# Több komponensből álló rendszerek

Feltételezzük, hogy egy rendszer több, egymással együtműködő alrendszerből épül fel. A következőkben ezen elemek tulajdonságainak és összekapcsolási lehetőségeinek a részletezésére kerül majd sor.

## Berendezések, mint komponensek

Minden berendezés esetében feltételezzük, hogy a következő tulajdonságokkal rendelkezik.

* id: Egy pozitív egész szám értékű egyedi azonosító.
* type\_id: A berendezés típusának az egyedi azonosítója. (Az ehhez tartozó információkat a rendszer külön kezeli.)
* description: Egy szabad szöveges leírás a berendezés sajátosságaira, működésére vonatkozóan.
* location: A komponens fizikális elhelyezkedése. Szöveges adatként jelenik meg. (Bizonyos berendezések esetében nagyobb távolság is lehet a rendszer részei között, így azok helyének a pontos ismerete szükséges lehet.)
* installation\_ts: A berendezés elhelyezésének az időpontja. Egy időbélyeg.
* ts: A bejegyzéshez tartozó adatok rögzítésének az időpontja. Egy időbélyeg.

Feltételezhetjük, hogy több, azonos típusú berendezésünk is van. Ezeknek az adatait egységesen lehet kezelni. A következő adatok kezelésére van szükség.

* id: A berendezés típusához tartozó bejegyzés egyedi azonosítója. Ez alapján lehet egyértelműen beazonosítani, hogy milyen fajta berendezésről van szó. Ez egy pozitív egész szám érték.
* name: A berendezés típusának emberek számára is könnyen értelmezhető neve.
* description: A típusra vonatkozó esetleges észrevétel (például, hogy ki a gyártója, milyen más berendezésekkel kompatibilis).
* base\_type\_id: Egy másik berendezés típusra mutató azonosító. Ennek a segítségével berendezés altípusok létrehozására van lehetőség. Minden egyes típus mutathat egy általánosabb típusra. Alapértelmezés szerint ennek nincs beállítva érték (például adatbázis szintjén NULL értékű).

A berendezések aktuális állapotának méréséhez, a további működés becsléséhez szenzorok segítségével adatokat kell gyűjtenünk. A berendezésekhez hasonlóan ezeket is érdemes típusokba sorolni. Egy szenzor típus esetében a következő adatok kezelésére van szükség.

* id: A szenzor típusához tartozó bejegyzés egyedi azonosítója.
* name: A szenzor típusának neve.
* description: A szenzor típusához tartozó szabad szöveges leírás. (Opcionális.)
* measured\_type: A szenzor által visszaadott érték típusa. Ez egy kategorikus adat, amely előre meghatározott értékeket vehet fel.

A szenzorok által visszaadott értékek típusai a következők lehetnek.

* BOOL: Logikai érték. Megfeleltethető a 0 és 1 számértékeknek.
* INT: Egészes érték. A kategorikus értékek is ezzel adhatók meg. (Jellemzően a szenzorok alacsony szinten ilyen formában biztosítják az adatokat.)
* FLOAT: Lebegőpontos érték.
* STRING: Szöveges érték. Elsősorban abban az esetben lehet ilyenre szükség, hogy ha a szenzor eleve szöveges formában adja meg az adatokat, továbbá, hogy ha feltételezhető, hogy hibaüzenetet is tartalmazhat a mért érték.

A szenzorok adatai a következők.

* id: A szenzor egyedi azonosítója, mint pozitív egész érték.
* sensor\_type\_id: A szenzor típusa. Hivatkozza a szenzor típusához tartozó leírót.
* description: Szabadszöveges leírás a szenzorra vonatkozóan. (Opcionális.)
* equipment\_id: A berendezés egyedi azonosítója, amelyre a szenzor fel van szerelve. Nem kötelező megadni, mivel előfordulhat, hogy a rendszerben bizonyos szenzorok egyidejűleg több géphez, vagy géptől függetlenül kerülnek felhasználásra.
* location: A szenzor helye szövegesen megadva. Esetlegesen a felszerelés módját is tartalmazhatja.
* ts: A szenzorhoz tartozó rekord létrehozásának az időpontja. Egy időbélyeg.

A rendszer használati módjára vonatkozóan különböző telemetria adatok kezelésére van szükség. A rendszerben ezt kezelhetjük általánosan. Feltételezhetjük, hogy minden esetben a mért értékek diszkrét időpontokhoz rendelve állnak rendelkezésre. Ezekből így a rendszer működése során egy idősort kapunk. Az idősor bejegyzései a következő adatokat tartalmazzák.

* sensor\_id: A szenzor egyedi azonosítója.
* value: A szenzor által mért érték az adott időpontban. Célszerű olyan általános típusnak megválasztani, amely lehetőséget ad a különböző adattípusok egyidejű kezelésére. (Gyengén típusos adatbázismotorok esetében közvetlenül használható ilyen formában. Sémához kötött esetben előfordulhat, hogy külön oszlopok vagy akár táblák megadása is szükséges.)
* ts: A mérés időpontjához tartozó időbélyeg.

## Berendezések közötti kapcsolatok

A berendezések között különféle összekapcsolási lehetőségek vannak. Ezek közül a két legfontosabbal: a soros és a párhuzamos kapcsolással érdemes foglalkozni. Minden kapcsolás esetében feltételezhetjük, hogy az összekapcsolással kapott topológia egy irányított, körmentes gráfot eredményez.

A soros kapcsolás esetén a berendezések egymás után vannak kapcsolva.

* A kapcsolást egy láncnak tekinthetjük.
* A modellezett rendszer működéséhez szükséges, hogy minden alrendszer önmagában is megfelelően működjön.
* Egy alrendszer kiesése a teljes rendszer meghibásodását eredményezi.

A párhuzamos kapcsolásnál az összekapcsoláson belül lévő berendezések működését alapvetően függetlennek tekinthetjük.

* Egy berendezés meghibásodása nem vonja maga után a teljes rendszer meghibásodását.
* Feltételezzük, hogy ha legalább egy elem működik az alrendszeren belül, akkor a teljes rendszer működőképes marad.

Minden berendezés esetében becsülni tudjuk a rendszer várható élettartamát.

* Soros kapcsolás esetében a várható élettartam az összekötött berendezések várható élettartamainak a minimuma.
* Párhuzamos kapcsolás esetén a várható élettartam az alrendszerben szereplő berendezések élettartamának maximumával egyenlő.

A számításokat elvégezhetjük a berendezés előre adott (rögzített) időintervallumban történő meghibásodásának az esetére is. Jelölje $p\_{1},…,p\_{n}$ az $n$ számú, sorba kötött berendezés adott időintervallumra vizsgált meghibásodásának a valószínűségét! Feltételezzük, hogy egy berendezés rossz műszaki állapota közvetlenül nincs hatással az azt megelőző, vagy azt követő berendezések meghibásodására. (Amennyiben ez nem teljesülne, úgy a meghibásodási valószínűségeket közvetlenül meg kell adni a megfelelő $p$ értékben.)

Vizsgáljuk meg először a soros kapcsolás esetét! Annak a valószínűsége, hogy az adott időintervallumon belül egyetlen berendezés sem hibásodik meg, a következőképpen számítható ki:

$$\prod\_{i=1}^{n}(1−p\_{i}).$$

Annak a valószínűsége tehát, hogy az adott időintervallumban a berendezés meghibásodik:

$$1−\prod\_{i=1}^{n}(1−p\_{i}).$$

Párhuzamos kapcsolás esetére a számítások hasonlóképpen elvégezhetők. Ekkor annak a valószínűsége, hogy mindegyik berendezés meghibásodna az adott intervallumon belül:

$$\prod\_{i=1}^{n}p\_{i}.$$

Legyen adott az $M=\{m\_{1},…,m\_{n}\}$ halmaz. A gépek összekapcsolását, mint egy relációt adhatjuk meg. Jelöljük ezt $R⊂M×M$ formában! A relációval kapcsolatban a következőket feltételezzük.

* Minden gép összeköttetésben van magával (tehát a reláció reflexív). A számítások szempontjából ezzel különösebben nem kell foglalkoznunk.
* Az összeköttetést tranzitívnak tekintjük. Amennyiben például egy gép soros összeköttetésben van egy másikkal, úgy az egyik meghibásodása maga után vonja azt, hogy az utána következők sem tudnak megfelelően működni.
* A reláció antiszimmetrikus, vagyis ha például van egy $(m\_{i},m\_{j})$ összeköttetés, akkor nem lehet egyidejűleg $(m\_{j},m\_{i})$ összeköttetés is.

Belátható, hogy ez a struktúra egy hálót eredményez. A számítások vizsgálatához tekintsük először az egyszerűbb, moduláris háló esetét! Minden esetben célszerű megadni explicit módon egy minimum és maximum értéket (amelyet a rendszer be- és kilépési pontjának tekinthetünk).

Érdemes azt észrevenni, hogy a gráfban a körmentesség egy szigorú megkötésnek tünne, de könnyen kezelhetővé teszi a gyakorlatban előforduló eseteket.

* Amennyiben az alrendszerben előfordul valamilyen körkörös függőség, az azt jelenti, hogy bármely, a körbe kapcsolat gép meghibásodása a teljes kör meghibásodását jelenti. Így egyszerűen felírható soros kapcsolásként is.
* A körmentesség teszi azt indokolttá, hogy egyáltalán számítások végzésére legyen szükség, mert egyébként elegendő lenne csak a várhatóan leghamarabb meghibásodó gépet tekinteni.

Jelölje $a\in M$ a rendszer belépési pontját (minimumát), $b\in M$ pedig a kilépési pontot (maximumot). Belátható, hogy (moduláris háló esetében) a rendszer bármely része, amelynek minimuma és maximuma a közrefogott elemek minimuma és maximuma között van, kiváltható egyetlen egy géppel. Az adott időintervallumra vonatkozó valószínűségek alapján meghatározható a kiváltott részre vonatkozó meghibásodási valószínűség az előzőekben bemutatott számítások felhasználásával. Ennek praktikusan az az előnye, hogy így a teljes rendszer meghibásodási valószínűségét becsülni lehet a részek, és azok összeköttetéseinek az ismeretében.

// flow 1

A párhuzamos részek egyszerűsítését követően a következő hálózatot kapjuk.

// flow 2

Ez alapján egyszerűen kiszámíthatjuk, hogy a teljes rendszerre vonatkozó meghibásodási valószínűség 0.510801472 körül van.

Az algoritmus bemutatása előtt érdemes azt megnézni, hogy magát a gráfot hogyan lehet ábrázolni. A következő elterjedt alternatívák állnak rendelkezésre.

* A gépek közötti relációt tárolhatjuk egy bináris mátrix formájában.
* A matematikai definíciónak megfelelően gyakorlatilag definíció szerint megadhatjuk a gráf éleinek a halmazát, mint rendezett indexpárok halmazát.
* Definiálhatunk a gépeknek megfelelő csomópontokat, melyek tartalmazzák a belőlük kifelé mutató éleket.

A belső reprezentációtól olyan tekintetben elvonatkoztathatunk, hogy a számításokhoz elegendő, hogy ha

* le tudjuk kérdezni, hogy az összeállításnak melyik a belépési pontja,
* egy csomóponthoz milyen meghibásodási valószínűség tartozik, továbbá hogy
* mely csomópontokba mutatnak az adott csomópontból kifelé mutató élek.

A továbbiakban is feltételezzük a következőket.

* Moduláris hálót eredményeznek az összeköttetések.
* Soros és párhuzamos összeköttetés esetében is kommutatív és asszociatív a valószínűség meghatározásához használt művelet.

Az utóbbi azt jelenti, hogy tulajdonképpen elegendő mindig csak 2 elem viszonyában gondolkozni. Egy aránylag egyszerű rekurzív módszer adható, amely tetszőleges moduláris háló esetében képes meghatározni az egész rendszerre vonatkozóan a valószínűségeket. Ennek a lépései a következők.

* 1. Kérdezzük le, hogy melyik a rendszer belépési pontja!
* 2. Nézzük meg, hogy belőle mennyi kimeneti él van.
* Hogy ha nincs kimeneti él, akkor az aktuálisan számított valószínűség lesz a végeredmény.
* Hogy ha pontosan egy kimeneti él van, akkor soros kapcsolásról van szó. Vonjuk össze az aktuális valószínűséget a következő elem valószínűségével.
* Hogy ha több kimeneti él is van, akkor mindegyik ágra végezzük el rekurzívan a valószínűség meghatározását, majd párhuzamos összeköttetésnek megfelelően aggregáljuk az eredményeket.

A gráf összeállításához felületet is biztosítani kell a felhasználóknak. Ezen a következő műveleteket kell tudni elvégezni.

* Létre kell tudni hozni egy új, több komponensből álló konfigurációt (amely kezdetben üres).
* Be kell tudni szúrni új komponenst.
* A beszúrt komponenseket ki kell tudni jelölni.
* A kijelölt komponenshez tartozó paramétereket be kell tudni állítani. (Ilyen paraméter például a meghibásodás valószínűsége az adott időintervallumra nézve, illetve a meghibásodás várható idejének az eloszlása.)
* Egy kijelölt komponenst tudni kell törölni.
* A komponenseket tudni kell mozgatni egy rajzterület félén.
* A komponensek közötti éleket meg kell tudni adni.
* A grafikusan megjelenített konfiguráción jelezni kell az esetleges problémákat (például, hogy ha kör lenne a gráfban, sérülne a moduláris tulajdonság a gráfra nézve, vagy ha valahol hiányozna valamilyen paraméter).
* Az élettartam előrejelzéshez tartozó paraméterek átállításakor a programnak automatikusan újra kell számolnia a teljes rendszerre vonatkozó számszerű becsléseket, és valós időben meg kell azokat jelenítenie.
* Az összeállított konfigurációt el kell tudni menteni.
* Korábban elmentett konfigurációt be kell tudni tölteni.
* A konfigurációhoz tartozóan metainformációkat (nevet, szöveges leírást) el kell tudni helyezni.

Nem moduláris esetben a számítások többségében elvégezhetők a moduláris esethez hasonlóan. A különbséget az adja, hogy ha a modularitás nem garantált, akkor figyelni kell, hogy minden számítás az antiláncok mentén, a háló szintjeinek megfelelően történjen. (Előfordulhat, hogy akár a be- és kimeneti csomópont között is közvetlenül van él. Ilyen esetben a később előforduló csomópontokhoz tartozó valószínűségeket a számítás közbülső lépéseiben aktualizálni kell.)

## További kapcsolódási módok

A felsorolt soros- és párhuzamos kapcsolódási módokon kívül előfordulhatnak további esetek is. Ilyen lehet például, hogy ha egy alrendszerben redundáns módon több eszköz jelenik meg, és a rendszer helyes működéséhez adott számú komponensnek egyidejűleg kell tudni működni. Ez tulajdonképpen a párhuzamos kapcsolás egy plusz feltétellel megszorított változatának tekinthető.

A gyakorlatban találkozhatunk ilyenekkel, például, hogy ha egy számítógép esetében redundáns módon vannak tápegységek, és legalább kettőre minden esetben szükség van.

Hasonló szisztémáról van szó RAID-be kötött winchesterek esetében is.

A sorosan összekötött egységek esetében külön vizsgálandó az, hogy egy eszköz tud-e működőképes maradni úgy, hogy a hozzá kötött további egységek helyes működését gátolja. Példa lehet erre egy túlmelegedett motor vagy a szükségesnél nagyobb feszültséggel működő elektromos egység.

# Szenzoradatok feldolgozása

Az összegyűjtött szenzoradatok áttekintéséhez, elemzéséhez az alkalmazásnak felületet kell biztosítania. Ennek a következő műveleteket kell tudnia biztosítania.

* Ki kell tudni jelölni, hogy mely szenzoroknak az adatainak a megjelenítésére van szükség. Ehhez ki kell tudni listáztatni, hogy milyen gépek és milyen szenzorok vannak a teljes rendszerben.
* Egyidejűleg több szenzort is hozzá kell tudni adni.
* A kiválasztott szenzorokhoz tartozó adatokat meg kell tudni jelenítenie a rendszernek (például popup ablak formájában).
* Ki kell tudni jelölni azt az időintervallumot, amelyre a vizsgálat vonatkozik.
* Az időszakon belül kiválasztott szenzorok adatait egymás alatt (közös időtengellyel) meg kell tudni jeleníteni.
* A megjelenítésből a szenzorokat ideiglenesen ki kell tudni hagyatni (például a nevére kattintva).
* A vizsgált időintervallumot dinamikusan lehessen szűkíteni, bővíteni (zoom jellegű funkcióval).
* Az alkalmazás használjon adatkurzort, amely segítségével az adott grafikonon (tipikusan töröttvonalas közelítésen) ki lehet jelölni időpontot és a hozzá tartozó értéket kényelmesen le lehet olvasni.
* Lehessen adott dátumra, időszakra ugrani.
* A megjelenített értéktartomány alapértelmezés szerint igazodjon a megjelenített értékekhez, de legyen arra opció, hogy azokat közvetlenül is be tudja állítani a felhasználó.
* Lehessen váltani a hisztorikus adatok elemzése, és az aktuálisan bejövő új értékek folyamatos megjelenítése (követése) között.
* A szenzor típusától függően lehessen grafikus felületről is megadni egyszerű aggregációs műveleteket, amelyek segítik a későbbiekben az elemzést. (Például, hogy egy adott időintervallumon belül mennyi hibajelzés következett be.)

A szenzoradatok automatizált feldolgozásához a felhasználónak ki kell tudni jelölni, hogy milyen bemenetekből milyen kimenetek becslését szeretné elvégezni. A rendszernek grafikus felületen meg kell tudnia jeleníteni a következőket.

* A kiválasztott szenzor értékeinek eloszlását egy adott időintervallumra nézve.
* A korábbi vizsgálatok eredményeképpen az eszköz várható élettartamának az eloszlását.
* Előre rögzített időintervallumra vonatkozóan a meghibásodás bekövetkezésének a valószínűségét.
* A felhasználónak meg kell tudnia adni szinteket, amelyet egy valószínűségi értékre vonatkozóan már kritikusnak tekint. Ezeket a rendszer egy zöld-sárga-piros színskálán kezeli. A felhasználónak definiálni kell tudni egy nemlineáris skálázást, vagyis, hogy az egyes százalékokat mennyire tekinti kritikusnak.

Az alrendszerekből összeállított rendszermodellnek, és az egy komponensre vonatkozó becslésnek egyidejűleg, egymással összekötve kell tudnia működni. Minden komponenshez a várható meghibásodásra vonatkozó színezéssel, jelöléssel áttekinthető formában meg kell tudnia jelenítenie a felületnek, hogy összességében a rendszer milyen állapotban van. A visszacsatolást ez esetben a szenzorokból érkező adatok jelentik. A felhasználóknak a szenzoradatok elemzésére a következő lehetőségek kell, hogy adottak legyenek.

* A szenzoradatoktól függetlenül a felhasználónak meg kell tudni azt adni, hogy az eszköz éppen milyen állapotban van. Ez a legegyszerűbb esetben azt jelenti, hogy a felhasználó maga adja meg kézzel, hogy milyen eloszlást követ az eszköz várható élettartama, illetve hogy az adott, rögzített időintervallumra vonatkozóan mennyi a meghibásodás valószínűsége.
* Amennyiben a rendszerben felvezetésre kerültek a korábbi meghibásodások, úgy azok alapján egyszerű, gyakorisági alapon történő számítással a rendszernek meg kell tudni becsülni a várható élettartamot, illetve a meghibásodási valószínűséget.
* A felhasználó ismerhet olyan összefüggéseket, amelyek alapján a szenzoradatokból közvetlenül becsülhető a gép további problémamentes működésének az ideje. (Ilyen lehet például, hogy ha egy adott hibaüzenetet ad ki a gép, akkor nagy valószínűséggel egy héten belül el fog romlani.) Ezek rendszerbe építéséhez a felhasználónak meg kell tudnia adni szabályokat, amelyek a szenzoros értékek alapján becslik a várható működési időt és a hozzá tartozó valószínűségeket.
* A rendszernek automatizált módon becslést kell tudnia adni arra, hogy a szenzoradatok és a korábbi szenzoradat-meghibásodás párok alapján milyen valószínűséggel következhet be egy meghibásodás egy adott időintervallumon belül.

## Várható élettartam becslése

Az eszköz várható élettartamára vonatkozóan a számításokat a korábban rendelkezésre álló élettartamok alapján lehet elvégezni. Ez alapvetően időtartamok sorozatát jelenti, amely egy működés üzembe helyezését követően az első meghibásodásig tartó időszakok hosszát jelenti. Jelöljük ezt $t\_{1},…,t\_{N}$ formában!

A meghibásodás várható időpontjának az eloszlásához a következő számítási lépéseket kell elvégezni.

* Tegyük fel, hogy van egy gépünk, amely az üzembe helyezését követően már $Δt$ ideje üzemel.
* A sorozatból ki kell választani egy részhalmazt, amelyben csak a $Δt$-nél nagyobb időértékek szerepelnek. Formálisan ez a $H=\{t:t>Δt\}$.
* Számítsuk ki a hisztogramot a $H$-ban lévő elemekre, majd jelenítsük meg.

A számításokhoz egyaránt használhatunk gyakoriság vagy sűrűség hisztogramot, mivel a különbség csak a függőleges tengely egységeiben jelentkezik.

Külön esetként érdemes azt kezelni, hogy ha a $H$ üres halmaz. Ekkor ugyanis a becslés nem végezhető el. A felhasználónak célszerű ezt külön jelezni.

Könnyen belátható, hogy a $H$ elemszáma $Δt$ növelésével folyamatosan csökken. A becslés pontossága az elemszám függvénye, vagyis szükségszerűen egyre pontatlanabb becslést fogunk tudni kapni.

A bekövetkezett meghibásodásokat érdemes beépíteni a becsléshez használt sorozatba.

A hisztogram felosztását érdemes állítható formában megadni. Alapértelmezett értékként a $\sqrt{|H|}$ értékkel lehet számolni.

A meghibásodás valószínűségének becsléséhez szintén egy $t\_{1},…,t\_{N}$ sorozatot tekinthetünk bemenetnek. A függvény, amely az eszköz meghibásodására utal egy adott időintervallumon belül, felírható a $P(ξ<t)$ formában, ahol $ξ$ a véletlentől függő változónk, $t$ pedig az az érték, amely az aktuális időponttól kezdve eltelt. A számítás lépései a következők.

* Tegyük fel, hogy van egy gépünk, amely az üzembe helyezését követően már $Δt$ ideje üzemel.
* Válasszunk egy felbontást, amely segítségével adott időpontokra nézve becsülni szeretnénk a valószínűségeket. (Ehhez támpontként szolgálhat például a megjelenítőrendszer aktuális felbontása is, továbbá, hogy a számítások elvégzéséhez a rendszernek mennyi időre van szüksége.)
* Minden egyes kiválasztott időpillanathoz határozzunk meg egy $k$ értéket, amely azt mutatja, hogy mennyi olyan eset fordult elő korábban, amelyben a gép meghibásodása $t$ időnél korábban következett be. Ez formálisan:

$$k(t)=|\{t\_{i}−Δt:0<t\_{i}−Δt<t\}|.$$

* Határozzuk meg, hogy mennyi olyan esetünk volt, amelyben a gép legalább $t$ ideig működött:

$$n(t)=|\{t\_{i}−Δt:0<t\_{i}−Δt\}|.$$

* Annak a valószínűsége, hogy az aktuálisan vizsgált gépünk $Δt$ idő után $t$ idő elteltén belül fog meghibásodni a következő formában becsülhető:

$$P(ξ<t)≈\frac{k(t)}{n(t)}.$$

Érdemes azt figyelembe venni, hogy ezen számításnál a $Δt$ érték magában a $k(t)$ és az $n(t)$ értékben jelentkezik, vagyis a $ξ$ valószínűségi változó nem úgy általában a gép működésének az idejére, hanem az aktuális időponttól kezdve ad becslést. Ennek praktikusan akkor van jelentősége, hogy ha a felhasználó számára rögzített időközönként újra szeretnénk számoltatni a meghibásodási valószínűségeket.

# Rejtett Markov Modell szoftveres komponensének tervezése

A Rejtett Markov Modellel történő számításokhoz a programnak kezelnie kell tudni az állapotokat (beleértve a rejtetteket és a megfigyelhetőket), továbbá az azok közötti átmeneteket. A következőkben ennek egy lehetséges szoftveres megvalósításának terveit tekinthetjük át.

Definiáljuk a Rejtett Markov Modellt egy HMM nevű osztály formájában. Ennek legyenek a metódusai a következők.

* setStates: Beállítja, hogy a modellben milyen állapotok lesznek.
* setInitialDistribution: Beállítja, hogy az elérhető állapotok milyen valószínűséggel lehetnek kezdő állapotok.
* setObservableStates: Beállítja, hogy melyek a megfigyelhető állapotok.
* setTransitionProbabilities: Beállítja, hogy az állapotok között milyen valószínűséggel történhetnek átmenetek.
* setObservationDistribution: Megadhatjuk vele, hogy az elérhető állapotokhoz milyen eloszlás szerint tartoznak a megfigyelhető állapotok.
* calcSequenceProbability: Megkapja a megfigyelhető állapotok egy sorozatát, és megállapítja, hogy azok milyen valószínűséggel fordulhatnak elő.
* predictNextState: Egy állapotváltozás szekvencia alapján becsli, hogy melyik lehetséges következő állapot előfordulásának a legnagyobb a valószínűsége.

A szoftver grafikus felületén érdemes megjeleníteni a következő adatokat.

* A rejtett állapotok számát.
* A megfigyelhető állapotokat, és a hozzájuk tartozó neveket.
* A kezdeti állapotokhoz tartozó valószínűségeket egy hisztogram formájában.
* Az átmeneti valószínűségek mátrixát. Ezt célszerű hőtérkép jelleggel megjeleníteni.
* Egy adott szekvenciához tartozó valószínűségeket.
* A felhasználó által megadott két állapot között a legnagyobb valószínűségű átmenetet. (Ezt olyan formában is érdemes megtenni, hogy a lépésszámot rögzíti hozzá a felhasználó.)
* A nagyobb (vagy akár csak a nem nulla) valószínűségű átmenetekhez tartozó irányított, súlyozott gráfot (mint a modell grafikus reprezentációját).

# Fuzzy Szabály Interpolációs komponens

A fuzzy szabályinterpolációs módszerhez tartozó komponens működésének egy áttekintő jellegű ábrája a következő.

// fri loop

Az elemei a következők.

* A grafikus felhasználói felület arra szolgál, hogy a felhasználó ezen keresztül definiálni tudja a szabályrendszert, majd a használatával kapott becsléseket, eredményeket meg tudja tekinteni. (Ez az ábra nem részletezi, de több felületi elemről van szó, mivel a szabályrendszer definiálása, a szimulációk és a becslések más jellegű problémák, így más felületet igényelnek.)
* Az ábrán a szenzorok az összes olyan érzékelőt jelzik, amely a következtetéshez bemenetet képes biztosítani. A következtetés során az, hogy ezekből mely adatokra van szükség, a szabályrendszer definiáláskor kerül meghatározásra.
* A nyelvi értelmező feladata, hogy a szöveges formában definiált szabályleírásból egy közbülső reprezentációt hozzon létre.
* A következtető automata a közbülső reprezentáció formájában megadott szabályrendszerből, és a szenzorokból érkező adatokból képes következtetni a kimenetekre.
* A következtető automata kimenetét, annak jelentését, hatását előre definiálni kell. Maga a következtető automata a kimenetet csak egy valós vektor formájában biztosítja. Arról, hogy az hogyan kerül megjelenítésre, vagy milyen további mechanizmusokat vált ki, a rendszer többi részének kell gondoskodnia.

A szenzorból érkező adatok esetében fontos, hogy azok folytonos értékre legyenek konvertálhatóak. Ez tehát tudja kezelni közvetlenül a logikai értékeket (mint 0 és 1 egész értékeket), és a kategorikus adatokat is. (Utóbbi esetében a számítási modellből kifolyólag van sorrendi kötöttség. Egy-egy adott szimbólum értéke szomszédsági kapcsolatot feltételez az ugyanazon dimenzióban/univerzumon lévő más szimbólumokéval.)

A nyelvi értelmező és a következtető automata egy szorosan összekapcsolt egységet képez. Nevezzük ezt a továbbiakban Engine-nek. Az ehhez tartozó metódusok a következők.

* loadBehavior: Egy szabályrendszer betöltésére szolgál. Minden esetben egy aktív szabályrendszere van a következtető automatának.
* updateBehavior: A működő rendszer alatt képe kicserélni a viselkedésleírást. Csak abban az esetben hajtja végre a műveletet, hogy ha az a szabályrendszer, amelyre cseréli az aktuálisat, minden tekintetben megfelelő. (Például, hogy nem vezet be olyan új bemeneti állapotot, amelyhez nem állnak rendelkezésre adat.)
* initialize: Beállítja a kezdeti állapotok értékét a paraméterben megadottaknak megfelelően.
* setState: Név alapján beállítja egy adott állapotváltozó értékét.
* getState: Név alapján lekérdezi egy adott állapotváltozó értékét.
* getStates: Lekérdezi az összes aktuális állapotot, és kulcs-érték párok formájában adja vissza.
* getSymbol: Név alapján lekérdezi egy állapotváltozó értékét, majd (a kimeneti univerzumon definiált) szimbólum formájában biztosítja a végeredményt.
* step: Az állapotgépet az aktuális állapotból a következő állapotba lépteti.
* collectRulebases: Listázza, hogy milyen szabályrendszerek vannak aktuálisan használatban.
* collectUniverses: Listázza, hogy milyen univerzumok vannak aktuálisan definiálva és használatban.

A szoftver elemhez tartozó további osztályok szerkezete a következő ábrán látható.

// fri engine

* :
* :
* :
* :