. Egy Python lista, pl. a `someList` hosszának meghatározásához használja a `someList|length` Jinja2 jelölést.