SINTAXIS DE ER PARA PRÁCTICAS DE ALF

Contenido

1 Sintaxis extendida de ER en aplicaciones

La manera de escribir expresiones regulares varía de unas aplicaciones a otras, aunque hay un núcleo común en las distintas variantes de sintaxis de las expresiones regulares (*regex flavors*) . Vamos a introducir la **sintaxis tipo Perl,** que es una extensión de la sintaxis teórica que se estudia en el tema 3. La sintaxis que se usa en algunos programas como *egrep, vi* o *flex* que forman parte de la distribución de Linux es algo diferente.

Motor de ER (regex engine): es un módulo integrado en una herramienta o lenguaje de programación que permite usar expresiones regulares para realizar operaciones de *regular pattern matching*, como validación de formato de cadenas, búsqueda o sustitución usando un patrón dado por una ER. Internamente el *regex engine* compila o traduce la ER en una implementación de autómata finito cuya tabla de transición se usa en los métodos para procesar las cadenas. Dicha implementación es transparente al usuario/programador, que se ocupa sólo de escribir la ER y de aplicar los métodos que proporciona el *regex engine* para resolver los problemas de procesamiento de cadenas de patrón regular.

1.1 Funcionamiento básico de EditPad Pro

Para probar la sintaxis de ER usaremos el editor de texto EditPad Pro que puede descargarse de:

http://www.editpadpro.com/download.html (free trial)

- En recursos de Aula Virtual, carpeta:

  Laboratorios de ejercicios-> Ficheros de prueba-EditPad

  incluimos ficheros para cargar en el editor *EditPad Pro* y probar ejemplos de prueba de sintaxis de ER y operaciones de búsqueda y sustitución con expresiones regulares.

- Carga, por ejemplo el fichero *prueba-EditPad-varios.html*

- Selecciona en el menú *Search* la opción *Show Search Panel.*
- Selecciona en el panel *Search&Replace* las casillas *Regular Expression y Case Sensitive.*
- Con *Find First* busca en el fichero cargado **la primera cadena del texto S coincidente con la expresión regular R** introducida en *Search* (primera cadena S tal que *R casa con S*) y la resalta en gris. Con *Next* busca la siguiente cadena coincidente **desde la posición en que finalizó el *matching* anterior**. Si en el texto no hay ninguna cadena que se ajuste al patrón de R, el editor lo advierte con una cruz roja.
- En la barra de *Search&Replace* se puede activar *Highlight*. Esto hace que se resalten **todas las cadenas coincidentes**, como si se hiciera *Next* de forma repetida. El resalte en algunos casos puede resultar confuso si se solapan las coincidencias.
- Precaución con espacios en blanco: en el panel *Search&Replace* hay que tener cuidado de no poner espacios en blanco y otros caracteres que no sean parte del patrón de laER, sobre todo al hacer *copy-paste* de Word a EditPad. Los blancos los señala con un punto gris claro para evitar confusión.

1.2 Tabla de sintaxis extendida

A continuación introducimos un resumen de la sintaxis extendida de ER (no se incluye toda la sintaxis completa, ver referencias para más información). Los ejemplos indicados pueden probarse con el fichero *Ficheros de prueba-EditPad → prueba-EditPad-varios.html*

Expresiones regulares que describen un único carácter o un conjunto de caracteres		
Sintaxis	**Descripción**	**Ejemplos de emparejamiento** **(Expresión regular en azul y cadena literal en amarillo)**
c	**c** representa un **carácter literal**, que es cualquier carácter que no sea uno de los siguientes **caracteres especiales** para la sintaxis [ ] \ ^ $. | ? * + ( ) { }	a casa con a Una ER que consiste en una secuencia de caracteres literales casa exactamente con la cadena que forma la ER: hola casa con hola (y ninguna más en ***modo case sensitive***, que diferencia mayúsculas de minúsculas) hola no casa con holaMundo HoLA
\e	**e** representa un carácter especial (listados arriba) y al poner **\e,** la barra de escape hace que el carácter especial **e** pierda su significado y entonces \e describe al carácter literal **e**.	8\+2 casa con 8+2 (y ninguna más) 8+2 no casa con 8+2
\n \r \t	ERs para describir caracteres de LF (*line feed*), CR (*carrige return*) y tabuladores, respectivamente.	La ER \r\n describe a un *newline* en Windows y \n a un *newline* en Unix/Linux
[secCar]	**[secCar]** describe un **conjunto de caracteres** literales. La ER casa con cualquier carácter individual que aparezca en la secuencia de caracteres **secCar.** Los caracteres especiales pierden su significado dentro de los corchetes, pasan a ser literales y no hay que ponerles la barra de escape.	[hola] casa con h (también los caracteres o l a y ninguno más) [hola] no casa con hola [hH][oO][Ll][aA] casa con hola y también con HoLa y sólo con la palabra "hola" con letras mayúsculas o minúsculas. [+*] casa con + (también * y ninguno más)
[^secCar]	Al poner ^ (gorro o caret) se complementa el conjunto de caracteres que va detrás con respecto al conjunto total de caracteres .	[^+*] casa con cualquier carácter que NO sea + ni *
[**c1-c2**]	**[c1-c2]** describe un **conjunto de caracteres dado por un rango** y casa con cualquier carácter individual cuyo	[0-9] equivale a la ER (0|1|2|3|4|5|6|7|8|9) que casa con cualquier dígito decimal [^0-9] casa con cualquier carácter que NO sea un dígito

	código ASCII esté comprendido entre el código del carácter **c1** y el código del carácter **c2**. Pueden aparecer varios rangos dentro de los corchetes y se puede complementar el rango con ^	[a-z] casa con cualquier letra minúscula ASCII [a-zA-Z] es una ER con dos rangos y casa con una letra mayúscula o minúscula ASCII [a-z][a-z][0-9] casa pe2 zb7 no casa con e2 variable1
\d	La ER \d es una abreviatura (*sorthand*) que describe al conjunto de los **dígitos decimales**.	\d equivale a [0-9] \d\d\d casa con 885 no casa con 12
\w	La ER \w es una abreviatura que describe al conjunto de los ***caracteres tipo word (palabra):*** son los dígitos decimales, letras ASCII y _ (subrayado).	\w equivale a [0-9a-zA-Z_] \w[\d.,] casa con h1 r, 9. (las abreviaturas pueden usarse dentro de corchetes)
\p{L} \p{Lu} \p{Ll}	\p{L} describe al **conjunto de letras Unicode** (mayúsculas o minúsculas). Permite que las letras acentos, diéresis, signos fonéticos…. \p{Lu} sólo para letras Unicode mayúsculas \p{Ll} sólo para letras Unicode minúsculas	La ER [a-zA-Z]+ no casa con ambigüedad ni con España ni con fácil. \p{L}+ casa con cualquiera de las anteriores. \p{Ll}+ casa con fácil no casa con España \p{Lu}+ casa con ESPAÑA no casa con España
\s	La ER \s es una abreviatura que describe el conjunto de ***caracteres de espaciado*** (blanco, \t, \r o \n). Equivale a [\n\r\t]	hola[\d\s]mundo casa con hola1mundo hola mundo no casa con holamundo
.	El punto (dot) es una ER que casa con **cualquier carácter *excepto* los caracteres de nueva línea** \r o \n	car. casa con cara caro car1 car. car+ car\. casa con car. (y ninguna cadena más)
Expresiones regulares con operador de alternancia y paréntesis de agrupación (*alternation, grouping*)		
R1|R2	El | (pipe) es el operador de **unión o alternacia de ER**. La expresión **R1|R2** tiene coincidencia con cadenas que se ajustan a R1 o a R2. Es el operador de menor predecencia y con los paréntesis se agrupa una subexpresión para alterar la precedencia.	America|Europe|Canada casa con America Europe Canada (y ninguna cadena más) (hola|bye)Mundo casa con holaMundo byeMundo (y ninguna más) (hola|bye)Mundo equivale por la propiedad distributiva a holaMundo|byeMundo mejor la primera
Expresiones regulares con operadores de opcionalidad y repetición (*quantifiers*)		
R*	La estrella * es el **operador de clausura**. R* indica **cero o más repeticiones** de cadenas que casan con el patrón R	[a-z]\d* casa con a1234 b2 c no casa con 12 d-83
R*?	*? es el operador de **clausura perezosa** (***lazy star***). Cuando se usa R*? en una	Buscando con <.*> en una línea que contiene CHARACTER SET y sin más ángulos > en línea,

	búsqueda, el *regex engine* selecciona la cadena más corta que se ajusta al patrón R*.	se selecciona <h3> CHARACTER SET </h3> Buscando con <.*?> selecciona <h3> (hasta el primer ángulo de cierre).
R+	Se corresponde con el **operador de clausura positiva** de lenguajes. R+ indica **una o más repeticiones** del patrón R. Equivale a RR*	[a-z]\d+ casa con a1234 b2 no casa con c 12 d-83
R+?	+? es versión perezosa del operador + (***lazy plus***). Análogo a R*? pero obligando al menos una coincidencia con R	Buscando con <.+?> en una línea que contiene <> </br> y sin más > en la línea, se selecciona <> </br> Si se busca con <.*?> entonces se selecciona <>
R?	El operador ? (no precedido de * ni +) es un **operador de opcionalidad** e indica que R es opcional . R? equivale a (R|λ)	abc? casa con ab abc (y ninguna cadena más) a(bc)? casa con a abc (y ninguna cadena más)
R{n} **R{n,m}** **R{n,}**	Indica **n repeticiones** del patrón R Indica **entre n y m repeticiones** de R Indica **n o más repeticiones** de R	a{3} casa con aaa (y ninguna cadena más) v\d{1,3} casa con v5 v60 v123 no casa con v v8965 [A-Z]{3}\d{2,} casa con BXZ47 ABC123 RSA1234 no casa con B43 576
ER con anclas y delimitadores de palabras *(anchors, word boundaries)*		
^R	EL ^ cuando va fuera de corchetes tiene coincidencia con la **posición de comienzo de una cadena o línea de texto.** Cuando se usa ^R hay matching con una cadena S que se ajusta al patrón R y aparece como prefijo.	Buscando con ^<[a-zA-Z]+> se selecciona en una línea que comience por Si antes del carácter '<' hay espacios u otro carácter entonces no hay *matching* en esa línea.
R$	Análogo al anterior pero con la **posición de final de cadena o línea de texto.**	Buscando con <[a-zA-Z]+>$ se selecciona <head> en una línea que acaba por <head> Buscando con la ER ^\d+$ selecciona cadenas de uno o más dígitos que ocupan toda la línea.
\bR\b	\b es un **delimitador de palabra** y en una búsqueda con \bR\b se selecciona una cadena que casa con R y está **delimitada a la izquierda y a la derecha por caracteres que NO son letras o dígitos.**	Buscando con \b\d+\b se selecciona 8859 en una línea que contiene charset=ISO-8859- Si fuera charset=ISO8859- no habría matching con 8859 Se usa el ancla \b para emparejamiento con palabras completas.

1.3 Modificadores de expresiones regulares

Los modificadores de las expresiones regulares sirven para alterar el modo en que el regex engine de cierta herramienta realiza las operaciones de *pattern matching* a partir de una expresión regular y de ese modo se puede acortar la expresión regular. Sólo deben usarse cuando realmente tengan algún efecto, no de forma indiscriminada, pues disminuyen la eficiencia.

Modificador para modo "dot matches newline"

En métodos con ER en Python y en editores de texto como EditPad Pro (ver sección de funcionamiento de EditPad más adelante),

el punto (*dot*) no tiene coincidencia por defecto con los caracteres de nueva línea (como \r o \n).
Ejemplo: si se usa la ER </head>.*<BODY> para buscar en un texto donde aparecen las etiquetas html </head> y luego <BODY> separadas por newlines entonces el *regex engine* no encuentra ninguna cadena coincidente.
→Para que el punto tenga coincidencia con caracteres de nueva línea (modo *dot matches newline)*, sin que dependa de la herramienta concreta, se usa el *modificador de dot* **(?s)** delante de laER.
Ejemplo: con (?s)</head>.*<BODY> se selecciona desde </head> hasta <BODY> incluido, aunque esas etiquetas html vayan en líneas separadas.

Modificador para modo "case insensitive"

El modo por defecto en los métodos con ER en Python es *case sensitive*, esto es, se distingue entre mayúsculas y minúsculas en las letras en las operaciones de matching.
→Para activar el modo contrario *case insensitive* para una ER sin que dependa de la herramienta concreta se pone el *modificador de case* **(?i)** delante de laER.

Ejemplo (?i)hola casa con HoLA hola HOLA (no se distinguen mayúsculas de minúsculas)
Los modificadores pueden juntarse, se puede poner **(?is)R** para indicar modo *insensitive* y *dot matches newlines* para la expresión regular R que va detrás de los modificadores.

Ejemplo: Con (?is)</head>.*<body> se selecciona desde la etiqueta </head> hasta <body> incluido, aunque esas etiquetas vayan en líneas separadas y con independencia de que las palabras de etiqueta "head" y "body" vayan con las letras en mayúsculas o minúsculas. Tiene sentido poner el modificador de case porque en el lenguaje Html no se diferencian las mayúsculas de las minúsculas en las etiquetas. La búsqueda también tendría éxito si, por ejemplo, la palabra clave "head" aparece en el documento escrita como "Head" o "HEAD".

1.4 Capturas y referencias a capturas (backreferences)

Cada una de las subexpresiones **entre paréntesis** que aparecen dentro de una expresión regular forma un **grupo**. Los grupos se enumeran desde 1 contando los paréntesis de apertura en una ER de izquierda a derecha y el grupo 0 siempre es la propia expresión regular, aunque no vaya entre paréntesis.
Ejemplo
En la ER ([A-Z]+)(\d+)([a-z]+) aparecen paréntesis que no son necesarios para alterar la precedencia, pero al ponerlos se distinguen 4 grupos en la expresión regular:

- El **grupo 0** siempre es la propia expresión regular, en este caso ([A-Z]+)(\d+)([a-z]+)
- El **grupo 1** consiste en la subexpresión regular: ([A-Z]+)
- El **grupo 2** es: (\d+)
- El **grupo 3** es: ([a-z]+)

Los grupos de una ER pueden ser usados para que el *regex engine* haga una captura de las subcadenas que se ajustan al patrón de cada grupo, proceso al que nos referimos de forma abreviada como **captura de grupos**. Las capturas **pueden referenciarse** mediante la referencia **\i** donde i es el número del grupo, por ej. \1 (captura de grupo 1), \3 (captura de grupo 3), o bien \0 para la captura de la cadena que se ajusta a la ER completa, si la ER no va entre paréntesis.

¿Cómo se usan las referencias a las capturas para hacer sustituciones con ER?

Una de las aplicaciones del proceso de captura de grupos de una ER es para realizar una sustitución que no es por cadena fija (la típica), sino que se sustituye por una cadena que depende del patrón de la ER.
Ejemplo
Supongamos que queremos buscar una cadena que se ajusta al patrón ([A-Z]+)(\d+)([a-z]+) para sustituirla por la cadena que consiste en las letras minúsculas de la cadena original seguidas de un guión, luego los dígitos encerrados entre corchetes, seguido de un guión y finalmente las letras mayúsculas. Por ejemplo para sustituir ABCD123xyz por xyz-123-ABCD.
Obviamente esto no se puede hacer con un *replace* o sustitución simple como se hace con los procesadores de texto de uso general. En el editor EditPad, que es un editor profesional para programadores y permite usar ER, se pondría lo siguiente en el panel *search&replace*:
SEARCH: ([A-Z]+)(\d+)([a-z]+)
REPLACE: \3-\2-\1
Con *next+replace* sucesivas veces podemos sustituir cada cadena coincidente con la ER del campo *Search*. Con *Replace All* se sustituyen de golpe todas las cadenas coincidentes.

Nota: toda la cadena que se usa para reemplazar es literal, excepto las referencias a las capturas con \i.

Las referencias también se pueden usar dentro de una ER para búsquedas más complejas

Las referencias permiten construir **expresiones regulares aumentadas más potentes** (describen lenguajes que no son regulares). Este tipo de expresiones regulares aumentadas no es traducible a un autómata finito puro, porque se requiere guardar en memoria la captura (parte de la cadena que se procesa) y el modelo de autómata finito es el de una máquina sin memoria.

Ejemplo

Supongamos que queremos buscar en un texto dos palabras repetidas, entendiendo que son cadenas iguales formadas por letras Unicode mayúsculas o minúsculas y las palabras van separadas por uno o más caracteres de tipo espacio (\s).

Por ejemplo, HOLA HOLA debería seleccionase y da lo mismo que vayan en la misma línea o en líneas separadas y que una de las palabras lleve minúsculas. También requerimos que se seleccionen **palabras completas**, no fragmentos de palabras. Por ejemplo, si en una línea aparece Fichero de ejemplo entonces no se debería resaltar Fichero de ejemplo porque la 'e' forma parte de una palabra, no es una palabra completa. Esto indica que debemos usar el delimitador de palabra \b. En EditPad pondríamos la siguiente ER para la búsqueda

SEARCH: \b(\p{L}+)\b\s+\b\1\b (\s también tiene coincidencia con caracteres de nueva línea)

1.5 Ejemplos para practicar con ER en EditPad

En las siguientes preguntas se pide una ER que debe describir exactamente al tipo de cadenas que se indican en el enunciado. **No debe ser demasiado general** (no debe casar con cadenas que no tienen el formato deseado) **ni demasiado estricta** (debe casar con todas las cadenas consideradas correctas, aunque no aparezcan en el texto).

- En *Seach&Replace* del editor EditPad selecciona por defecto *Regular Expression* y *Case Sensitive.*
- Abre en EditPad el fichero *prueba-EditPad-Varios.html*

1. Expresión regular para buscar la cadena literal |2^3*2=16|
   ER:

2. Indica dos posibles ER para buscar la cadena um.es o la cadena dif.um.es, una con el operador | y otra (mejor) sin usar el operador |
   ER1: um\.es|dif\.um\.es
   ER2:

3. Indica ER para buscar cadenas de 4 o más dígitos, una de ellas con rangos y otra con llaves de repetición
   ER1: [0-9][0-9][0-9][0-9]+
   ER2:

4. Supongamos que queremos comprobar si en un texto aparece una abreviatura de "expresión regular" en español o "regular expression" en inglés. Las abreviaturas que se suelen usar son las siguientes:

   Regex regex regEx RegEx ER

Proporciona una ER lo más corta posible que sirva para buscar cualquiera de esas abreviaturas (y sólo esas). Además queremos buscarlas como palabras completas. Por ejemplo, no queremos que se seleccione ER si aparece en LITERAL.

   ER:

5. Queremos comprobar si en un texto aparece una palabra formada por 3 o más letras Unicode minúsculas a principio de línea. Indicar la ER adecuada para esa búsqueda:
   ER: ^\p{Ll}{3,}

6. Queremos comprobar si en un texto aparece una palabra formada por 3 o más letras Unicode mayúsculas al final de línea. Indicar la ER adecuada para esa búsqueda:
   ER:

7. Obtener ER para buscar cualquier secuencia de dos o más números separados por comas o blancos. Cada número es una secuencia de uno o más dígitos.
   ER:

8. Una cadena con formato de **etiqueta Html simple** es del tipo **<idEtiq>** o **</idEtiq>** donde **idEtiq** es un identificador de etiqueta que está formado por secuencia de letras ASCII mayúsculas o minúsculas seguida opcionalmente de una secuencia de dígitos, como en
 <h1> sin espacios dentro de los ángulos. Encuentra una ER lo más corta posible que sirva para describir cadenas tipo etiqueta Html simple (aunque algunas no se correspondan con etiquetas que se usen realmente en el lenguaje Html).
   ER:

9. Obtén una ER que sirva para buscar cadenas tipo **texto html etiquetado simple.** Una cadena de este tipo empieza por una etiqueta html simple de inicio, que es del tipo **<idEtiq>**, seguido de una secuencia opcional de caracteres y acaba por la correspondiente etiqueta simple de fin, que es **</idEtiq>** (no necesariamente en la misma línea). Por ejemplo, en el fichero se debería seleccionar el texto etiquetado que va desde hasta incluido, pero no desde hasta </h1>. El texto entre dos etiquetas de inicio y fin puede llevar a su vez otras etiquetas. Se requiere además que la búsqueda seleccione hasta la primera etiqueta de fin que aparezca después de la de inicio. Por ejemplo, que no se resalte <h3> ALTERNANCIA </h3> <h3>REPETICIÓN </h3> sino <h3> ALTERNANCIA </h3>

   Hay que usar una referencia a una captura de grupo en la ER, clausura perezosa y modificador de dot.
   ER: (?s)<([a-zA-Z]+\d*)>.*?</\1>

10. Obtén una ER que sirva para buscar texto que empieza por una **etiqueta Html simple de inicio** y acaba por la correspondiente **etiqueta simple de fin**, con el formato de etiquetas como se indica en la pregunta anterior. Pero ahora se requiere que en el texto entre las etiquetas de inicio y fin no aparezcan caracteres de ángulo '<' o '>' y de esa manera se evita que el texto entre etiquetas lleve otras etiquetas.
    ER:

- Abre en EditPad ***prueba-EditPad-formatos***

11. Los comentarios multilínea en C o en Java comienzan por /* y llegan hasta la primera secuencia de cierre */ y pueden ocupar varias líneas. Indica la ER adecuada para buscar este tipo de comentarios.
    ER: (?s)/*.*?*/

12. Los comentarios en Html son como los de C multilínea pero comenzando por <!-- y terminando por --> Queremos convertir con *Replace All* todos los comentarios de C multilínea a comentarios en Html, lógicamente conservando el texto que hay dentro del inicio y fin del comentario.
    SEARCH: (?s)/*(.*?)*/
    REPLACE: <!--\1-->

13. ER (1) para buscar cadenas que vayan entre comillas, sin comillas dentro y en una línea y ER(2) lo mismo pero la cadena entre comillas puede ocupar varias líneas.
    ER (1):
    ER(2):

14. Queremos sustituir con *Replace All* cualquier hora con formato hh:mm:ss con 00<= hh <=23, 00<= mm <=59 y 00<= ss <=59, por el texto hh horas, mm minutos, ss segundos, donde hh, mm, ss son los dígitos

que aparecen en la hora encontrada. Sólo deben sustituirse las horas de formato válido según lo descrito. Conviene activar Hightlight para comprobar la selección antes de sustituir.

SEARCH:

REPLACE:

1.6 Recetas de expresiones regulares comunes (regex recipes)

Las siguientes expresiones regulares describen a cadenas con patrón complejo, que suelen usarse en búsquedas o extracción de datos. Ver **más ejemplos de recetas** en:

http://www.regular-expressions.info/examples.html

http://regexlib.com/Search.aspx

Las siguientes recetas pueden probarse con el fichero ***prueba-Formatos.html***

Direcciones de correo

Las cadenas que representan direcciones de correo electrónico tienen su propia sintaxis según el estándar RFC822. La **sintaxis** establece el formato de lo que debe ir antes de la arroba y lo que debe ir después y es bastante compleja. Una dirección de e-mail es del tipo:

nombreUsuario@nombreDominio

donde **nombreDominio** es una cadena que identifica a la máquina servidor de correo y está formada por una o más cadenas de subdominios separados por punto y termina en un dominio de nivel superior (como el dominio *es*, *org*, *com*, etc). Por ejemplo:

- tony-user@example.com → es sintácticamente correcta aunque posiblemente inexistente
- jefecorreo@correo.dis.ulpgc.es → es válida, con varios subdominios
- webmaster.inf@um → no es correcta, le falta dominio superior
- webmaster.inf@ejemplo..es → no es correcta por los dos puntos seguidos en dominio

La siguiente expresión regular sirve para describir de forma aproximada cadenas de formato **tipo dirección de e-mail**. Sólo respeta de forma aproximada la sintaxis real. Aquí entendemos que en las direcciones de correo la parte **nombreUsuario** es una cadena que empieza por letra ASCII o dígito y sigue una secuencia opcional de este tipo de caracteres junto a algunos signos de puntuación como – + % y el punto. La parte **nombreDominio** es una cadena con, al menos, un subdominio y para el nombre de subdominio y dominio superior se suelen usar los caracteres tipo word (\w, que equivale a [0-9a-zA-Z_]) y el guion, mientras que el punto se reserva para separar los nombres de los subdominios. El dominio de nivel superior siempre debe aparecer y está compuesto de 2 a 6 letras ASCII. En la siguiente ER se indica en diferente color cada parte que describe a los distintos componentes de la dirección de correo:

\w[\w.%+-]*@(\w[\w-]*\.)+[a-zA-Z]{2,6}

nombreUsuario **nombreDominio**

URL de protocolo http o https con nombre de dominio

Un localizador uniforme de recursos (*URL uniform resource locator*), es una cadena que se usa para nombrar y localizar recursos en internet, como páginas web, imágenes, videos, etc.

El URL contiene información sobre el **protocolo** a usar para recuperar los datos (http, https, ftp, telnet, etc.), la dirección IP o el **nombre de dominio** (como en una dirección de correo) , opcionalmente puede aparecer un **puerto TCP** tras el nombre de domino y termina con **la ruta** del recurso, con o sin parámetros. La ruta puede omitirse (y en ese caso un navegador añadiría / al final del URL). Por ejemplo, la siguiente es una URL de protocolo http con puerto TCP 80 y parámetros tras /wiki/

http://es.wikipedia.org:80/wiki/Special:Search?search=regex&go=Go

Igual que en caso de direcciones de correo, una cadena de URL debe ajustarse a una **sintaxis** que también resulta complicada de describir. La siguiente ER sirve para describir de forma aproximada el formato de las **cadenas tipo URL de protocolo http** o **https** (modo seguro) **con nombre de dominio** (no dirección IP). Se indica en diferente color cada parte de la ER que describe a los distintos componentes del URL y se pone modificador de *case insensitive* para no diferenciar mayúsculas de minúsculas.

```
(?i)https?://([\w-]+\.)+[a-z]{2,4}(:\d{1,4})?(/[\w\/#~:.?+=&%@~-]+)?
```

Modif Protocolo Nombre-Dominio Puerto Ruta+parámetros

Direcciones IP

Una **dirección IP** es un grupo de números que identifica usualmente a un ordenador en Internet. En lugar de usar las direcciones IP, que son difíciles de recordar, los usuarios de Internet usamos en su lugar los **nombres de dominio,** como los indicados anteriormente para las direcciones de correo o los URL-http. Los servidores de dominios (**DNS**) traducen los nombres de dominio a direcciones IP y viceversa.

Sintaxis: una dirección IP se compone de 4 números de 1 a 3 dígitos separados por puntos, como 182.51.205.241 Cada uno de los cuatro números debe estar comprendido entre 0 y 255. Las direcciones IP puede describirse de forma exacta mediante la siguiente ER, donde aparecen 3 repeticiones de número seguido de punto y la subexpresión final tras {3} es para el número de más a la derecha. El patrón obliga a que los cuatro números sean entre 0 y 255:

```
((2[0-4]\d|25[0-5]|1\d{2}|\d{1,2})\.){3}(2[0-4]\d|25[0-5]|1\d{2}|\d{1,2})
```