UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Segundo cuatrimestre de 2017

Versión 3: 23 de Octubre

TPI - Sala de Reuniones

En esta oportunidad, deberán implementar varias de las funciones especificadas para el Trabajo Práctico de Especificación "Sala de Reuniones" en lenguaje imperativo (C++). Para realizar este trabajo contarán con conversaciones reales en donde podrán probar sus funciones, ya sea a través de pruebas manuales o tests.
En nuestra implementación y especificación, contaremos con los siguientes renombres de tipos:

```
typedef vector<int> audio;
typedef vector<audio> sala;
typedef float tiempo;
typedef tuple<float,float> intervalo;
typedef vector<intervalo> lista_intervalos;
typedef vector<float> distancias;
```

1. Ejercicios

1.1. Parte 1: Funciones vistas

Para el TPI se requiere que se implementen las siguientes funciones, ya presentadas durante el TPE, siguiendo la especificación **provista por la cátedra**.

1. `bool esGrabacionValida(audio vec, int prof, int freq);`
2. `int elAcaparador(sala m, int freq, int prof);`
3. `sala ardillizar(sala m, int freq, int prof);`
4. `sala flashElPerezoso(sala m, int freq, int prof);`
5. `vector<intervalo> silencios(audio vec, int freq, int prof, int umbral);`
6. `bool hayQuilombo(sala m, int freq, int prof, int umbral);`

1.2. Parte 2: Nuevas funcionalidades

7. Con el propósito de poder comparar nuestro algoritmo de detección de silencios con los silencios reales en la conversación, se pide implementar la siguiente función: `float compararSilencios(audio vec, int freq, int prof, int locutor, in`
   Este procedimiento (que no posee especificación explícita) evalúa los resultados devueltos por **silencios** comparándolo contra silencios reales en la conversación. Se necesita que este procedimiento cumpla con los siguientes pasos:

   a) Usando I/O cargar los intervalos de habla de los archivos de los locutores en la carpeta **datos**. Los nombres de los archivos son **habla_spkX.txt** donde la X indica el numero del locutor (un entero entre 0 y 5). Cada línea de estos archivos posee el tiempo de inicio y fin de cada intervalo de habla para el locutor dado. Por ejemplo, el siguiente caso se trata de una persona que habla en 3 intervalos distintos de tiempo.

   ```
   0.02 0.04
   0.07 0.11
   0.12 0.13
   ```

   Suponer que el archivo de entrada contendrá intervalos ordenados con números y que todo intervalo será válido, en rango y disjunto con el resto.

   b) Implementar y utilizar la función **enmascarar** que convierte la lista de intervalos (t0, tf) a una máscara (vector booleanos) que represente si hay habla o no cada 10 milisegundos (es decir, la longitud de la máscara será 100 veces la duración del audio en segundos). Por ejemplo, para el caso anterior, si suponemos que la duración del audio es 0.15 segundos, la máscara resultante será <F, F, T, T, F, F, F, T, T, T, T, F, T, F, F>. Ver especificación para más detalles.

   c) Implementar y utilizar una función **negacionLogica** que a partir de la máscara anterior, genere una máscara inversa. Ver especificación.

d) Convertir a máscara de silencios los intervalos de silencios detectados por el algoritmo (con el umbral pasado como parámetro). Para ello utilizar **silencios** y **enmascarar**.

e) Comparar las mascaras obtenidas. Para ello, computaremos la cantidad de **verdaderos positivos**[1], **verdaderos negativos**[2], **falsos positivos**[3] y **falsos negativos**[4]. Luego, a partir de estos números computar el F1 score. F1-score se define como:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

donde $precision = vp/(vp + fp)$ y $recall = vp/(vp + fn)$. El estadístico F1 es mejor si es cercano a 1. El peor valor de F1 es 0.

Finalmente, la salida de la función **compararSilencios** será la métrica F1-score aplicada a los valores calculados para ese locutor.

f) Implementar la funcion `float resultadoFinal(sala m, int freq, int prof, int umbralSilencio);` que determina el $F1$-score promedio entre todos los locutores.

g) **opcional** (por la gloria)
Encontrar, para el algoritmo propuesto, el mejor umbral de silencio posible. Además, probar variando duración mínima del silencio, tomando en cuenta si en otro canal se detectó o no silencio, etc. para lograr mejorar la puntuación de $F1 - score$ promedio.

Para pensar:

- El audio que les otorgamos representa una parte de una conversación real. Teniendo este dato, ¿piensan que las mejoras propuestas funcionarán para el resto de la conversación?

8. `audio sinSilencio(audio vec, int freq, int prof);`

   Esta función elimina los silencios del audio. El nuevo audio, cuya longitud puede ser menor o igual que el original, no contiene ningún silencio. Ver especificación. Se recomienda probar esta funcionalidad (ya sea con las conversaciones que provee el TP o audios propios convertidos al formato requerido).

9. En el área del procesamiento de habla, es muy común intentar buscar apariciones de señal dentro de otra. Esto ocurre por ejemplo cuando en una sala con más de un micrófono se quiere saber qué persona produjo cada palabra.

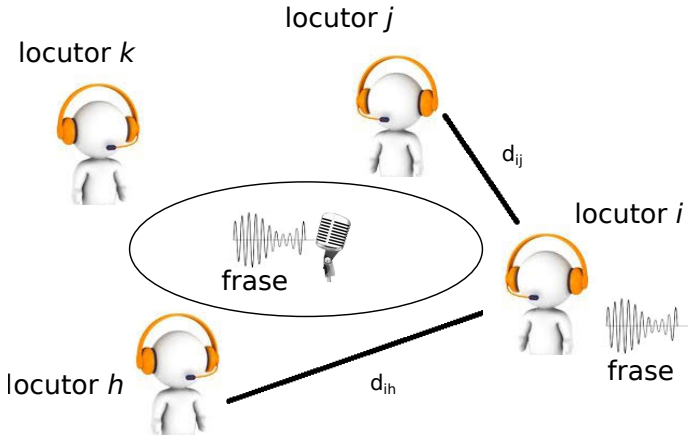
   A continuación deberán implementar la función

   `int encontrarAparicion(audio target, audio s, int freq, int prof);`

   Esta función deberá devolver el índice en la cuál comience la subsecuencia de la señal s que más se parezca a la señal $target$.

   Hay distintos métodos para comparar dos señales, en este caso utilizaremos la correlación entre señales (función provista por la cátedra). Ver especificación para más detalles.

10. Por último, aprovechando que sabemos buscar apariciones de señales en señales y que se cuenta con un micrófono en medio de la sala capturando todo lo que ocurre alrededor, nos convertiremos en físicos por un día y calcularemos la distancia a cada micrófono desde la persona que habló ¡utilizando la velocidad del sonido!



Se pide implementar la función

[1] vp = cantidad de T detectados que eran T realmente
[2] vn = cantidad de F detectados que eran F realmente
[3] fp = cantidad de T detectados que eran F realmente
[4] fp = Cantidad de F detectados que eran T realmente

```
vector<float> medirLaDistancia(sala m, audio frase, int freq, int prof);
```

Esta función recibirá como parámetro una subsecuencia de audio capturada por el micrófono ambiente, que registra las voces de la sala de reuniones.

Las tareas que tiene que realizar esta función son:

- Encontrar el locutor i que dijo la frase.
- Calcular la distancia de los otros locutores j al locutor i.

El locutor que dijo la frase será aquel con la intensidad media mayor en el fragmento de audio que corresponde a la máxima correlación con la señal recibida por parámetro.

Luego, se debe calcular las distancias de los otros locutores a este locutor a partir de la demora de tiempo encontrado comparando contra el hablante.

La salida es una lista de distancias al locutor ordenadas según el orden de m.

Se asume que los registros de los locutores, tomados a partir los headsets, están a la misma distancia de la boca y que todos los micrófonos capturan la frase.

2. Funciones Auxiliares

Las siguientes funciones son provistas como herramientas para leer los datos guardados en el disco, generar la *matriz* de audios y también para poder guardarlos en un archivo de texto, cuando se quiera verificar el resultado.

- `audio leerVectorAudio(string nombreArchivo);`
  Lee el audio completo y lo devuelve en un vector de enteros.
- `audio leerSubVectorAudio(string nombreArchivo, int ini, int longitud);`
  Lee un segmento del audio comenzando en *ini* y con una longitud dada.
- `void grabarVectorAudio(audio vec, string nombreArchivo);`
  Guarda el contenido de un vector de enteros en un archivo de texto.
- `sala cargarSalaAudio(vector<string> archivos, int ini, int longitud);`
  Carga en la matriz (secuencia de secuencias) sala los audios correspondientes a un vector de nombres de archivos desde la posición *ini*.
- `float correlacion(audio x, audio y);`
  Aplica la función de correlación entre las señales x e y. Supone que las dos señales tienen la misma longitud.

3. Herramientas y Datos

Como guía de implementación, el alumno tiene a sus disposición audios convertidos a txt en el directorio **datos**, y un conjunto de casos de test (test suite) para evaluar el correcto funcionamiento de la implementación.
Los audios fueron descargados de la pagina de la Universidad de Columbia `https://www.ee.columbia.edu/~dpwe/sounds/mr/` y corresponden al proyecto **ICSI Meeting Recorder**. El corpus de la base son audios de 6 locutores que utilizaban headsets individuales. De esos audios se extrajo una muestra desde el segundo 60 hasta el segundo 180. Las transcripciones se generaron de manera que el momento 0 coincida con el momento 0 del audio recordato. Además se agrega un audio correspondiente a un micrófono ambiente ubicado en el centro de la mesa de la sala de reunión.
En el directorio UtilesPython, se provee un script para convertir los vectores de texto a archivo .wav, y poder verificar el resultado de alguno de los ejercicios. Para poder correr este script en una PC que no sea del laboratorio es necesario instalar Python. En Linux y Mac vienen instalado por defecto y en Windows se recomienda instalarse la distribución Anaconda `https://anaconda.org/anaconda/python` de Python versión 3.

4. Entregable

1. Entregar una implementación de las funciones anteriormente descritas que cumplan el comportamiento especificado en la Especificación. El entregable debe estar compuesto por todos los archivos necesarios para leer y ejecutar el proyecto y los casos de test adicionales propuestos por el grupo.
2. **Importante: Utilizar la especificación diseñada para este TP, no la solución del TPE!**.
3. **Importante: Es condición necesaria que esa implementación pase todos los casos de tests provistos en el directorio tests. Estos casos sirven de guía para la implementación, existiendo otros TESTS SUITES *secretos* en posesión de la cátedra que serán usados para la corrección.**

4. En caso que la cobertura de líneas no sea del 100 %, extender el conjunto de casos de tests con nuevos casos de tests hasta alcanzar una cobertura de líneas de 100 %, o explicar el motivo por el que esa cobertura no puede ser alcanzada. La cobertura debe estar chequeada con herramientas que se verán en laboratorio de la materia.

5. Especificación

proc esGrabacionValida (in a: audio, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, out result: Bool) {
    Pre $\{True\}$
    Post $\{result = true \leftrightarrow audioValido(a, prof, freq)\}$
}

proc elAcaparador (in m: sala, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, out persona: $\mathbb{Z}$) {
    Pre $\{salaValida(m, prof, freq) \wedge hayUnicoAcaparador(m, prof, freq)\}$
    Post $\{0 \leq persona < |m| \wedge_L acapara(m, persona, prof, freq)\}$
    pred hayUnicoAcaparador (m: sala, prof: $\mathbb{Z}$, freq: $\mathbb{Z}$) $\{|m| > 0 \wedge_L (\exists p : \mathbb{Z})\ 0 \leq p < |m| \wedge_L acapara(m, p, prof, freq)\}$
    pred acapara (m: sala, p: $\mathbb{Z}$, prof: $\mathbb{Z}$, freq: $\mathbb{Z}$) {
        $(\forall x : \mathbb{Z})\ 0 \leq x < |m| \wedge x \neq p \longrightarrow_L intensidadMedia(m[x]) < intensidadMedia(m[p])\}$
}

proc ardillizar (in m: sala, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, out result: sala) {
    Pre $\{salaValida(m, prof, freq) \wedge esValidaAlSacarImpares(m, freq)\}$
    Post $\{|m| = |result| \wedge_L salaArdillizada(result, m)\}$
    pred esValidaAlSacarImpares (m: sala, freq: $\mathbb{Z}$) $\{(\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L duraMasDe(m[a], freq, 2)$
        $\wedge \neg hayCerosEnPosPares(m[a], freq)\}$
    pred hayCerosEnPosPares (a: audio, freq: $\mathbb{Z}$) $\{(\exists i, j : \mathbb{Z})\ 0 \leq i, j < |a| \wedge_L (duraMasDe(2, subseq(a, i, j), freq)$
        $\wedge todosCerosEnPosPares(subseq(a, i, j)))\}$
    pred todosCerosEnPosPares (a: audio) $\{(\forall i : \mathbb{Z})\ 0 \leq i < |a| \wedge i\ mod\ 2 = 0 \longrightarrow_L a[i] = 0\}$
    pred salaArdillizada (m: sala, n: sala) $\{(\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L audioArdillizado(m[a], n[a])\}$
    pred audioArdillizado (a: audio, a_0: audio) $\{|a| = \lfloor(|a_0| + 1)/2\rfloor \wedge_L (\forall i : \mathbb{Z})\ 0 \leq i < |a| \longrightarrow_L a[i] = a_0[2 * i]\}$
}

proc flashElPerezoso (in m: sala, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, out result: sala) {
    Pre $\{salaValida(m, prof, freq) \wedge esValidaAlInterpolar(m, freq)\}$
    Post $\{|m| = |result| \wedge_L salaInterpolada(result, m)\}$
    pred esValidaAlInterpolar (m: sala, freq: $\mathbb{Z}$) $\{(\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L duraMasDe(0.5, m[a], freq)$
        $\wedge\ \neg existenTodosCeros(m[a])\}$
    pred existenTodosCeros (a: audio) $\{(\exists i, j : \mathbb{Z})\ 0 \leq i, j < |a| \wedge duraMasDe(0.5, subseq(a, i, j), freq)$
        $\wedge\ sonTodosCeros(subseq(a, i, j))\}$
    pred salaInterpolada (m: sala, n: sala) $\{(\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L audioInterpolado(m[a], n[a])\}$
    pred audioInterpolado (a: audio, a_0: audio) $\{|a| = 2 * |a_0| - 1 \wedge_L$
        $((\forall i : \mathbb{Z})\ (0 \leq i < |a| \wedge i\ mod\ 2 = 1 \longrightarrow_L \lfloor a[i-1] + a[i+1])/2\rfloor = a[i])) \wedge$
        $(\forall i : \mathbb{Z})\ 0 \leq i < |a_0| \longrightarrow_L a_0[i] = a[2 * i]\}$
}

proc silencios (in a: audio, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, in umbral: amplitud, out tiempos: $seq\langle intervalo\rangle$) {
    Pre $\{audioValido(a, prof, freq)\}$
    Post $\{intervalosValidos(tiempos, duracion(a, freq)) \wedge_L$
        $(\forall inter : intervalo)$
            $inter \in tiempos \leftrightarrow esSilencio(a, inter, freq, umbral)\}$
}

proc hayQuilombo (in m: sala, in prof: $\mathbb{Z}$, in freq: $\mathbb{Z}$, in umbral: amplitud, out result: Bool) {
    Pre $\{salaValida(m, prof, freq)\}$
    Post $\{result = true \leftrightarrow (\exists p1, p2 : \mathbb{Z})\ 0 \leq p1 < p2 < |m| \wedge_L \neg seRespetan(m, p1, p2, freq, umbral)\}$
    pred seRespetan (m: sala, p1: $\mathbb{Z}$, p2: $\mathbb{Z}$, freq: $\mathbb{Z}$, umbral: amplitud) {
        $(\forall i : \mathbb{Z})\ 0 \leq i < |m[p1]| \longrightarrow_L$
            $haySilencioQueLoContiene(m[p1], i, freq, umbral) \vee haySilencioQueLoContiene(m[p2], i, freq, umbral)\}$
}

proc sinSilencios (in a: audio, freq: $\mathbb{Z}$, prof: $\mathbb{Z}$, umbral: amplitud, out result: audio) {
    Pre $\{audioValido(a, prof, freq) \wedge esValidoAlSacarSilencios(a, freq, umbral)\}$
    Post $\{|result| = |a| - cantSilencios(a, freq, umbral, |a| - 1) \wedge_L$
        $(\forall i : \mathbb{Z})(\ 0 \leq i < |a| \wedge_L \neg haySilencioQueLoContiene(a, i, freq, umbral)) \longrightarrow_L$
            $a[i] = result[i - cantSilencios(a, freq, umbral, i)]\}$
    pred esValidoAlSacarSilencios (a: audio, freq: $\mathbb{Z}$, umbral: $\mathbb{Z}$) {
        $enSegundos(|a| - cantSilencios(a, freq, umbral), freq) > 1.0\}$
    fun cantSilencios (a: audio, freq: $\mathbb{Z}$, umbral: $\mathbb{Z}$, hasta: $\mathbb{Z}$) : $\mathbb{Z}$ =
        $\sum_{i=0}^{hasta}$ if $haySilencioQueLoContiene(a, i, freq, umbral)$ then 1 else 0 fi ;

}

proc encontrarAparicion (in s: audio, in target: audio, in freq: $\mathbb{Z}$, in prof: $\mathbb{Z}$, out res: $\mathbb{Z}$) {
    Pre {$audioValido(s, prof, freq) \ \wedge \ audioValido(target, prof, freq) \ \wedge |s| \geq |target|$}
    Post {$res = comienzoCorrelacion(target, s)$)}
}

proc medirLaDistancia (in m: sala, in frase: audio, in freq: $\mathbb{Z}$, in prof: $\mathbb{Z}$, out res: ($\mathbb{Z}$, $seq\langle\mathbb{R}\rangle$)) {
    Pre {$salaValida(m, prof, freq) \ \wedge_L \ alguienLoDijo(m, frase) \ \wedge$
        $(\forall a : audio)\ 0 \leq a < |m| \longrightarrow_L hayMaximaCorrelacion(m[a], frase)$}
    Post {$0 \leq res_0 < |m| \ \wedge_L \ esElQueLoDijo(res_0, m, frase) \ \wedge \ |res_1| = |m| \ \wedge_L$
        $sonLasDistanciasAP(m, res_0, res_1, frase, freq)$}
    pred alguienLoDijo (m: sala, frase: audio) {$(\exists p : \mathbb{Z})\ 0 \leq p < |m| \wedge_L esElQueLoDijo(p, m, frase) \ \wedge$
        $loDijoAntesQueTodos(m, p, frase)$}
    pred esElQueLoDijo (p: $\mathbb{Z}$, m: sala, frase: audio) {$(\forall x : \mathbb{Z})\ 0 \leq x < |m| \ \wedge \ x \neq p \longrightarrow_L$
        $intensidadCorrelacion(m[x], frase) < intensidadCorrelacion(m[p], frase)$}
    pred loDijoAntesQueTodos (m: sala, p: $\mathbb{Z}$, frase: audio) {$(\forall x : \mathbb{Z})\ 0 \leq x < |m| \ \wedge \ x \neq p \longrightarrow_L$
        $comienzoCorrelacion(m[x], frase) > comienzoCorrelacion(m[p], frase)$}
    fun intensidadCorrelacion (a: audio, frase: audio) : $\mathbb{R}$ =
        $\sum_{i=0}^{|a|-|frase|}$ if $esMaximaCorrelacion(a, i, frase)$ then $intensidadMedia(subseq(a, i, i + |frase|))$ else 0 fi;
    pred sonLasDistanciasAP (m: sala, p: $\mathbb{Z}$, distancias: $seq\langle\mathbb{R}\rangle$, frase: audio, freq: $\mathbb{Z}$) {
        $(\forall i : \mathbb{Z})\ 0 \leq i < |distancias| \longrightarrow_L distancias[i] = distanciaAP(m, i, p, freq, frase)$}
    fun distanciaAP (m: sala, p1: $\mathbb{Z}$, p2: $\mathbb{Z}$, freq: $\mathbb{Z}$, frase: audio) : $\mathbb{R}$ =
        $abs(comienzoCorrelacion(m[p1], frase) - comienzoCorrelacion(m[p2], frase)) * velSonido/freq$;
    fun velSonido : $\mathbb{R}$ = $343,2$;
}

proc enmascarar (in dur : tiempo, in tiempos : $seq\langle intervalo\rangle$, out mascara: $seq\langle$Bool$\rangle$) {
    Pre {$intervalosValidos(tiempos) \wedge (conPrecision(dur, 2) \wedge dur > 0$}
    Post {$|mascara| = 100 * dur) \wedge_L (\forall i : \mathbb{Z})\ 0 \leq i < |mascara| \longrightarrow_L mascara[i] = (\exists t : intervalo)\, t \in tiempos \wedge t_0 \leq$
        $tiempoEnPosicion(i) < t_1$}
    fun tiempoEnPosicion (i : $\mathbb{Z}$) : tiempo = $i/100$;
}

proc negacionLogica (inout mascara: $seq\langle$Bool$\rangle$) {
    Pre {$mascara_0 = mascara$}
    Post {$|mascara_0| = |mascara| \wedge_L (\forall i : \mathbb{Z})\ 0 \leq i < |mascara| \longrightarrow_L mascara[i] = \neg mascara_0[i]$}
}

5.1. Auxiliares generales

pred audioValido (a: audio, prof: $\mathbb{Z}$, freq: $\mathbb{Z}$) {
    $freqValida(freq) \wedge_L$ (
    $enRango(a, prof)$
    $\wedge\, profValida(prof)$
    $\wedge\, micFunciona(a, freq)$
    $\wedge\, duraMasDe(1.0, a, freq)$)
}
pred intervalosValidos (intervalos : $seq\langle intervalo\rangle$, durTotal : tiempo) {
    $intervalosEnRango(intervalos, durTotal)$
    $\wedge\, enOrden(intervalos)$
    $\wedge\, todosConPrecision(intervalos, 2)$}
pred intervalosEnRango (intervalos : $seq\langle intervalo\rangle$, durTotal : tiempo) {
    $(\forall i : intervalo)\ intervaloEnRango(i, durTotal)$}
pred intervaloEnRango (i : intervalo, durTotal : tiempo) {$0 \leq i_0 < i_1 < durTotal$}
pred enOrden (intervalos : $seq\langle intervalo\rangle$) {
    $(\forall i, j : \mathbb{Z})\ 0 \leq i < j < |intervalos| \longrightarrow_L intervalos[i]_1 < intervalos[j]_0$}
pred todosConPrecision (intervalos : $seq\langle intervalo\rangle$, precision : $\mathbb{Z}$) {
    $(\forall p : intervalo)\ p \in intervalos \longrightarrow_L (conPrecision(p_0, precision) \wedge conPrecision(p_1, precision)$}
pred conPrecision (t : tiempo, p : $\mathbb{Z}$) {$\lfloor t * 10^p \rfloor / 10^p = t$}
pred freqValida (freq: $\mathbb{Z}$) {$freq \geq 4$}
pred enRango (a: audio, prof: $\mathbb{Z}$) {$(\forall\ x : \mathbb{Z})\ x \in a \longrightarrow -2^{(prof-1)} \leq x \leq 2^{(prof-1)} - 1$}
pred profValida (prof: $\mathbb{Z}$) {$prof = 16 \vee prof = 32$}
pred micFunciona (a: audio, freq: $\mathbb{Z}$) {$\neg(\exists i, j : \mathbb{Z})\ duraMasDe(1, subseq(a, i, j), freq) \ \wedge \ sonTodosCeros(subseq(a, i, j))$}

pred sonTodosCeros (s: $seq\langle\mathbb{Z}\rangle$) $\{(\forall i : \mathbb{Z})\ i \in s \longrightarrow i = 0\}$
pred salaValida (m: sala, prof: $\mathbb{Z}$, freq: $\mathbb{Z}$) $\{esMatriz(m) \wedge (\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L audioValido(m[a], prof, freq)\}$
pred esMatriz (m: sala) $\{(\forall a : \mathbb{Z})\ 0 \leq a < |m| \longrightarrow_L |m[a]| = |m[0]|\}$
pred esSilencio (a: audio, inter : intervalo, freq: $\mathbb{Z}$, umbral: amplitud) {
$intervaloEnRango(inter, duracion(a, freq) \longrightarrow_L$
$(inter_1 - inter_0 > 0.1 \wedge conPrecision(inter_0, 2) \wedge conPrecision(inter_1, 2)) \wedge_L$
$(\forall k : \mathbb{Z})\ indiceEnTiempo(inter_0, freq) \leq k \leq indiceEnTiempo(inter_1, freq) \longrightarrow_L$
$abs(a[k]) < umbral$
$\wedge\ indiceEnTiempo(inter_0, freq) \neq 0 \longrightarrow_L a[indiceEnTiempo(inter_0, freq) - 1] \geq umbral$
$\wedge\ indiceEnTiempo(inter_1, freq) \neq |a| - 1 \longrightarrow_L a[indiceEnTiempo(inter_1, freq) + 1] \geq umbral$
}
pred haySilencioQueLoContiene (a : audio, i : $\mathbb{Z}$, freq : $\mathbb{Z}$, umbral : $\mathbb{Z}$) {
$(\exists inter : intervalo)(intervaloEnRango(inter, duracion(a, freq)) \wedge_L$
$inter_0 \leq enSegundos(i, freq) < inter_1 \wedge$
$esSilencio(a, inter, freq, umbral)\}$
fun intensidadMedia (a: audio) : $\mathbb{R}\ = \sum_{i=0}^{|a|-1} abs(a[i])/|a|$;
fun indiceEnTiempo (t : tiempo, freq : $\mathbb{Z}$) : $\mathbb{Z}\ = \lfloor freq * t \rfloor$;
pred duraMasDe (t: tiempo, a: audio, freq: $\mathbb{Z}$) $\{duracion(a, freq) > t\}$
fun enSegundos (n : $\mathbb{Z}$, freq : $\mathbb{Z}$) : tiempo $= n/freq$;
fun duracion (a : audio, freq : $\mathbb{Z}$) : tiempo $= enSegundos(|a|, freq)$;
pred hayMaximaCorrelacion (a: audio, frase: audio) $\{(\exists i : \mathbb{Z})\ 0 \leq i < |a| - |frase| \longrightarrow_L esMaximaCorrelacion(a, i, frase)\}$
pred esMaximaCorrelacion (a: audio, startPoint: $\mathbb{Z}$, frase: audio) $\{(\forall i : \mathbb{Z})\ 0 \leq i < |a| - |frase|\ \wedge\ i \neq startPoint \longrightarrow_L$
$correlacion(subseq(a, i, i + |frase|), frase) < correlacion(subseq(a, startPoint, startPoint + |frase|), frase)\}$
fun comienzoCorrelacion (a: audio, frase: audio) : $\mathbb{Z}\ = \sum_{i=0}^{|a|-|frase|}$ if $esMaximaCorrelacion(a, i, frase)$ then i else 0 fi ;