Anexo: Regex

Expresión completa:

T0((?:T(?:\d+))*?)(?:(?:T1((?:T(?:\d+))*?)T3)|(?:T2((?:T(?:\d+))*?)T4))((?:T(?:\d+))*?)(?:(?:T5((?:T(?:\d+))*?)T7)|(?:T6((?:T(?:\d+))*?)T8))((?:T(?:\d+))*?)(?:(?:T10((?:T(?:\d+))*?)T12)|(?:T9((?:T(?:\d+))*?)T11))((?:T(?:\d+))*?)T13((?:T(?:\d+))*?)T14

T0((?:T(?:\d+))*?):

- Busca coincidencia con T0.
- Captura un grupo de transiciones T<dígitos> repetidas, usando **(?:...)** grupo no capturador, **T(?:\d+)** T seguido de uno o más dígitos, ***?** repetición cero o más veces.

(?:(?:T1((?:T(?:\d+))*?)T3) | (?:T2((?:T(?:\d+))*?)T4)):

- **T1((?:T(?:\d+))*?)T3** busca coincidencia con T1 y captura transiciones entre T1 y T3.
- **T2((?:T(?:\d+))*?)T4** busca coincidencia con T2, y captura transiciones entre T2 y T4.

((?:T(?:\d+))*?):

- Captura cualquier grupo T<digitos> entre los bloques mas grandes.

(?:(?:T5((?:T(?:\d+))*?)T7)|(?:T6((?:T(?:\d+))*?)T8)):

- Idem al bloque de T3 & T4 , pero realiza las coincidencias entre T5 y T7, o entre T6 y T8.

((?:T(?:\d+))*?):

- Término intermedio igual al anteriormente explicado.

(?:(?:T10((?:T(?:\d+))*?)T12)|(?:T9((?:T(?:\d+))*?)T11)):

- Igual que los bloques anteriores pero entre T10 y T12, o entre T9 y T11.

((?:T(?:\d+))*?)T13((?:T(?:\d+))*?)T14:

- Captura transiciones antes de T13 y T14

Resumen de grupos de captura:

- Grupo 1: Transiciones opcionales entre T0 y (T1-T3 ó T2-T4).
- Grupo 2: Transiciones opcionales en el camino T1-T3.
- Grupo 3: Transiciones opcionales en el camino T2-T4.
- Grupo 4: Transiciones opcionales después del bloque completo (T1-T3 ó T2-T4) y antes del segundo bloque (T5-T7 ó T6-T8).
- Grupo 5: Transiciones opcionales en el camino T5-T7.
- Grupo 6: Transiciones opcionales en el camino T6-T8.
- Grupo 7: Transiciones opcionales después del bloque (T5-T7 ó T6-T8) y antes del tercer bloque (T9-T11 ó T10-T12).
- Grupo 8: Transiciones opcionales en el camino T9-T11.
- Grupo 9: Transiciones opcionales en el camino T10-T12.

- Grupo 10: Transiciones opcionales después del bloque (T9-T11 ó T10-T12) y antes de T13
- Grupo 11: Transiciones opcionales entre T13 y T14.

Script en Python

El script analiza un .txt que tiene las transiciones en una ejecución completa de nuestra red con 200 invariantes completados, estas transiciones están agrupadas en el archivo en el orden que se van ejecutando, es decir no siguen el orden del invariante.

Para el análisis lee línea a línea el archivo:

```
linea_actual = [archivo.readline(), 0]
```

definido así, linea_actual tiene en la primera posición la línea leída, y en la otra un contador de reemplazos que se realizan, iniciado en 0.

Dentro de un bucle tenemos lo siguiente:

```
resultado = re.subn(
    r'T0((?:T(?:\d+))*?)(?:(?:T1((?:T(?:\d+))*?)T3)|(?:T2((?:T(?:\d+))*?)T4))((?:T(?:\d+))*?)'
    r'(?:(?:T5((?:T(?:\d+))*?)T7)|(?:T6((?:T(?:\d+))*?)T8))((?:T(?:\d+))*?)'
    r'(?:(?:T10((?:T(?:\d+))*?)T12)|(?:T9((?:T(?:\d+))*?)T11))((?:T(?:\d+))*?)T13((?:T(?:\d+))*?)T14',
    r'\g<1>\g<2>\g<3>\g<4>\g<5>\g<6>\g<7>\g<8>\g<9>\g<10>\g<11>',
    linea_actual[0].strip()
)
linea_actual = resultado

if linea_actual[1] == 0:  # Si no hubo cambios, salir del bucle
    break
```

con la función re.susbn() se buscan los grupos de captura (especificados con nuestra regex explicada anteriormente). *Resultado* va a también funcionar como un arreglo de dos dimensiones, en la primera la línea modificada y en la otra el número de reemplazos en cada iteración. Finalmente se chequea si es cero linea_actual, por lo que no se puede modificar más y el bucle termina, es decir, si no queda nada todas las transiciones fueron extraídas siguiendo la regex.

Ejemplo reducido:
Si tenemos en la línea **inicial** T0T1T3 el script puede matchear con el grupo 1 que busca un T0 para el bloque T1-T3 ó T2-T4 (teniendo en cuenta que puede haber también transiciones entre T1-T3). A la **salida** de la iteración la línea queda T1T3, eliminando T0.