MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

Trabajo Fin de Máster

Itinerario: Ingeniería de Software

Especificación y verificación de sistemas de electrificación ferroviaria

Alumno: Rubén González Martín
Director: Juan Antonio Mascarell Estruch
Código Asignatura: 31105151
Convocatoria: Junio
Curso: 2017 – 2018.

MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

Trabajo Fin de Máster

Itinerario: Ingeniería de Software

Especificación y verificación de sistemas de electrificación ferroviaria.

Alumno: Rubén González Martín
Director: Juan Antonio Mascarell Estruch
Código Asignatura: 31105151
Tipo de Trabajo: B

CALIFICACIONES

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 01/06/2018.

Quién suscribe:

Autor(a):Rubén González Martín D.N.I/N.I.E/Pasaporte.:50113450S

Hace constar que es la autor(a) del trabajo:

Especificación y verificación de sistemas de electrificación ferroviaria.

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores, se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo. Rubén González Martín

Impreso TFdM05_Autor. Autorización de publicación
y difusión del TFdM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Juan del Rosal, 16
28040, *Madrid*

Tel: 91 398 89 10
Fax: 91 398 89 09

www.issi.uned.es

RESUMEN

Los sistemas de electrificación ferroviaria son una parte fundamental dentro de las instalaciones y la operativa de las líneas de ferrocarril. De esta forma, los proyectos de electrificación son claves a la hora de acometer trabajos para la realización de nuevas líneas ferroviarias, así como para la modernización de algunas existentes.

Los técnicos proyectistas de electrificación utilizan principalmente los llamados esquemas eléctricos para realizar su trabajo y proyectar sus soluciones. No obstante, el diseño y verificación de los mismos es una labor compleja y tediosa, lo cual implica un gran esfuerzo a los proyectistas para realizar un diseño sin errores. A esta situación hay que añadir la máxima exigencia existente, debido al carácter crítico de los sistemas de electrificación (un error no sólo puede originar grandes pérdidas económicas, sino que además puede producir pérdida de vidas humanas).

De esta forma, en el presente Trabajo Fin de Máster se llevarán a cabo actividades especialmente en dos líneas de trabajo:

- Toma de requisitos de todas las propiedades clave a verificar en los diseños de esquemas eléctricos y se realizará una especificación formal de las mismas.
- Desarrollo de una herramienta para la verificación de esquemas eléctricos ferroviarios.

Así, el trabajo aporta una solución transversal entre el ámbito de la Ingeniería Ferroviaria y el de Ingeniería Informática, aportando soluciones para conseguir por un lado minimizar el esfuerzo de los técnicos a la hora de verificar

los diseños y, por otro lado, neutralizar posibles errores que se pudieran producir en el diseño de sistemas de electrificación de líneas ferroviarias.

LISTA DE PALABRAS CLAVE

Electrificación ferroviaria, esquema eléctrico, especificación, verificación, haskell, programación funcional, línea ferroviaria, grafos, líneas de alta velocidad, líneas convencionales.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1 - INTRODUCCIÓN

El presente Trabajo Fin de Máster se sitúa dentro del ámbito de la Ingeniería Ferroviaria y, más concretamente, dentro de los proyectos de electrificación para la construcción de nuevas líneas ferroviarias, así como para la modernización de otras ya existentes.

Así, dentro de los proyectos de electrificación ferroviaria, el diseño y la verificación de esquemas eléctricos lleva siendo durante años una de las actividades más importantes para lograr el buen funcionamiento de las líneas ferroviarias a proyectar. A esto, hay que añadirle el contexto tan competitivo en el que la industria se mueve en la actualidad, de tal forma que:

- Cada vez son necesarios diseños más precisos, que deriven en un ajuste económico mayor, ya que en la situación económica actual el cliente busca realizar el menor gasto posible para conseguir la funcionalidad deseada.

- El cliente plantea diferentes hipótesis variando condiciones de partida y espera que se proyecten soluciones acordes a esos condicionantes de partida en el tiempo y la forma adecuada.

- A la exigencia del cliente y el alto grado de competitividad en el sector hay que añadir que se está trabajando con sistemas críticos y, por ello, neutralizar los posibles errores es prioritario.

Con el fin de conseguir todo lo anterior se plantean en el presente Trabajo Fin de Máster soluciones apoyadas en el ámbito de la Ingeniería Informática y en el de la especificación formal que permitan realizar los trabajos de una forma más competitiva y minimizando errores, ya que la adecuada especificación de las

propiedades que debe cumplir el diseño del esquema eléctrico de la línea, así como la verificación propiamente dicha a través de la implementación de todos los requisitos especificados formalmente, dotarán al proyectista de herramientas muy valiosas para acometer su trabajo.

2 - DESCRIPCIÓN DEL PROBLEMA

El problema que se desea resolver con la solución planteada es el de abordar la especificación y verificación de las propiedades de esquemas eléctricos ferroviarios que se consideran relevantes, bien por su carácter crítico, o bien porque caracterizaran el futuro sistema eléctrico de la línea ferroviaria a proyectar.

Actualmente es un proceso que se realiza de forma manual, realizando varias iteraciones sobre el diseño que requieren la inversión de muchas horas por parte del proyectista, lo cual es un proceso largo y tedioso. Además, será necesaria la posterior revisión de un experto.

A lo anteriormente comentado hay que añadir que, como toda actividad realizada por el ser humano, lleva inherente una probabilidad de error que, por tratarse de un sistema crítico agrava las consecuencias de dichos errores.
Por otro lado, al ser un proceso largo y pesado, permite muy poco margen de reacción ante la variación de las condiciones de partida o el planteamiento de nuevas hipótesis por parte del cliente.

Todo lo anteriormente comentado, describe una problemática que es necesario abordar. De esta forma, a lo largo del documento se plantea una solución para ello.

3 - OBJETIVOS

De esta manera los objetivos a alcanzar serán los siguientes:

- Realizar una especificación formal de las características a verificar en sistemas de electrificación ferroviarios y, más concretamente, en los esquemas eléctricos que los representan.

- Facilitar dicha verificación a los proyectistas, gracias a la implementación de una herramienta que permita modelizar los esquemas eléctricos y verificar las propiedades correspondientes, para garantizar un diseño con un funcionamiento adecuado y seguro de la electrificación proyectada. En dicha herramienta, se implementará la lógica de negocio de una aplicación que, además, como se verá en la sección de *"Futuros trabajos"*, permitirá en un futuro ampliar la utilidad al proyectista en otros ámbitos como la conexión con programas CAD (computer-aided design) y la interacción con otras técnicas ferroviarias aparte de la de electrificación.

- Aumentar el margen de reacción ante variaciones de condiciones de partida para el diseño o de alguna de las variables de las que depende el estado del sistema.

- Minimizar errores.

4 - SOFTWARE Y MATERIAL DE APOYO UTILIZADO

El software y el material de apoyo utilizado para el desarrollo del proyecto ha sido el siguiente:

- Haskell Platform: que dispone del compilador GHC ("Glasgow Haskell Compiler") integrado, así como un entorno y herramientas para la programación en Haskell.
- Autocad: Software CAD, que es el soporte de los planos utilizados, que aportan gran parte de la información necesaria utilizada posteriormente para implementar la solución software.
- Normativa y otra documentación específica del ámbito ferroviario

5 - SOLUCIÓN PROPUESTA

Debido a la naturaleza del problema a resolver hay que destacar que, el diseño e implementación de la herramienta ha de realizarse teniendo en cuenta varios aspectos (en los que más adelante se realizará más hincapié), de los cuales los principales son:

- Adecuación. Es necesario que la relación entre la especificación de las propiedades a verificar y la implementación sea lo más natural posible. De hecho, en caso de que en un futuro se desee una homologación de la aplicación por parte de un administrador de sistemas ferroviarios será necesario tener una especificación de las propiedades del mismo al tratarse de un sistema crítico. De esta forma, tener una especificación de partida de las propiedades a verificar es un punto a favor si en un futuro se realizan trabajos en este sentido.

- Inmutable. Es aconsejable la utilización de un lenguaje de programación que aporte inmutabilidad en sus entidades, ya que hay que recordar que se trata de sistemas críticos cuyo estado depende de diversas variables y, que, además son escenarios compartidos por diversas técnicas ferroviarias.

- Versatilidad. Es importante que se puedan añadir funcionalidades, modificar criterios etc. sin que esto implique gran cantidad de horas de desarrollo, puesto que las propiedades o reglas a cumplir están supeditadas a normas que pueden variar o requerir nuevas exigencias de cumplimiento. Por tanto, es necesario un sistema cuya mantenibilidad esté garantizada con cohesión y poco acoplamiento en el código.

Por último, hay que resaltar que, en base a lo anteriormente comentado y, teniendo en cuenta el dominio y la naturaleza de los trabajos a realizar, se ha decidido realizar la implementación de nuestra solución en Haskell.

Además, a todas estas razones hay que añadir que comparte el hecho de utilizar un paradigma funcional con el de los lenguajes de programación utilizados en software CAD (como por ejemplo Autolisp y VisualLisp en AutoCad), lo cual también será un punto a favor de cara a poder conectar la solución aportada en el presente Trabajo Fin de Máster con este tipo de programas (como se comentará en la sección de "*Futuros trabajos*" de este documento).

5.1 - METODOLOGÍA UTILIZADA

La metodología empleada para lograr los objetivos planteados será la siguiente:

- Descripción de la propiedad a verificar. Indicando la problemática que se plantea si no se cumple y justificando por tanto la necesidad de su verificación.
- Especificación formal de las propiedades del esquema eléctrico a verificar.
- Verificación de dicha propiedad y comentarios de implementación utilizando para ello ejemplos representativos.

5.2 - CONCEPTOS PREVIOS

5.2.1 - CONCEPTOS DE ESPECIFICACIÓN FORMAL

Para proporcionar una descripción precisa de las propiedades a verificar en el sistema y, poder aportar información adecuada de ellas dentro del dominio del problema, se realizará una especificación formal de dichas propiedades valiéndonos para ello de: lógica formal, notación conjuntista, cuantificadores y conectivas lógicas.

La especificación formal de las propiedades a verificar en los esquemas eléctricos es necesaria principalmente por las siguientes razones:

- El número de propiedades a verificar se va incrementando con el paso del tiempo, puesto que la normativa de seguridad a cumplir va aumentando y es cada vez más estricta. A esto hay que añadir que, en el momento en que se interaccione con otras técnicas en un marco común de trabajo para proyectos ferroviarios, el número de propiedades a verificar se elevará de manera más que considerable. Por ello, se precisa:
    - Tener una relación de todas las propiedades para que, en el momento en que el proyectista lo considere oportuno, pueda consultar si la verificación de una determinada propiedad está incluida en las funcionalidades de la herramienta o no.

    - Las propiedades verificadas por la herramienta han de estar expresadas de forma que no se creen equívocos, ambigüedades o distintas interpretaciones, que sí se dan, por ejemplo, en una especificación con el lenguaje natural escrito.

- El mantenimiento y ampliación de la herramienta debe ser realizado por equipos multidisciplinares. De esta forma, al igual que se ha comentado en el punto anterior es necesario poder comunicarse sin ambigüedades, de tal manera que se tenga claro por parte de todas las disciplinas qué propiedades y reglas de verificación se están incluyendo en el sistema y las características de las mismas.

- La especificación del problema permite describir el comportamiento que se espera del sistema.

- La especificación del problema también sirve para documentar los requisitos de la herramienta a desarrollar.

- Dotar al problema del rigor necesario que exigen los sistemas críticos.

Por último, es conveniente resaltar que la especificación aportada es la del problema a resolver, es decir, la de cada propiedad a verificar en el sistema, sin entrar en detalles de implementación que forman parte de la solución codificada en Haskell. De esta manera, se dota a la especificación del problema de la abstracción necesaria para que aporte los beneficios anteriormente descritos.

Especificación de conceptos generales

A continuación, con ánimo de aportar más claridad y evitar ser repetitivos posteriormente, se describen entidades que se utilizarán en diversas ocasiones en la especificación realizada de cada una de las propiedades a verificar del sistema:

Con *ee* denotamos un esquema eléctrico con la siguiente estructura de tupla:

$$ee = (svpee, elee, gee)$$

$$svpee \equiv sectoresEnVp(ee)$$

$$sectoresEnVp(ee)$$

$$\forall sector \in \{1..nodos(gee)\} \cdot$$

$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot ((sector_i \in resultado) \longrightarrow estaEnViaPrincipal(sector_i)))$$

Con *gee* se denota al grafo que, como veremos en el apartado de "Descripción de la solución", modeliza el esquema eléctrico, cuyos vértices o nodos serán los sectores del esquema eléctrico y cuyas aristas o arcos conforman la configuración eléctrica que define la transición de un sector a otro.

$$gee \equiv conexionesEE(ee)$$

Con *elee* se hace referencia a las características del esquema eléctrico en lo que a electrificación se refiere:

$$elee \equiv electrificacionEE(ee) = (alimee, sf, sc)$$

Con *alimee* se denota al tipo de alimentación que tiene el esquema eléctrico (corriente alterna o corriente continua).

$$alimee \equiv alimentacion(ee) = CC \vee CA$$

Con *sf* se hará referencia al conjunto de sectores fuente existentes en un esquema eléctrico:

$$sf \equiv sectoresFuente(see)$$

$$\forall sector \in \{1..nodos(gee)\} \cdot$$

$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot ((sector_i \in resultado) \longrightarrow esSectorFuente(sector_i)))$$

Con *sc* se hará referencia al conjunto de sectores caídos existentes en un esquema eléctrico:

$$sc \equiv sectoresCaidos(see)$$

$$\forall sector \in \{1..nodos(gee)\} \cdot$$

$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot ((sector_i \in resultado) \longrightarrow esSectorCaido(sector_i)))$$

5.2.2 - CONCEPTOS DE ELECTRIFICACIÓN Y ESQUEMAS ELÉCTRICOS

Es imprescindible incluir algunos conceptos básicos de electrificación, ya que son importantes de cara al desarrollo y la comprensión de las funcionalidades de la herramienta.

Para ello, hagamos un recorrido de los inputs, desde un punto de vista eléctrico, que tendrá el sistema. Dicha información nos la aporta el esquema eléctrico de partida o bien la propia naturaleza del proyecto.

Alimentación

La alimentación del sistema considera las dos posibilidades habituales dentro de la red de transporte ferroviaria española: en corriente continua o corriente alterna.

Los sistemas en corriente continua son los utilizados en líneas convencionales que están alimentadas a 3000 V.

Los sistemas en corriente alterna son los utilizados en líneas de alta velocidad que están alimentadas a 25.000 V (25 kV).

Equipamiento eléctrico

A continuación, veamos los equipos eléctricos que podemos encontrar en los esquemas eléctricos.

- *Aislador de Sección:* Formado por aisladores insertados en la línea de contacto y por dispositivos que aseguran la captación ininterrumpida de la corriente. Se utilizan principalmente para separar unas vías de otras, formando paquetes de vías.

- *Seccionamiento de lámina de aire:* Seccionamiento (solape de dos catenarias) en el que las dos catenarias permanecen independientes eléctricamente.

- *Seccionador:* Se utilizan para separar o unir eléctricamente distintos tramos de catenaria.

La simbología utilizada se muestra en la siguiente figura:

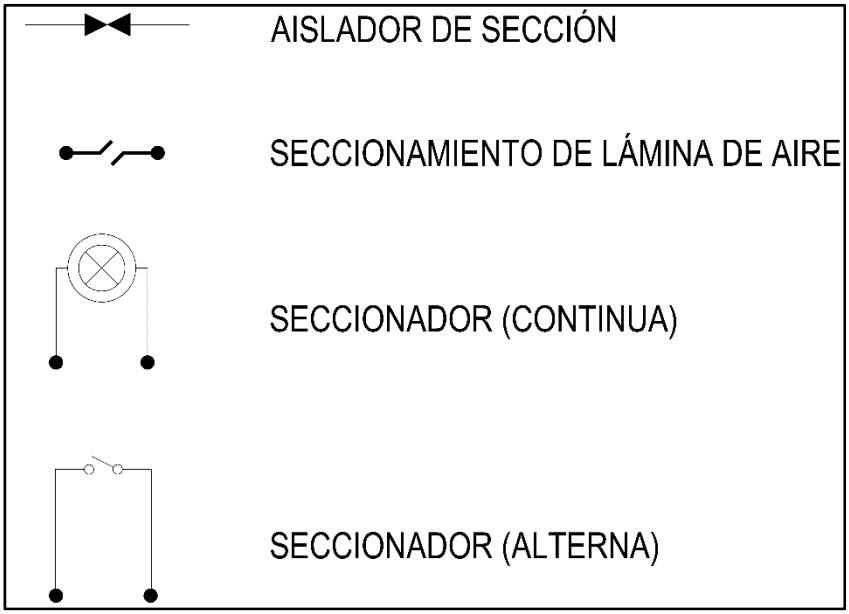

Ilustración 1-Simbología de Equipamiento eléctrico

Zonas Neutras:

Son zonas de catenaria cuya longitud es variable según la explotación y que poseen las siguientes características:

- Se instalan para poder aislar unos sectores eléctricos de otros (porque tengan tensiones diferentes, estén alimentados por distintas subestaciones, etc.).
- Son zonas sin tensión.
- Están conectadas a tierra.

5.3 - DESCRIPCIÓN DE LA SOLUCIÓN

5.3.1 MODELIZACIÓN DEL ESQUEMA ELÉCTRICO

La solución planteada se basa en la modelización del esquema eléctrico a través de un grafo. De esta forma utilizamos una estructura matemática, con definiciones, conceptos, teoremas y algoritmos ampliamente estudiados y conocidos, lo cual ayuda a la hora de establecer una especificación formal de las propiedades a verificar con el rigor exigido por un sistema crítico, como es el que nos compete en nuestro estudio.

A continuación, se muestra un ejemplo de esquema eléctrico:

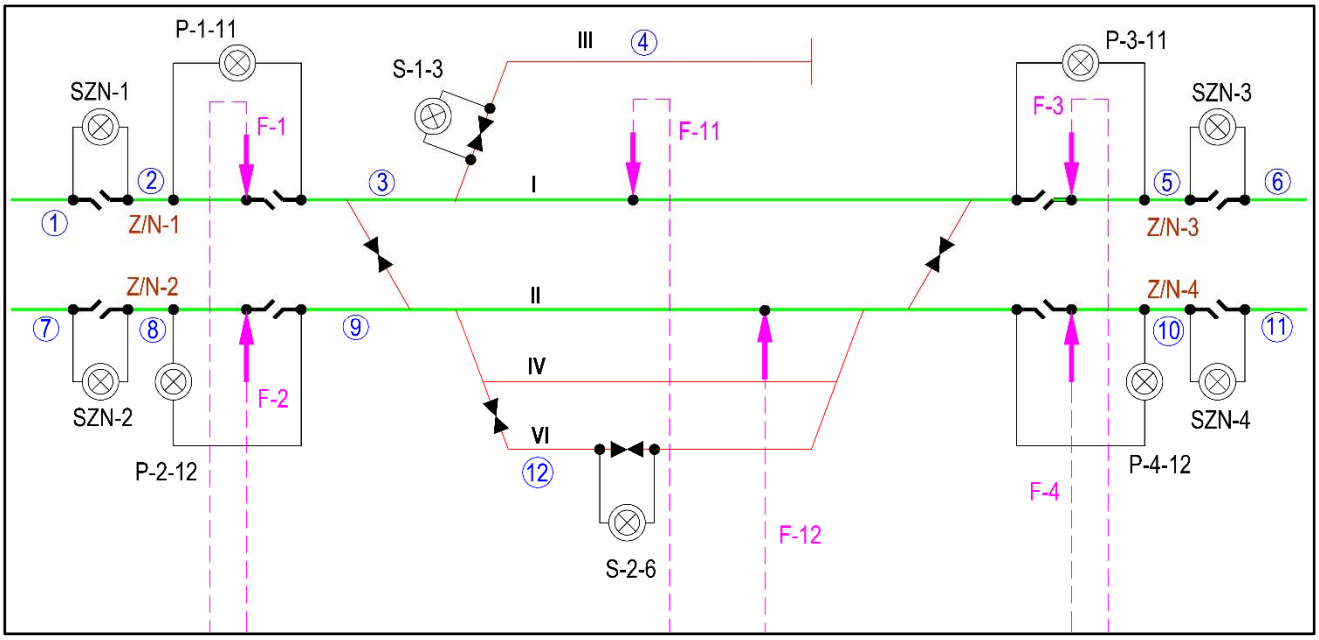

Ilustración 2-Esquema eléctrico para ejemplo de modelización

En el esquema eléctrico se han numerado en azul los sectores eléctricos y se ha omitido voluntariamente cierta información (PKs, tipo de electrificación, etc.) que se mostrará posteriormente y se explicará cuando se aborden las verificaciones de propiedades del sistema propiamente dichas. De esta manera,

el esquema eléctrico queda más sencillo y “limpio” lo que ayuda a cumplir con el objetivo de este apartado que es el de mostrar cómo se modelizará el problema.

Vemos cómo quedaría modelizado el esquema eléctrico anterior mediante un grafo:

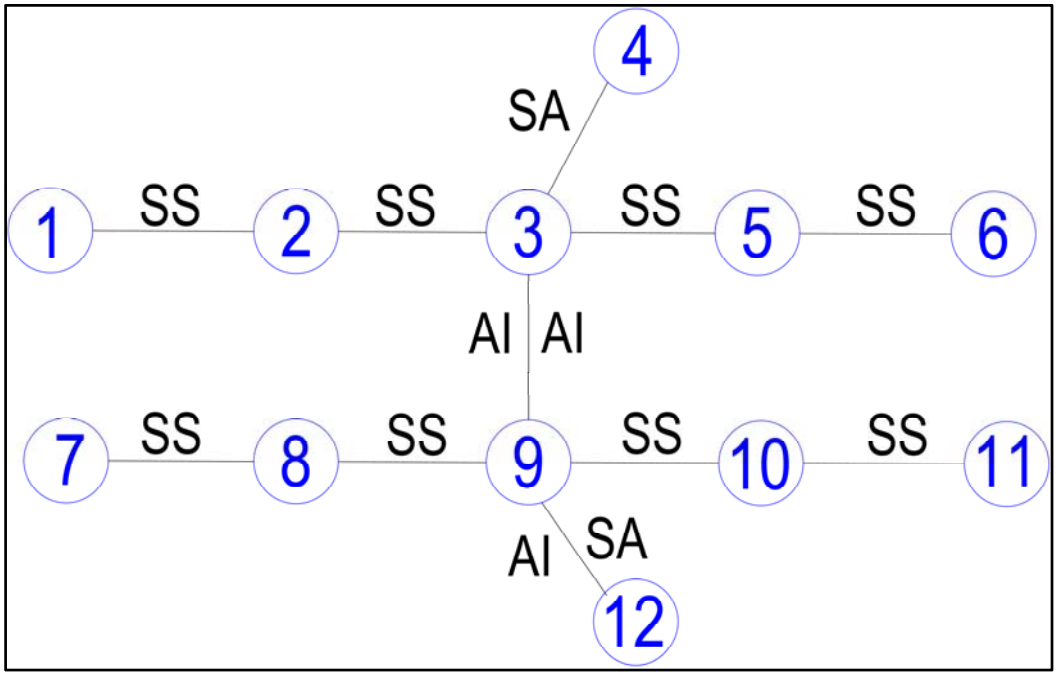

Ilustración 3-Grafo para ejemplo de modelización

Cómo se puede observar:

- Nodos: sectores eléctricos.
- Aristas: paso de un sector eléctrico a otro a través del equipo eléctrico que hay entre ambos.

A continuación, se realizan otras consideraciones al respecto:

- Con dos círculos concéntricos se representan los nodos fuente, es decir los nodos a los que les llega tensión directamente desde la subestación (este hecho se muestra en magenta con una flecha representativa).

- En determinados casos como en los escapes entre vías principales, se puede producir que dos sectores eléctricos estén comunicados por más de un equipo. Este aspecto, se trata en la implementación gestionando adecuadamente la estructura de datos necesaria para, de esta manera, poder modelizar el sistema mediante un grafo, ya que en caso contrario habría que utilizar multigrafos.

5.3.2 PROPIEDADES A VERIFICAR

A continuación, se indican todos los requisitos de seguridad exigidos que debemos verificar sobre un diseño de esquema eléctrico desde un punto de vista funcional.

De esta forma, es necesario presentar los esquemas eléctricos sobre los que se irán ilustrando las distintas propiedades a verificar en los diseños.

Así, en la siguiente figura se muestra un esquema eléctrico en corriente continua sobre el que se irán realizando progresivamente las distintas verificaciones necesarias para poder asegurar que el diseño es correcto. Dicho, esquema eléctrico será referido a largo del documento como *ee1*.

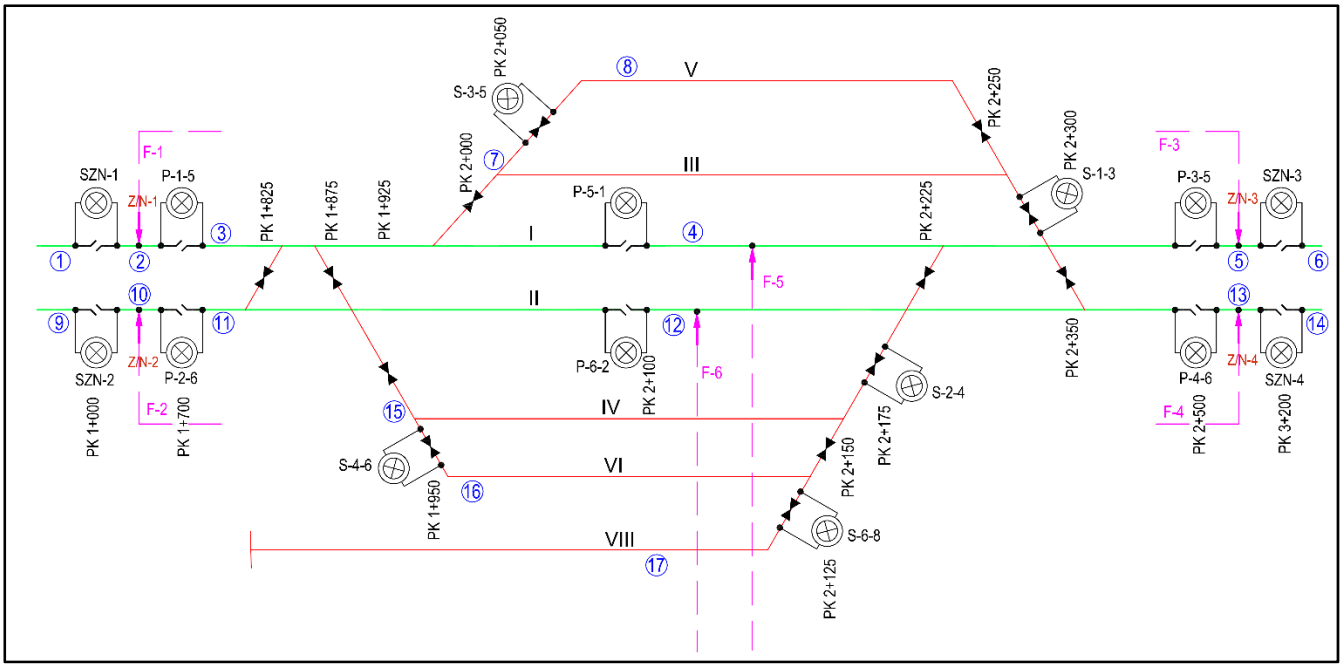

Ilustración 4-Esquema eléctrico ee1

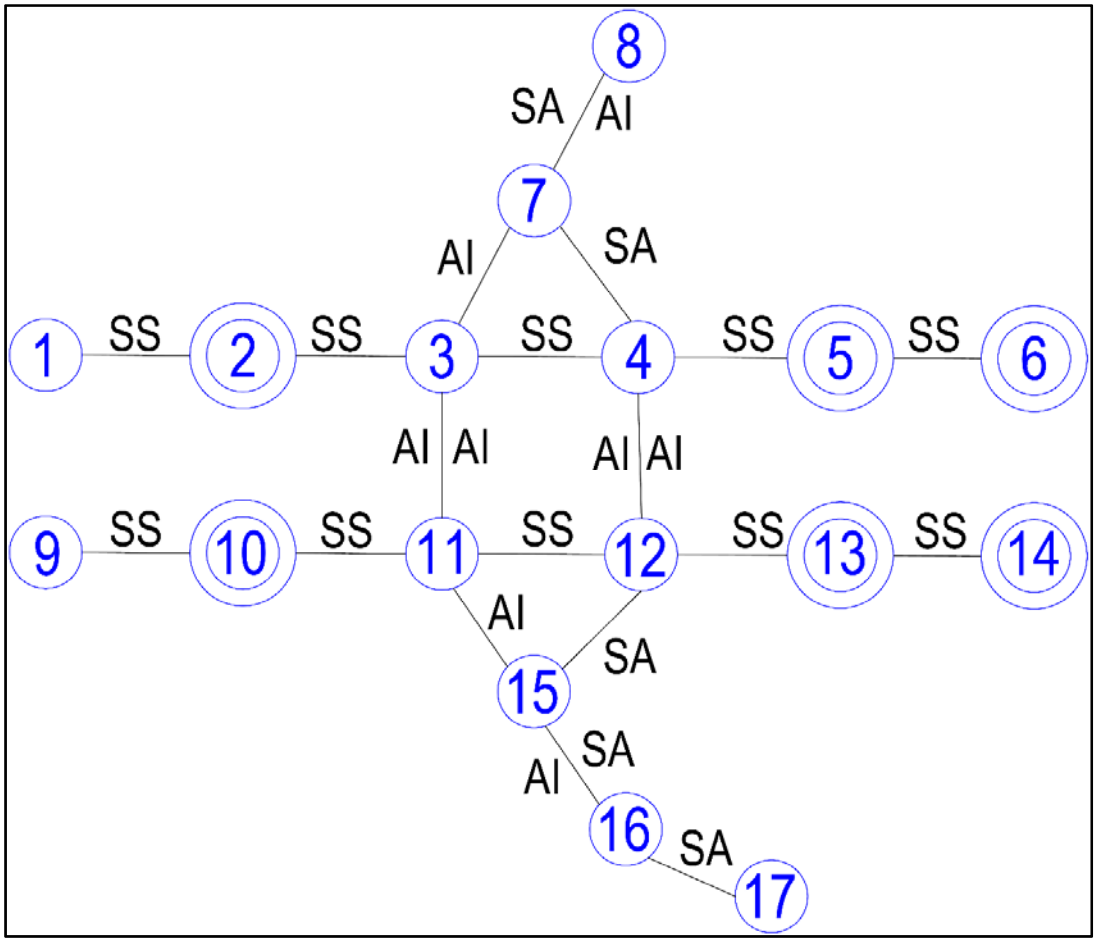

Ilustración 5-Grafo para modelización de esquema eléctrico ee1

```
conexionesEE1 = creaGrafo (1,17) [
                                  ([(SS,True,(1.000,(1,1)))],1,2),
                                  ([(SS,False,(1.700,(1,1)))],2,3),
                                  ([(SS,True,(2.100,(1,1)))],3,4), ([(AI,True,(2.000,(1,3)))],3,7), ([(AI,True,(1.825,(1,2))),
                                       (AI,True,(1.875,(1,2)))],3,11),
                                  ([(SS,False,(2.500,(1,1)))],4,5), ([(SA,True,(2.300,(1,3)))],4,7),([(AI,True,(2.225,(1,2))),
                                       (AI,True,(2.350,(1,2)))],4,12),
                                  ([(SS,True,(3.200,(1,1)))],5,6),
                                  ([(SA,True,(2.050,(3,5))), (AI,True,(2.250,(3,5)))],7,8),
                                  ([(SS,True,(1.000,(2,2)))],9,10),
                                  ([(SS,False,(1.700,(2,2)))],10,11),
                                  ([(SS,True,(2.100,(2,2)))],11,12),([(AI,True,(1.925,(2,4)))],11,15),
                                  ([(SS,False,(2.500,(2,2)))],12,13),([(SA,True,(2.175,(2,4)))],12,15),
                                  ([(SS,True,(3.200,(2,2)))],13,14),
                                  ([(SA,True,(1.950,(4,6))), (AI,True,(2.150,(4,6)))],15,16),
                                  ([(SA,True,(2.125,(6,8)))],16,17)
                                 ]
```

Ilustración 6-Conexiones de esquema eléctrico ee1

```
electrificacionEE1 = creaElectrificacion ([(1, ZS, ((0.00,(1,1)), (1.000,(1,1)))),
                                           (2, ZN, ((1.000,(1,1)), (1.700,(1,1)))),
                                           (3, ZS, ((1.700,(1,1)), (2.100,(1,1)))),
                                           (4, ZS, ((2.100,(1,1)), (2.500,(1,1)))),
                                           (5, ZN, ((2.500,(1,1)), (3.200,(1,1)))),
                                           (6, ZS, ((3.200,(1,1)), (3.299,(1,1)))),
                                           (7, ZS, ((2.000,(1,3)), (2.300,(1,3)))),
                                           (8, ZS, ((2.050,(3,5)), (2.250,(3,5)))),
                                           (9, ZS, ((0.00,(2,2)), (1.000,(2,2)))),
                                           (10, ZN, ((1.000,(2,2)), (1.700,(2,2)))),
                                           (11, ZS, ((1.700,(2,2)), (2.100,(2,2)))),
                                           (12, ZS, ((2.100,(2,2)), (2.500,(2,2)))),
                                           (13, ZN, ((2.500,(2,2)), (3.200,(2,2)))),
                                           (14, ZS, ((3.200,(2,2)), (3.299,(2,2)))),
                                           (15, ZS, ((1.925,(2,4)), (2.175,(2,4)))),
                                           (16, ZS, ((1.950,(4,6)), (2.150,(4,6)))),
                                           (17, ZS, ((1.825,(8,8)), (2.125,(6,8))))],
                                          CC, [(2,N), (4,N), (5,N),(10,N), (12,N), (13,N)], [])
```

Ilustración 7-Electrificación de esquema eléctrico ee1

```
sectoresVpEE1 = [1,2,3,4,5,6,9,10,11,12,13,14]


ee1 = crearEsquemaElectrico sectoresVpEE1 electrificacionEE1 conexionesEE1
```

Ilustración 8-Sectores en Vía principal y creación de esquema eléctrico ee1

En las siguientes figuras se muestra el esquema eléctrico de corriente alterna que se utilizará en las verificaciones, dicho esquema será referido en el resto del documento como *ee2*.

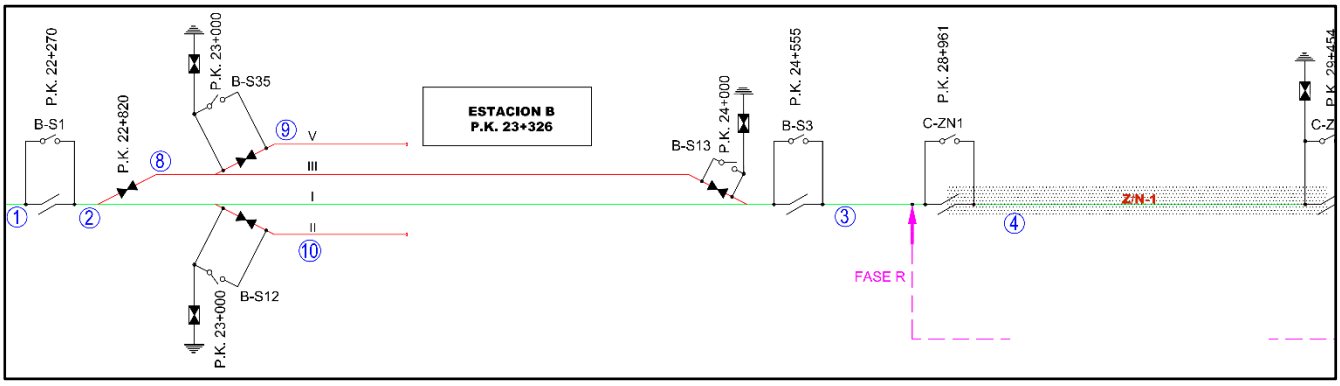

Ilustración 9-Esquema eléctrico ee2 (parte 1).

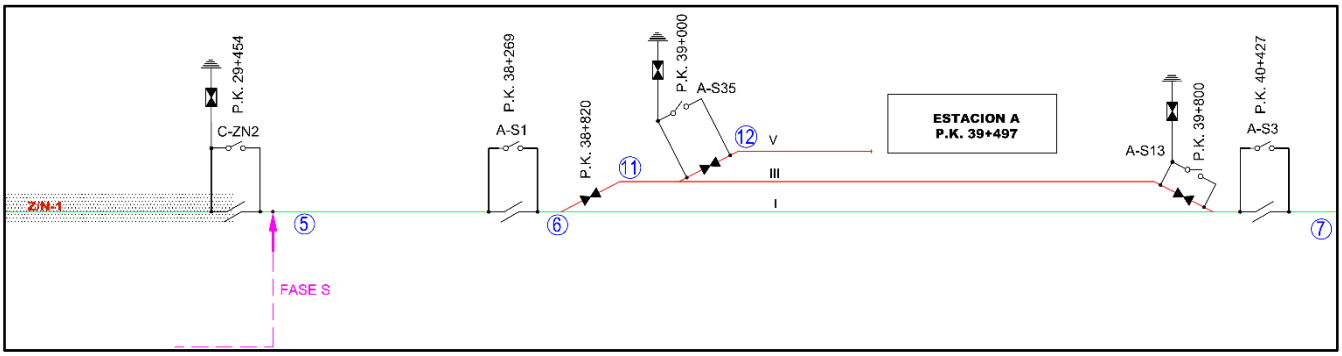

Ilustración 10-Esquema eléctrico ee2 (parte 2).

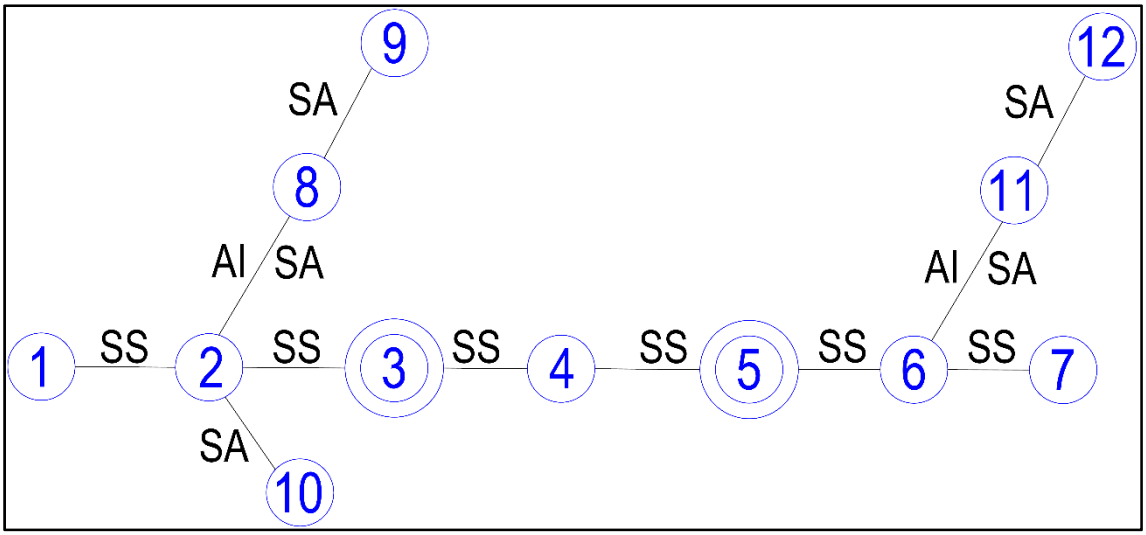

Ilustración 11-Grafo para modelización de esquema eléctrico ee2

```
conexionesEE2 = creaGrafo (1,12) [
                                  ([(SS,True,(22.270,(1,1)))],1,2),
                                  ([(SS,True,(24.555,(1,1)))],2,3),
                                  ([(SS,False,(28.961,(1,1)))],3,4),
                                  ([(SS,False,(29.454,(1,1)))],4,5),
                                  ([(SS,True,(38.269,(1,1)))],5,6),
                                  ([(SS,True,(40.427,(1,1)))],6,7),
                                  ([(AI,True,(22.820,(1,3))),(SA,True,(24.000,(1,3)))],2,8),
                                  ([(SA,True,(23.000,(3,5)))],8,9),
                                  ([(SA,True,(23.000,(1,2)))],2,10),
                                  ([(AI,True,(38.820,(1,3))), (SA,True,(39.800,(1,3)))],6,11),
                                  ([(SA,True,(39.000,(3,5)))],11,12)
                                 ]
```

Ilustración 12-Conexiones de esquema eléctrico ee2

```
electrificacionEE2 = creaElectrificacion ([(1, ZS, ((22.000,(1,1)), (22.270,(1,1)))),
                                           (2, ZS, ((22.270,(1,1)), (24.555,(1,1)))),
                                           (3, ZS, ((24.555,(1,1)), (28.961,(1,1)))),
                                           (4, ZN, ((28.961,(1,1)), (29.454,(1,1)))),
                                           (5, ZS, ((29.454,(1,1)), (38.269,(1,1)))),
                                           (6, ZS, ((38.269,(1,1)), (40.427,(1,1)))),
                                           (7, ZS, ((40.427,(1,1)), (40.499,(1,1)))),
                                           (8, ZS, ((22.820,(1,3)), (24.000,(1,3)))),
                                           (9, ZS, ((23.000,(3,5)), (23.230,(5,5)))),
                                           (10, ZS, ((23.000,(1,2)), (23.230,(2,2)))),
                                           (11, ZS, ((38.820,(1,3)), (39.800,(1,3)))),
                                           (12, ZS, ((39.000,(3,5)), (2.500,(39,510))))],
                                            CA, [(3,R), (5,S)], [])

sectoresVpEE2 = [1,2,3,4,5,6,7]


ee2 = crearEsquemaElectrico sectoresVpEE2 electrificacionEE2 conexionesEE2
```

Ilustración 13-Electrificación, sectores en vía principal y creación de esquema eléctrico ee2

5.3.2.1 - SECTORES ELÉCTRICOS CORRECTAMENTE ALIMENTADOS

- Descripción de característica a verificar y problemática.

Una de las funcionalidades básicas a conseguir en el diseño de esquemas eléctricos es que todos los sectores eléctricos estén correctamente alimentados, es decir, que a todos les llegue tensión. Así, a un sector le podrá llegar tensión bien directamente de la subestación o bien procedente de otro sector eléctrico, al transmitirse dicha tensión gracias a los seccionadores correspondientes.

De esta manera, es un error grave que podría ocasionar problemas en caso de no ser localizado a tiempo, el hecho de que un sector no pueda llegar a ser alimentado debido a un diseño incorrecto del esquema eléctrico realizado por el proyectista.

- Especificación de propiedad

Un sector eléctrico estará alimentado si existe un camino a un sector fuente (sector al que le llega corriente directamente procedente de subestación) o si él es un sector fuente por sí mismo. De esta forma, podemos especificar dicha propiedad de la siguiente manera:

$$todosSectoresNoCaidosBienAlimentadosEE(ee)$$

$$resultado = \forall sector \in \{1..nodos(gee)\} \cdot$$

$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot esSectorAlimentadoSinEstado(sector_i, ee))$$

$$esSectorAlimentadoSinEstado(s, ee)$$

$$resultado = (\neg esSectorCaido(s, sc) \wedge EsSectorFuente(s, sf)) \vee hayFuente(setoresAccesiblesEE(s, gee, RSEC), sf)$$

Con sectoresAccesiblesEE se obtienen los nodos por los que pasan los recorridos (a través de seccionadores, indicado como RESEC) que parten del sector en cuestión. Al ser el recorrido una operación sobre la estructura matemática grafo cuyas propiedades y operaciones ya han sido formalizadas y demostrados no es necesario su especificación, ya que responde a la indicada en cualquier libro y no es una particularidad de nuestro problema que pueda crear ambigüedad o problemas de falta de información.

$$hayFuente(ls, sf)$$

$$resultado = \exists i \in \{1..Longitud(ls)\} \cdot esSectorFuente(ls_i, sf)$$

$$esSectorCaido(s, sc)$$

$$resultado = \exists i \in \{1..Longitud(sc)\} \cdot sc_i = s$$

$$esSectorFuente(s, sf)$$

$$resultado = \exists i \in \{1..Longitud(sf)\} \cdot sf_i = s$$

- Verificación de propiedad en esquema eléctrico

En la solución implementada se contemplan funcionalidades para poder acometer los trabajos de verificación asociados a comprobar que todos los sectores eléctricos están correctamente alimentados.

A continuación, se realizan una serie de consideraciones relevantes:

- La implementación permite al proyectista tanto verificar que un sector eléctrico está correctamente alimentado de manera individual como realizar la verificación sobre todos los sectores del esquema eléctrico a la vez.

- Se permite realizar la verificación tanto teniendo en cuenta el estado del esquema eléctrico en un momento dado (marcado principalmente

por la posición en que se encuentran los seccionadores) como sin tenerlo en cuenta, es decir, sobre el diseño de manera general. Normalmente se utiliza esta segunda opción, pues generalmente la manera de acometer un nuevo diseño o una variación en el mismo es considerando que bajo alguna configuración posible todos los sectores eléctricos estarán bien alimentados. No obstante, puede ser que, en algún momento, bien a petición del cliente o bien por alguna particularidad del diseño o del esquema eléctrico en cuestión, sea útil poder verificar si un sector eléctrico está correctamente alimentado bajo un estado de funcionamiento concreto del esquema eléctrico.

- Se puede consultar la implementación con detalle en el anexo de listado de código (esSectorAlimentadoConEstado, esSectorAlimentadoSinEstado, todosSectoresNoCaidosBienAlimentadosEE).

Veamos sobre el ejemplo del esquema eléctrico de continua *ee1*, como se realizaría la verificación. Sí queremos ver si algún sector eléctrico en particular con el diseño actual está alimentado:

```
*Esquema1> esSectorAlimentadoSinEstado 2 ee1
True
*Esquema1> esSectorAlimentadoSinEstado 10 ee1
True
*Esquema1> esSectorAlimentadoSinEstado 4 ee1
True
```

Ilustración 14-Verificación sectores 2,10 y 4 bien alimentados en ee1

Si se quiere comprobar de una vez que todos los sectores eléctricos del esquema eléctrico están correctamente alimentados:

```
*Esquema1> todosSectoresNoCaidosBienAlimentadosEE ee1
True
```

Ilustración 15-Verificación todos sectores bien alimentados en ee1

De la misma manera se podría proceder sobre un esquema eléctrico de corriente alterna:

```
*Esquema2> todosSectoresNoCaidosBienAlimentadosEE ee2
True
```

Ilustración 16-Verificación todos sectores bien alimentados en ee2

Supongamos que el proyectista en el diseño del esquema eléctrico de alterna *ee2* no ha considerado la inclusión del seccionador B-S13 en la estación B. Es decir, esa zona del esquema eléctrico pasará del siguiente diseño:

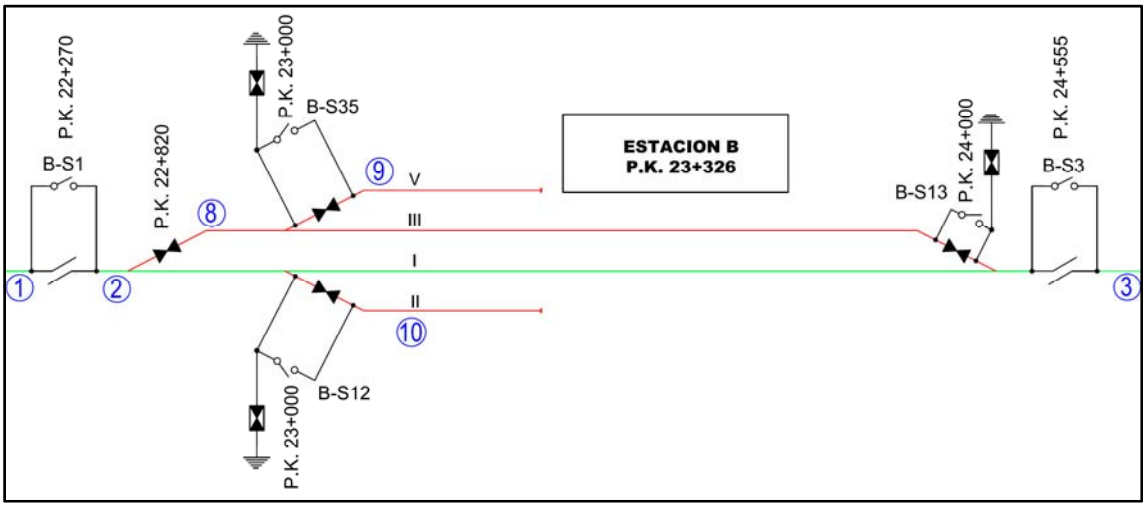

Ilustración 17-Esquema eléctrico de partida para ee3

A este otro:

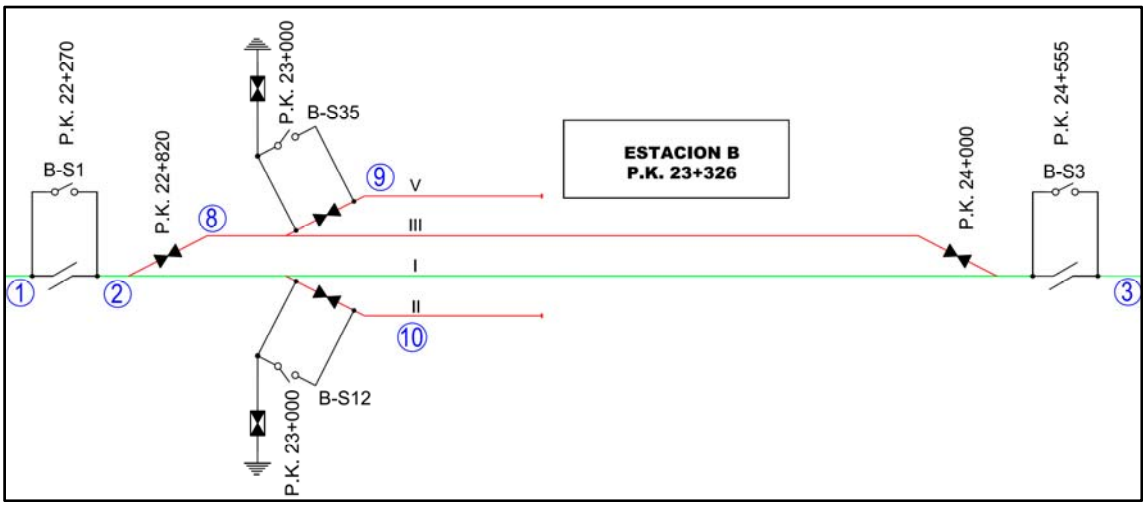

Ilustración 18-Esquema eléctrico ee3

Al esquema eléctrico resultante le llamaremos *ee3*.

Cuando procedamos a la verificación correspondiente a que el sector 8 esté correctamente alimentado obtenemos lo siguiente:

```
*Esquema3> esSectorAlimentadoSinEstado 8 ee3
False
```

Ilustración 19-Verificación sector 8 bien alimentado en ee3

La verificación no es satisfactoria y, por tanto, no hay posibilidad con el diseño actual de que el sector eléctrico 8 quede alimentado.

De la misma forma, si procedemos a la verificación de la propiedad en sector 9:

```
*Esquema3> esSectorAlimentadoSinEstado 9 ee3
False
```

Ilustración 20-Verificación sector 9 bien alimentado en ee3

Por tanto, en este caso el resultado de la comprobación también es falso, lo que nos indica que el sector eléctrico 9 tampoco tiene posibilidad de ser alimentado en el diseño actual.

Gracias al resultado de las verificaciones anteriores el proyectista puede identificar que el diseño del esquema eléctrico *ee3* es erróneo. De esta manera, podría analizar el error y darse cuenta que, tal como está diseñado *ee3*, se dejaría sin posibilidad de alimentación a la vía III y, además, tampoco permitiría alimentar a la vía V, ya que, aunque la vía V cuenta con seccionador y está correctamente aislada, no puede ser alimentada por alimentarse solamente de la vía III.

Lógicamente, si anteriormente a la verificación individual de cada sector eléctrico se hubiera comprobado la propiedad en todo el esquema eléctrico a la vez, el resultado de la evaluación de dicha propiedad hubiese sido también falso:

```
*Esquema3> todosSectoresNoCaidosBienAlimentadosEE ee3
False
```

Ilustración 21-Verificación todos sectores bien alimentados en ee3

Supongamos ahora que el técnico proyectista ha hecho un diseño como el del esquema eléctrico *ee1* pero considerando que el seccionador S-3-5 no debe existir. Es decir, pasamos del siguiente diseño:

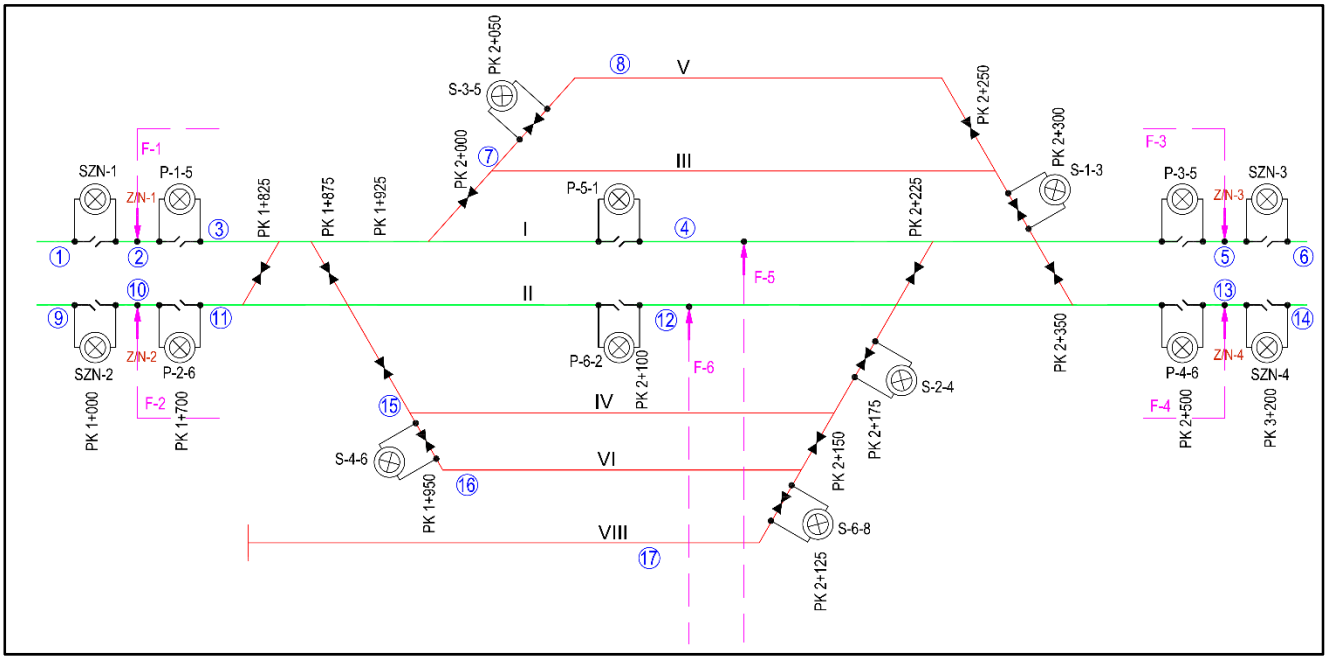

Ilustración 22-Esquema eléctrico de partida para ee4

A este otro:

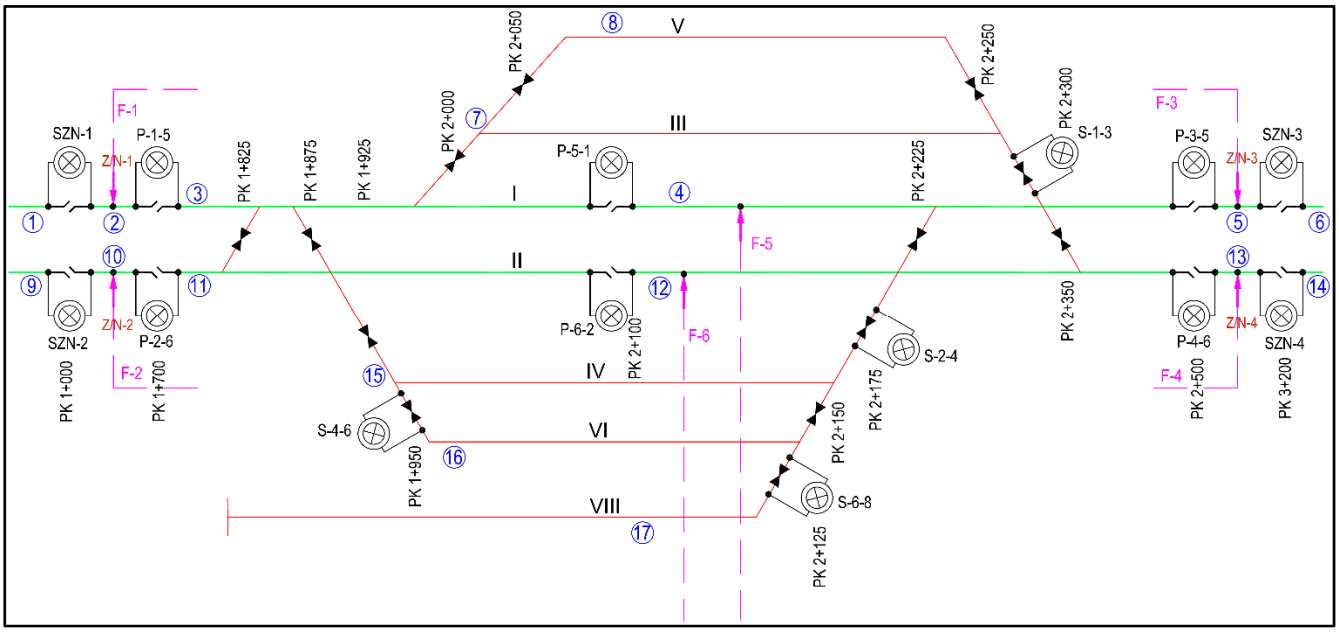

Ilustración 23-Esquema eléctrico ee4

A este esquema resultante le llamaremos *ee4*.

Al realizar la verificación sobre todo el esquema eléctrico el proyectista obtendría el siguiente resultado:

```
*Esquema4> todosSectoresNoCaidosBienAlimentadosEE ee4
False
```

Ilustración 24-Verificación todos sectores bien alimentados en ee4

Y concretamente al realizar la verificación en el sector 8:

```
*Esquema4> esSectorAlimentadoSinEstado 8 ee4
False
```

Ilustración 25-Verificación sector 8 bien alimentado en ee4.

De esta forma, gracias a los resultados obtenidos de las verificaciones anteriores el técnico sabrá que su diseño no es correcto. De hecho, analizándolo, se dará cuenta que tal como se encuentra el diseño en *ee4* dejaría sin posibilidad de alimentación a la vía V.

5.3.2.2 - VERIFICACIÓN EN CORRIENTE ALTERNA CONSISTENTE EN QUE UN MISMO SECTOR NO SE ENCUENTRE SIMULTÁNEAMENTE ALIMENTADO POR DOS FASES DIFERENTES

- Descripción de característica a verificar y problemática.

En los esquemas eléctricos de corriente alterna es necesario verificar que un mismo sector eléctrico no está alimentado por dos fases eléctricas diferentes simultáneamente. Dos fases no pueden alimentar a la vez a un mismo sector eléctrico, debido a que esto produciría un cortocircuito entre dos fases. La transición del tren entre tramos de catenaria alimentados a distinta fase eléctrica

debe ser convenientemente proyectada o, de lo contrario, podría producirse una conexión eléctrica entre dos fases distintas de la línea de transporte. Esta zona de transición es denominada zona neutra de separación de fases eléctricas.

En caso de que no se verifique de forma adecuada esta propiedad dentro del esquema eléctrico se generaría la siguiente problemática: la conexión de dos fases diferentes provocaría un cortocircuito que puede causar importantes daños en las instalaciones eléctricas e incluso incendios, algo que lógicamente debe evitarse con el fin de proteger a las personas y a los bienes materiales.

Para verificar esta propiedad será necesario considerar el estado del esquema eléctrico y, concretamente, la posición de los seccionadores, ya que es importante volver a destacar que un sector no podrá recibir tensión con dos fases diferentes simultáneamente.

- Especificación de propiedad

La propiedad a verificar será, por tanto, que todas las fuentes de un sector han de tener la misma fase asociada.

$$todosSectoresConSoloUnTipoDeFaseEE(ee)$$

$PRE: alimee = CA$

$resultado = \ \forall sector \in \{1..nodos(gee)\} \cdot$

$(\forall i \in \{1..longitud(nodos(gee))\} \cdot esSectorConUnicaFuente(sector_i, ee))$

$$esSectorConUnicaFuente(s, ee)$$

$PRE: EstaAlimentado(s)$

$resultado = mismaFase\ (fuentesDeSectorConEstado(s, ee))$

$$fuentesDeSectorConEstado(s, ee)$$

$$\forall i \in \{1..Longitud(resultado)\} \cdot esSectorFuente(resultado_i, sf)$$

$$\bigwedge resultado_i \in sectoresAccessiblesEE(s, gee, RSEC)$$

$$mismaFase(ls)$$

$$resultado = \ \forall i, j \in \{1..Longitud(ls)\} \ \cdot Fase(ls_i) = Fase(ls_j)$$

- Verificación de propiedad en esquema eléctrico

En la implementación se consideran todos los posibles sectores eléctricamente comunicados con el que se desea verificar y, a partir de ahí, obtenemos las fuentes que alimentan directa o indirectamente al sector a estudio. Posteriormente, se verificará que la fase que llega al sector es siempre la misma.

La forma de proceder por parte del proyectista para realizar estas verificaciones es comprobar que en cada sector alimentado no se produzca la circunstancia de que concurran dos fases diferentes.

De esta forma, se ofrece al usuario la posibilidad de realizar la verificación en varios sectores a la vez. Veamos un ejemplo sobre el esquema eléctrico *ee2*:

```
*Esquema2> todosSectoresConSoloUnTipoDeFase [1,2,3,5,6,7,8,9,12] ee2
True
```

Ilustración 26-Verificación sectores 1,2,3,5,6,7,8,9,12 no tienen 2 fases diferentes simultáneamente en ee2.

Nótese que no se comprueba el sector 4, puesto que no está alimentado teniendo en cuenta el estado y la configuración del sistema *ee2* (es una zona neutra).

Con la siguiente función que se mostrará a continuación se ha realizado la implementación para que directamente se pueda realizar la comprobación de todos los sectores del esquema eléctrico, pero hay que tener en cuenta que en los sectores que no estén alimentados se considerará que no hay repetición de fases porque, de hecho, no hay fase al no llegarles tensión y es algo que el técnico debe tener en cuenta desde un punto de vista conceptual del diseño y su verificación.

```
*Esquema2> todosSectoresConSoloUnTipoDeFaseEE ee2
True
```

Ilustración 27-Verificación todos sectores no tienen 2 fases diferentes simultáneamente en ee2.

Se puede consultar la implementación con detalle en el anexo de listado de código (todosSectoresConSoloUnTipoDeFase, todosSectoresConSoloUnTipoDeFaseEE).

Hay que tener en cuenta que, en la práctica, la verificación de esta propiedad de forma manual por parte del proyectista es más compleja de lo que parece ya que, por ejemplo, la verificación de una línea entera supondrá mucho más trabajo que la de dos estaciones como se muestran en el esquema *ee2* y, ahí radica la potencia y la utilidad de la herramienta desarrollada en lo que a la verificación de esta propiedad se refiere.

5.3.2.3 - COMPROBAR QUE EN CORRIENTE ALTERNA SECTORES CONTIGUOS A AMBOS LADOS DE UNA ZONA NEUTRA TIENEN DISTINTA FASE EN CONDICIONES NORMALES DE EXPLOTACIÓN.

- Descripción de característica a verificar y problemática.

- Esta propiedad es otra de las que se deben comprobar para verificar un diseño de esquema eléctrico que funciona en corriente alterna. El objetivo de dicha comprobación es asegurar que en condiciones normales de explotación a ambos extremos de la zona neutra (la cual estará puesta a tierra y por tanto no tendrá tensión ni fase) la tensión llega con fases diferentes.

- El problema que surgiría si, en condiciones normales de explotación tuvieran la misma fase, sería que no se está explotando la línea ferroviaria en condiciones óptimas desde el punto de vista eléctrico y, que debido a los grandes consumos sobre el sistema eléctrico, se está produciendo un desequilibrio de fases en el sistema de distribución eléctrico de las LAAT (líneas aéreas de alta tensión) que alimentan la subestación.

- Es importante destacar que esta verificación no debe realizarse en condiciones degradadas, puesto que justo cuando falla una fase es cuando se alimenta todo de la otra y, entonces, lógicamente sí que se debe cumplir la condición de que la fase a la derecha y a la izquierda de la zona neutra (y en este caso en la propia zona neutra) es la misma.

- Especificación de propiedad

La propiedad a verificar será que los sectores a ambos extremos de la zona neutra de corriente han de estar a distinta fase. Así:

$$compruebaFasesSectoresExtremoZn(s, ee)$$

$PRE: EsZn(s) \wedge alimee = CA$

$resultado = \forall s1, s2 \in nodos(gee) \cdot (((pkFinal(s1) < pkInicial(s)) \wedge \text{adyacentes(s1,s)}) \wedge ((pkInicial(s2) > pkFinal(s)) \wedge \text{adyacentes(s2,s)})) \longrightarrow Fase(s1) \neq Fase(s2)$

- Verificación de propiedad en esquema eléctrico

El proyectista podrá verificar esta característica de su diseño de forma simple, como podremos ver a continuación.

En el esquema de corriente alterna *ee2* vemos que la zona neutra corresponde al sector eléctrico 4 y, como vemos, los sectores que hay a cada extremo son el sector 3 y el 5. Puesto que a ambos les llega distinta fase el resultado de nuestra verificación ha de ser satisfactorio:

```
*Esquema2> compruebaFasesSectoresExtremosZn 4 ee2
True
```

Ilustración 28-Verificación sectores en extremos de ZN 4 tienen distinta fase en ee2.

Suponiendo un esquema eléctrico *ee2'* que fuera como *ee2* pero variando que tanto el sector 3 como el 5 recibieran tensión con fase R (las fuentes serían [(3,R), (5,R)]) y, se realizase la verificación en el esquema *ee2'* , en este caso el resultado de la comprobación debe ser que no se cumple la propiedad:

```
*Esquema2> compruebaFasesSectoresExtremosZn 4 ee2'
False
```

Ilustración 29-Verificación sectores en extremos de ZN 4 tienen distinta fase en ee2 modificando fases de partida.

Se puede consultar la implementación con detalle en el anexo de listado de código (compruebaFasesSectoresExtremosZn).

5.3.2.4 - VERIFICAR SI LOS TRENES PUEDEN ATRAVESAR ESTACIÓN ANTE INCIDENCIA EN VÍA SECUNDARIA.

- Descripción de característica a verificar y problemática.

Uno de los problemas a evitar a la hora de realizar el diseño de un esquema eléctrico es que, ante una incidencia producida en vía secundaria, se impida que los trenes puedan atravesar una estación.

- Especificación de propiedad

$$puedeLLegarTren(s1, s2, ee)$$
$$resultado = s2 \in sectoresLlegaTrenCirculando\ (s1, ee)$$

$$sectoresLlegaTrenCirculando\ (s1, ee)$$
$$\forall i \in \{1..Longitud(resultado) - 1\} \cdot pasaTrenSectoresContiguos(resultado_i, resultado_{i+1})$$

$$pasaTrenSectoresContiguos\ (s1, s2)$$
$$resultado =$$
$$\begin{cases} ((equipo(s1, s2) = AI) \vee estadoConfiguracion\big(equipo(s1, s2)\big) = FALSO) \longrightarrow \\ \quad estaAlimentadoConEstado(s1) \wedge estaAlimentadoConEstado(s2) \\ \\ e.o.c \longrightarrow estaAlimentadConEstado(s1) \vee estaAlimentadoConEstado(s2) \end{cases}$$

- Verificación de propiedad en esquema eléctrico

La manera de proceder en estos casos es ir comprobando que ante posibles incidencias en las vías secundarias que indica el cliente (en caso de que lo haga) o a criterio del técnico (normalmente se considera que caen sectores debido a incidencias en vías secundarias) los trenes podrán atravesar la estación.

En este caso, es una verificación que tiene sentido realizar teniendo en cuenta el estado de los seccionadores del esquema eléctrico en un momento determinado, de hecho, como veremos a continuación, dependiendo entre otras cosas de la posición del seccionador un tren podrá pasar o no de un sector a otro.

Para ello, en la implementación se han tenido en cuenta las posibilidades que tiene un tren para atravesar los distintos sectores, dependiendo del equipamiento eléctrico que los delimite. Así es importante destacar:

- Entre dos sectores eléctricos separados por un aislador de sección, un tren sólo podrá pasar de un sector a otro si ambos sectores están alimentados.
- Entre dos sectores eléctricos separados por alguna configuración con seccionador (seccionador + aislador de sección o seccionador + seccionamiento de lámina de aire):
    - Si el seccionador está abierto el tren sólo puede pasar de un lado a otro si ambos sectores están alimentados y, por tanto, con corriente.
    - Si el seccionador está cerrado será necesario como mínimo que uno de los dos sectores esté alimentado para que el tren pueda pasar de un sector a otro.

Se puede consultar la implementación con detalle en el anexo de listado de código (puedeLlegarTren).

Veamos algunos ejemplos de utilización de la solución desarrollada, tanto en el esquema de corriente continua como en el de alterna. Para ello, el usuario provoca la caída del sector que desea que deje de estar alimentado y, a continuación, prueba si se puede llegar de algún sector de un extremo de la estación a otro sector del extremo opuesto.

Por ejemplo, sobre el esquema eléctrico *ee1*:

Supongamos que hay una incidencia en vía III, por tanto cae el sector 7. De esta manera *ee1* pasa a un estado *ee1'*.

```
ee1' = aislarSectorEE 7 ee1
```

Ilustración 30-Incidencia en vía secundaria III en ee1 (caída de sector 7).

Así, en el estado *ee1'* se ha realizado la operativa necesaria tras la caída de un sector (se abren todos sus seccionadores para aislar eléctricamente el sector con el fallo y se informa que ese sector es sector caído). En este caso, se abren los seccionadores S-1-3 y S-3-5 y se añade el sector 7 al conjunto de sectores caídos.

A continuación, habría que comprobar que el tren pueda llegar de alguno de los sectores de un extremo (1 y 9) a los del otro (6 y 14). De esta manera, podemos observar que con el funcionamiento del sistema en degradado tras la caída del sector 7, los trenes pueden atravesar la estación, ya que hay combinaciones en que el tren puede ir de un extremo de la estación a otra, por ejemplo:

```
*Esquema1> puedeLlegarTren 1 6 ee1'
True
*Esquema1> puedeLlegarTren 9 14 ee1'
True
```

Ilustración 31-Verificación trenes atraviesan estación con incidencia en vía secundaria III en ee1.

Lo mismo se puede comprobar en el esquema *ee2* de corriente alterna. En este caso suponemos que hay una incidencia en la vía secundaria II (cae el sector 10) y los sectores eléctricos de los extremos de la estación en cuestión son 1 y 3:

```
*Esquema2> ee2' = aislarSectorEE 10 ee2
*Esquema2> puedeLlegarTren 1 3 ee2'
True
```

Ilustración 32-Verificación trenes atraviesan estación con incidencia en vía secundaria II en ee2.

Con lo que quedaría verificada la propiedad también en este caso.

Supongamos un nuevo diseño *ee6*, que difiere de *ee2* en que se han eliminado el aislador de sección y el aislador de sección + seccionador que unían vía I y vía III.

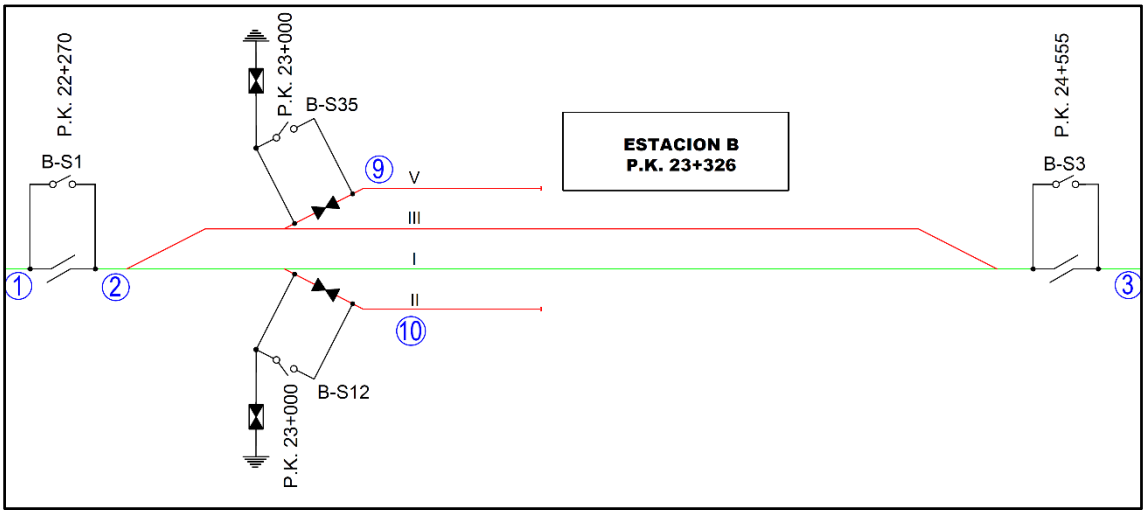

Ilustración 33-Esquema eléctrico ee6.

En este caso, una incidencia en la catenaria de vía secundaria III, plantearía un escenario en el que no se podría atravesar la estación, ya que la vía I no estaría operativa. Al igual que veremos en el próximo ejemplo, el proyectista se podría haber percatado de esta situación rápidamente y sin tener que invertir tiempo y esfuerzo en estudiar las posibles alternativas manualmente en esquema eléctrico:

```
*Esquema2> ee6' = aislarSectorEE 2 ee6
*Esquema2> puedeLlegarTren 3 1 ee6'
False
```

Ilustración 34-Verificación trenes atraviesan estación con incidencia en vía secundaria III en ee6.

Supongamos un nuevo diseño de esquema eléctrico (lo llamaremos *ee7*) realizado por el proyectista, que es como el de *ee1* pero con las siguientes modificaciones:

- Sin los aisladores de sección en los escapes entre vías principales
- Sin el aislador de sección y el aislador de sección + seccionador en los escapes de vía III.

Así, el diseño quedaría de la siguiente manera:

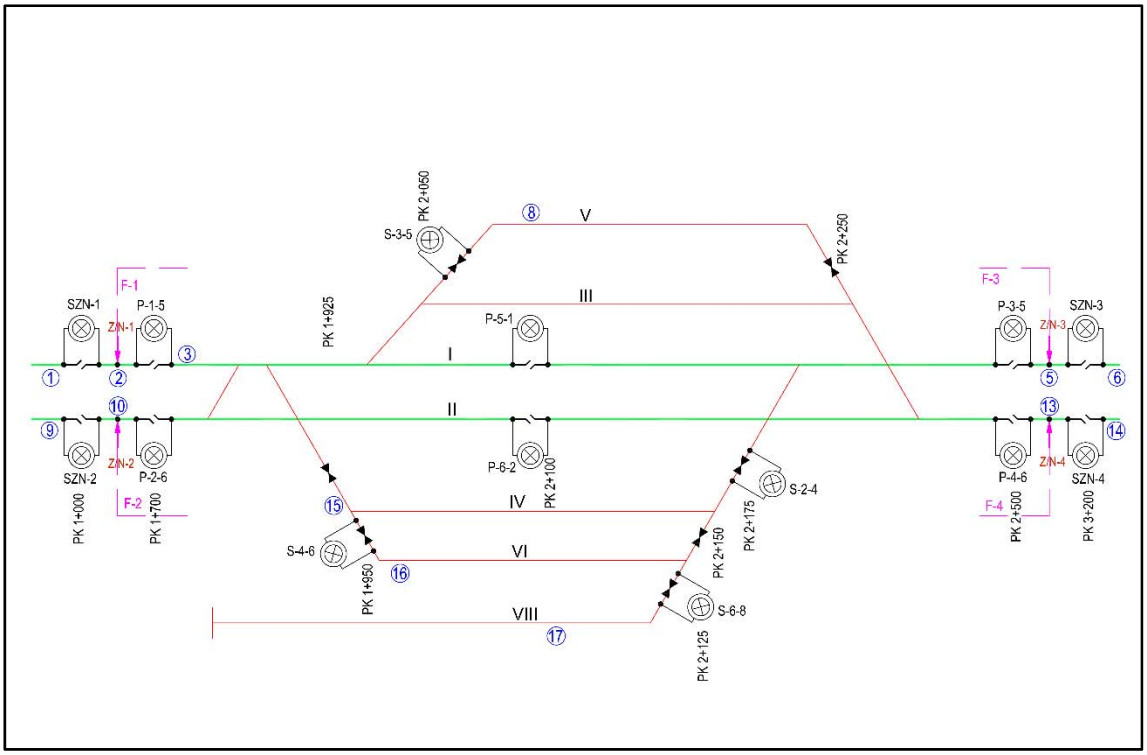

Ilustración 35-Esquema eléctrico ee7

Con este diseño ante un fallo en la catenaria de vía III se impide la circulación del tren de un lado al otro de la estación. De esta forma, el proyectista utilizando la herramienta propuesta habría advertido esta situación de manera automática y sin tener que hacer seguimientos manuales en el esquema eléctrico. Como se muestra a continuación:

```
*Esquema1> ee7' = aislarSectorEE 3 ee7
*Esquema1> puedeLlegarTren 1 6 ee7'
False
*Esquema1> puedeLlegarTren 1 14 ee7'
False
*Esquema1> puedeLlegarTren 9 14 ee7'
False
*Esquema1> puedeLlegarTren 9 6 ee7'
False
```

Ilustración 36-Verificación trenes atraviesan estación con incidencia en vía secundaría III en ee7.

5.3.2.5 - COMPROBACIÓN SI ANTE INCIDENCIA EN VÍA SECUNDARIA SE PRODUCE AFECCIÓN A VÍA PRINCIPAL

- Descripción de característica a verificar y problemática.

Esta propiedad por verificar refleja la necesidad de comprobar que una incidencia en vía secundaria no afecta a vía principal.

En este sentido, un efecto muy negativo para la explotación de la infraestructura y, por tanto, nada deseado por el cliente, es que un mal diseño del esquema eléctrico, con el correspondiente mal reparto de los sectores eléctricos, origine que, ante una incidencia en una vía de importancia menor en el sistema, los trenes se queden parados o no puedan circular por otra vía de importancia clave, como es el caso de las vías principales

CAIDA DE UN SECTOR Y MODO DE FUNCIONAMIENTO EN DEGRADADO

Para ayudar a la comprensión de esta verificación, es necesario explicar el concepto de caída de un sector.

El concepto de caída de un sector se refiere al hecho que, por una determinada incidencia en línea, un sector eléctrico, que en condiciones normales de funcionamiento estaría alimentado, deja de estarlo. En estas condiciones el sistema funciona en lo que se denomina modo degradado.

Dicha incidencia se puede originar principalmente por:

- Avería en la subestación.
- El pantógrafo “engancha” la catenaria.

- ❖ Avería en un feeder de alimentación.
- ❖ Avería en un seccionador.
- ❖ Cortocircuito de la catenaria.
- ❖ Rotura de la catenaria.

- o <u>Especificación de propiedad</u>

$$esSectorBienAlimentadoConCaida(s1, s2, ee)$$
$$resultado$$
$$= (EsSectorFuente(s1)$$
$$\vee\ hayFuente(sectoresAccesiblesEE(s1, cee', recorrido), elee)$$

DONDE:

$$cee' = aislarSectorEE\ (s2, ee)$$
$$recorrido = RELE\ \vee RESEC$$

$$aislarSectorEE(s, ee)$$

$$resultado = a\tilde{n}adirCaido(s, sc) \wedge$$
$$(\forall configuracion \in \{1..aristasND(gee)\}\ \cdot$$
$$\forall i \in \{1..longitud(aristasND(gee))\}$$
$$\cdot (equipo(configuracion_i) = SA\ \vee equipo(configuracion_i) = SS)$$
$$\longrightarrow abrirSeccionador(equipo(configuracion_i))$$

- o <u>Verificación de propiedad en esquema eléctrico</u>

En la solución implementada se ofrece la posibilidad de poder realizar los trabajos de verificación asociados a comprobar que una incidencia en vía secundaria no repercute en vía principal.

A continuación, comentamos detalles importantes para ello:

- Para aportar una solución más flexible el usuario podrá introducir los sectores que desea verificar ante la caída de otro, por si en alguna determinada situación requiere la comprobación de algún sector en especial.

- La verificación se realiza tanto considerando el estado puntual del esquema eléctrico en un momento dado como en el diseño general en condiciones habituales de funcionamiento. Esta doble funcionalidad se le ofrece al usuario a través de un parámetro booleano en la función.

- Se puede consultar la implementación con detalle en el anexo de listado de código (sectoresVpAlimentadosConCaidaSectorVs).

Veamos sobre el ejemplo del esquema eléctrico de continua *ee1*, como se realizaría la verificación. Sí queremos ver si un conjunto de sectores eléctricos (consideramos los que forman parte de vía principal) se ve afectado por la caída de otro (en este caso como en el diseño hay algunos que sólo están en vía secundaria consideramos alguno de ellos, en este caso 15 y 17) se procederá de la siguiente manera:

```
*Esquema1> sectoresVpAlimentadosConCaidaSectorVs sectoresVpEE1 15 ee1 False
True
*Esquema1> sectoresVpAlimentadosConCaidaSectorVs sectoresVpEE1 17 ee1 False
True
```

Ilustración 37-Verificación incidencias en vía secundarias IV y VIII no afectan vías principales en ee1.

Y, de la misma forma, se puede hacer la verificación en un esquema de corriente alterna (*ee2*):

```
*Esquema2> sectoresVpAlimentadosConCaidaSectorVs sectoresVpEE2 10 ee2 False
True
*Esquema2> sectoresVpAlimentadosConCaidaSectorVs sectoresVpEE2 12 ee2 False
True
```

Ilustración 38-Verificación incidencias en vía secundarias II y V no afectan vías principales en ee2.

Aunque no lo desarrollaremos de manera íntegra para no ser repetitivos y complicar más la comprensión, ya que el nuevo diseño originaría cambios de cierta entidad sobre el original y habría que volver a realizar todos los pasos ya descritos y realizados anteriormente (además casos con una estructura similar se han considerado en el punto 5.3.2.4), veamos un ejemplo de la problemática que se podría causar en caso de no verificar esta propiedad adecuadamente de manera manual. Supongamos que el técnico realizase un diseño sin el aislador de sección y el aislador de sección+seccionamiento existente en los desvíos de vía I a vía III.

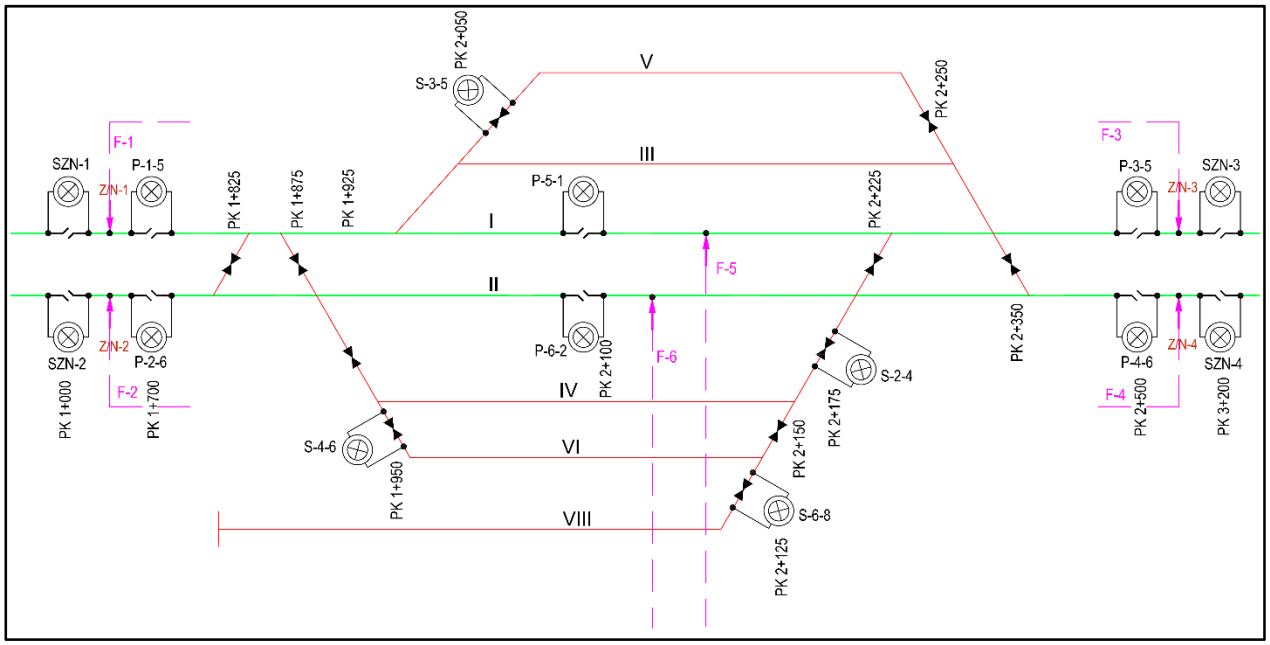

Ilustración 39-Modificación ee1 (sin el aislador de sección y el aislador de sección+seccionamiento en desvíos de vías I-III).

Este diseño implica cambios conceptuales de peso respecto al inicial *ee1* ya que, por ejemplo, además de no tener los equipos eléctricos eliminados ya comentados, el sector eléctrico 7 desaparece. Así, vemos que la via III pasa a formar parte en su totalidad de otro sector eléctrico que también incluye zonas de vía principal. Esto origina consecuencias nefastas, puesto que una incidencia en vía III dejaría sin servicio la vía I. Esta circunstancia la podría haber detectado con nuestra herramienta el proyectista sin tener que hacer seguimiento manual, al verificar los sectores de vía principal ante una incidencia en vía III que motivaría la caída del sector que lo contiene. Lógicamente, la caída de dicho sector eléctrico origina una problemática de funcionamiento del sistema que hace que a la vía I no le pueda llegar ninguna fuente y esto se detecta y se tiene en cuenta en nuestra implementación, concretamente, a la hora de tratar la operativa ante la incidencia en un sector y la caída del mismo (el sector caído se incorpora al conjunto de sectores caídos y se abren todos sus seccionadores).

De la misma manera si en el esquema de corriente alterna *ee2* quitamos en los desvíos de vía I a vía III el aislador de sección y el aislador de sección+seccionador existentes:

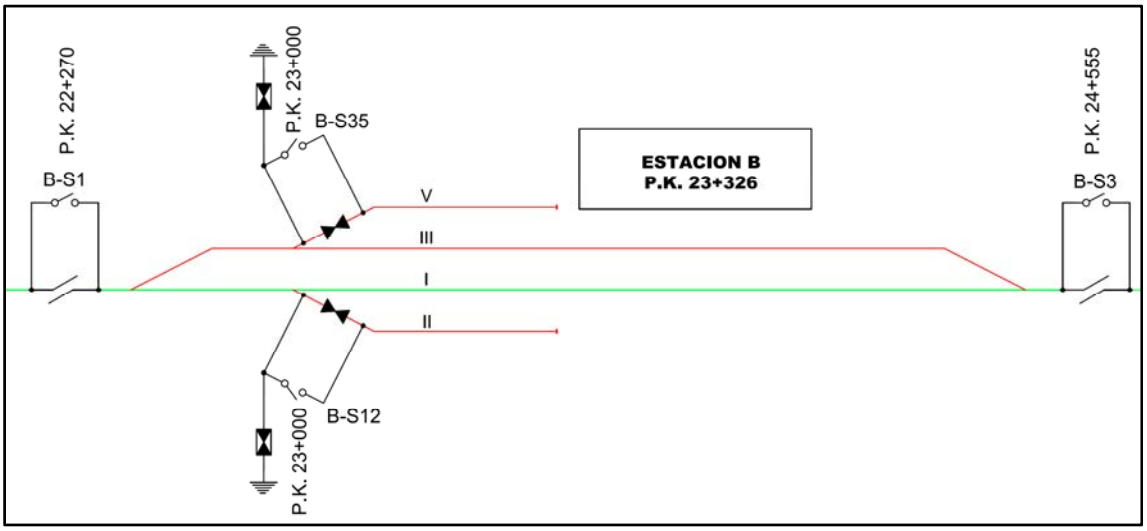

Ilustración 40-Modificación ee2 (sin el aislador de sección y el aislador de sección+seccionamiento en desvíos de vías I-III).

Vemos que un fallo en vía III dejaría sin servicio a la vía I, por el mismo razonamiento ya comentado anteriormente en el caso del diseño en corriente continua.

5.3.2.6 - ZONAS NEUTRAS COMPATIBLES CON LONGITUDES DE LOS TRENES

- Descripción de característica a verificar y problemática.

En esquemas eléctricos de corriente alterna se debe comprobar que las zonas neutras de separación entre subestaciones poseen una longitud mínima de 402m.

Un mal diseño que incumpla esta regla podría derivar en la siguiente problemática: en corriente alterna podría producirse un puenteo de fases diferentes en la zona neutra al circular por la misma una composición con los pantógrafos levantados y unidos en ambos extremos que pondría en contacto ambas fases y, por tanto, provocarían un cortocircuito entre distintas fases en corriente alterna.

- Especificación de propiedad

$$cumpleDistanciaZnEE(s)$$
$$resultado = longitudSector(s) > 402\,m$$

$$cumpleDistanciaTodasZnEE(ee)$$
$$resultado = \forall sector \in \{1..nodos(gee)\} \cdot$$
$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot$$
$$(EsZn(sector_i) \longrightarrow cumpleDistanciaZnEE(sector_i)))$$

- Verificación de propiedad en esquema eléctrico

El técnico proyectista puede realizar las verificaciones anteriormente descritas utilizando la solución implementada de la manera que se mostrará a continuación.

En el esquema eléctrico *ee2* de corriente alterna podemos verificar que la zona neutra que separa las estaciones A y B (incluida en el sector eléctrico 4) cumple la propiedad buscada:

```
*Esquema2> cumpleDistanciaZnEE 4 ee2
True
```

Ilustración 41-Verificación longitud Zona Neutra de ee2.

No obstante, si la zona neutra tuviese una longitud inferior a los 402 m requeridos, la verificación debe fallar. Veamos por ejemplo que pasaría si el pk final del sector 4 pasase a ser 29.200 en vez de 29.454 que tenía anteriormente:

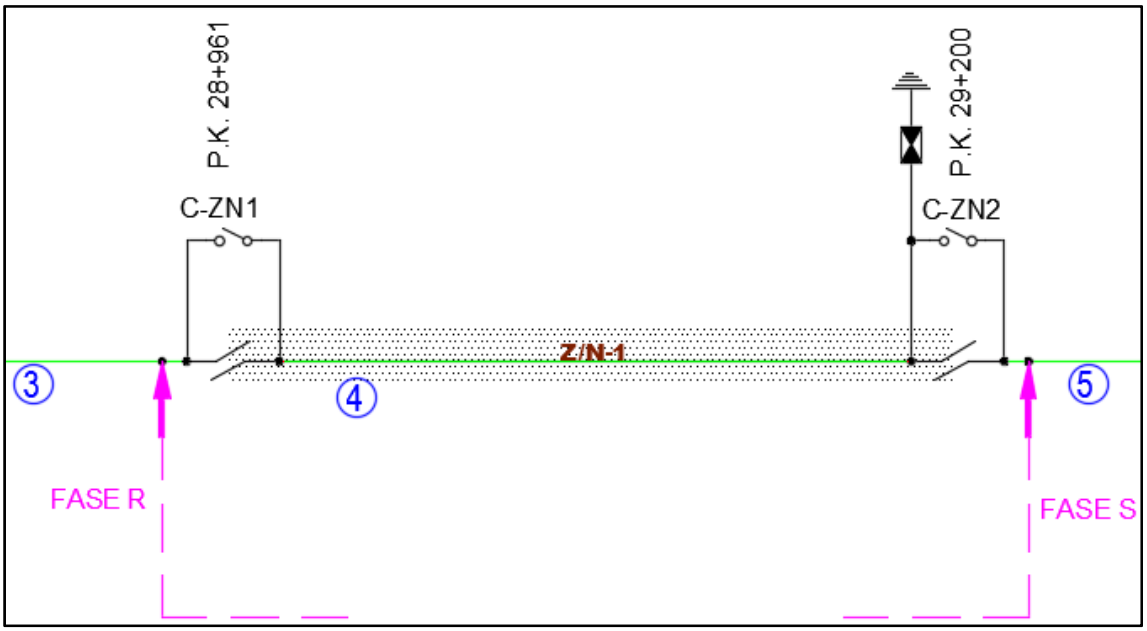

Ilustración 42-Modificación ee2 (cambia pk final de ZN).

Por tanto, la ubicación del sector 4 varía:

```
(4, ZN, ((28.961,(1,1)), (29.200,(1,1))))
```

Ilustración 43-Sector 4 con pk actualizado.

Y al verificar la propiedad, (teniendo en cuenta que el esquema eléctrico con la anterior variación pasase a llamarse *ee2'*) obtenemos que el resultado de dicha comprobación es falso:

```
*Esquema2> cumpleDistanciaZnEE 4 ee2'
False
```

Ilustración 44-Verificación longitud Zona Neutra con pk final modificado en ee2.

Si se desea se puede consultar la implementación con detalle en el anexo de listado de código (cumpleDistanciaZnEE, cumpleDistanciaTodasZnEE).

5.3.2.7 - COMPROBACIÓN DE LA DISTANCIA DE LA SEÑAL DE DETENCIÓN AL SECCIONAMIENTO DE LÁMINA DE AIRE MÁS PRÓXIMO. EJEMPLO DE INTERACCIÓN CON OTRA TÉCNICA.

- Descripción de característica a verificar y problemática.

En la actualidad cada vez es más importante gestionar correctamente la interacción entre las distintas técnicas ferroviarias. Esto es debido principalmente a dos razones:

- Cada vez es más habitual entregar proyectos conjuntos de varias técnicas.

- El mercado cada vez más competitivo en el que nos encontramos implica tener que ajustar los tiempos de realización de proyectos y los recursos invertidos. Esto hace que sea muy necesaria la gestión adecuada de la interacción de las diversas técnicas, ya que pueden ocurrir aspectos no deseables como: cambios en soluciones específicas de una técnica que afectan a otras (estas últimas lo desconocen), cambios en los criterios o condiciones de partida de unas técnicas que influyen en otras y estas no tenían constancia de ello etc. que originan graves problemas en los proyectos tanto en lo económico como en lo que a plazo se refiere, repercutiendo al final en otro aspecto básico como es la calidad del proyecto realizado.

De esta forma, la idea de la solución planteada en el presente Trabajo Fin de Máster es que permita la interacción con otras técnicas de manera sencilla, de tal forma que sus equipos, particularidades y soluciones puedan ser incorporadas y tenidas en cuenta por el resto de técnicas de manera sencilla. Así, todas las técnicas ferroviarias pueden desarrollar su labor en un marco de trabajo y escenario comunes.

En este apartado se pone de manifiesto lo anteriormente comentado, gracias a un ejemplo de interacción con otra técnica ferroviaria diferente a la técnica de Electrificación, que en este caso será la técnica de Control, Mando y Señalización.

Concretamente, la propiedad a estudio es la verificación de la distancia de las señales de detención con el seccionamiento de lámina de aire más próximo, ya que dicha distancia debe respetar una distancia mínima (160 m en caso de corriente continua y 402 m en caso de alterna), con objeto de que un tren que se

detenga en la señal no puentee con sus pantógrafos los seccionamientos de lámina de Aire.

Aunque existen otro tipo de señales como son las de proximidad, en nuestro caso para verificar esta propiedad las que se necesita tener en cuenta (ya que son a las que aplica la norma) son las de detención o parada. Por tanto, la técnica de Control, Mando y Señalización aporta al sistema la información necesaria de dichas señales.

En caso de que la propiedad no se cumpla se planteará la siguiente problemática en la que un seccionamiento de lámina de aire puede tener el seccionador abierto con objeto de no tener uno de los dos sectores que comunica alimentados y que un tren que se encuentre detenido en la señal y con los dos pantógrafos elevados puentee dicho seccionamiento con sus pantógrafos y ponga en tensión el sector eléctrico que se quería mantener aislado, ocasionando un problema grave en la seguridad de la instalación de electrificación.

De esta forma, para ver unos ejemplos de cómo se realizaría la verificación de esta propiedad hemos de incluir las señales de parada en el esquema eléctrico de continua *ee1*.

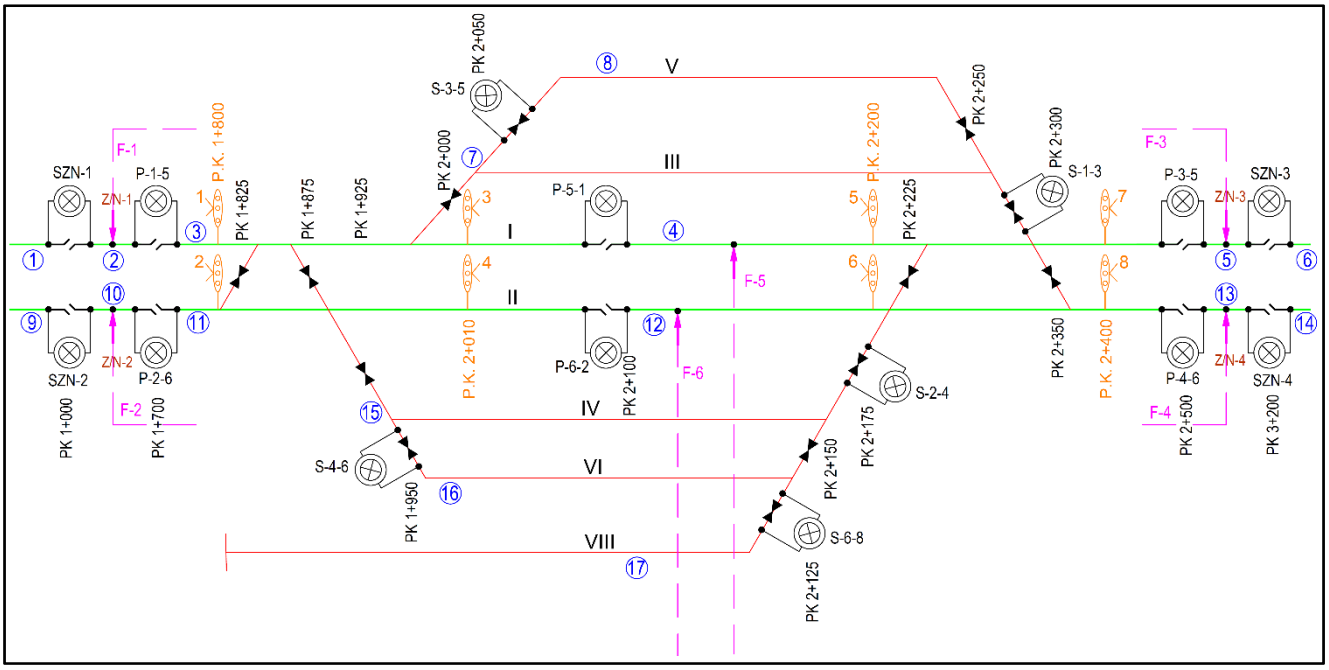

Ilustración 45-Esquema eléctrico ee1 con señales de parada.

Haremos lo propio para el esquema eléctrico de corriente alterna *ee2*.

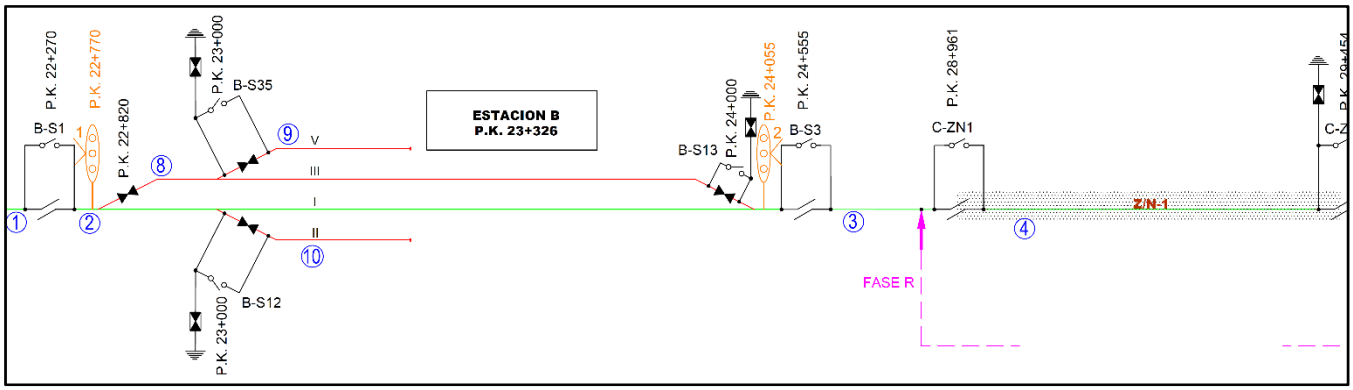

Ilustración 46-Esquema eléctrico ee2 con señales de parada (parte 1).

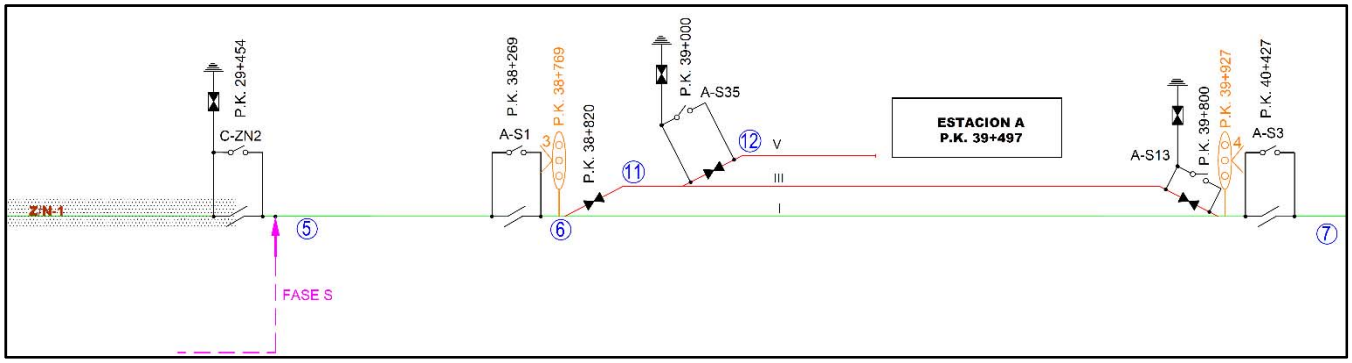

Ilustración 47-Esquema eléctrico ee2 con señales de parada (parte 2).

- Especificación de propiedad

$$compruebaSeñalSeccionamientoEE(ee)$$

$$resultado = \forall sector \in \{1..nodos(gee)\} \cdot$$

$$(\forall i \in \{1..longitud(nodos(gee))\} \cdot (\forall señal \in sector_i . compruebaSeñalSeccionamiento(señal, ee)))$$

$$compruebaSeñalSeccionamiento(señal, ee)$$

$$resultado = (EsSeñalParada(señal) \longrightarrow cumpleNormaDistanciaSeñalPkSeccionamiento\ (señal, pkseccmasprox, ee)$$

$$DONDE:$$
$$pkseccmasprox = pkSeccionamientoMasProximo(señal, ee)$$

$$pkSeccionamientoMasProximo(señal, ee)$$

$$resultado = \begin{cases} máximo\ (pksAdyacentesSS) & si\ Orientacion(señal) = PKC \\ mínimo\ (pksAdyacentesSS) & si\ Orientacion(señal) = PKD \end{cases}$$

$$DONDE:$$
$$pksAdyacentesSS = \{pk \in \mathbb{R} \cdot mismaVia(señal, equipo(pk)) \wedge configuracionEquipo(equipo(pk)) = SS \wedge sector(equipo(pk)) \in AdyacentesTrazado(sector(señal), ee)\ \}$$

$$cumpleNormaDistanciaSeñalPkSeccionamiento\ (pkseñal, pksecc, ee)$$

$$resultado = ((alimee = CC) \longrightarrow abs(pk(señal) - pksecc) * 1000) > 160) \vee ((alimee = CA) \longrightarrow abs(pk(señal) - pksecc) * 1000) > 402)$$

- Verificación de propiedad en esquema eléctrico

El usuario podrá realizar la verificación sobre cada señal en particular o verificar todas las señales de detención existentes en el esquema eléctrico simultáneamente. Para ello, el sistema poseerá la información actualizada

correspondiente a las señales de la que dispone la técnica de Control, Mando y Señalización. Así, el sistema ya dispondrá de los datos de las señales que poseen los esquemas eléctricos *ee1* y el *ee2*.

```
señal1 = construirSeñal(1.800,(1,1)) 3 PARADA PKD
señal2 = construirSeñal(1.800,(2,2)) 11 PARADA PKD
señal3 = construirSeñal(2.010,(1,1)) 3 PARADA PKC
señal4 = construirSeñal(2.010,(2,2)) 11 PARADA PKC
señal5 = construirSeñal(2.200,(1,1)) 4 PARADA PKD
señal6 = construirSeñal(2.200,(2,2)) 12 PARADA PKD
señal7 = construirSeñal(2.400,(1,1)) 4 PARADA PKC
señal8 = construirSeñal(2.400,(2,2)) 12 PARADA PKC

señalesEE1 = [señal1, señal2, señal3, señal4, señal5, señal6, señal7, señal8]
```

Ilustración 48-Señales de parada de esquema eléctrico ee1.

```
señal1 = construirSeñal(22.770,(1,1)) 2 PARADA PKD
señal2 = construirSeñal(24.055,(1,1)) 2 PARADA PKC
señal3 = construirSeñal(38.769,(1,1)) 6 PARADA PKD
señal4 = construirSeñal(39.927,(1,1)) 6 PARADA PKC

señalesEE2 = [señal1, señal2, señal3, señal4]
```

Ilustración 49-Señales de parada de esquema eléctrico ee2.

Como podemos observar los datos necesarios a conocer respecto a las señales son:

- Ubicación: punto kilométrico y vías.
- Tipo de Señal: de parada o de proximidad.
- Orientación de la Señal
    - PKD: La señal “mira” hacia PKs decrecientes.
    - PKC: La señal “mira” hacia PKs crecientes.

Verifiquemos cada una de las señales del esquema eléctrico *ee1*:

```
*Esquema1> compruebaSeñalSeccionamiento señal1 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal2 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal3 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal4 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal5 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal6 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal7 ee1
False
*Esquema1> compruebaSeñalSeccionamiento señal8 ee1
False
```

Ilustración 50-Verificación individual de distancia señales parada-seccionamiento más próximo en ee1

Como podemos observar, la verificación en todas las señales no es satisfactoria. Por tanto, la verificación sobre el esquema eléctrico resultará errónea (con que fallase en una señal ya lo sería):

```
*Esquema1> compruebaSeñalesSeccionamientoEE señalesEE1 ee1
False
```

Ilustración 51-Verificación todas señales parada-seccionamiento más próximo en ee1.

Por ello, hemos visto que al interactuar con la técnica de Control, Mando y Señalización y, concretamente con sus señales de parada, hemos de modificar nuestro diseño para que cumpla la norma.

Así, llevamos las señales 1 y 2 del PK 1+800 al 1+870:

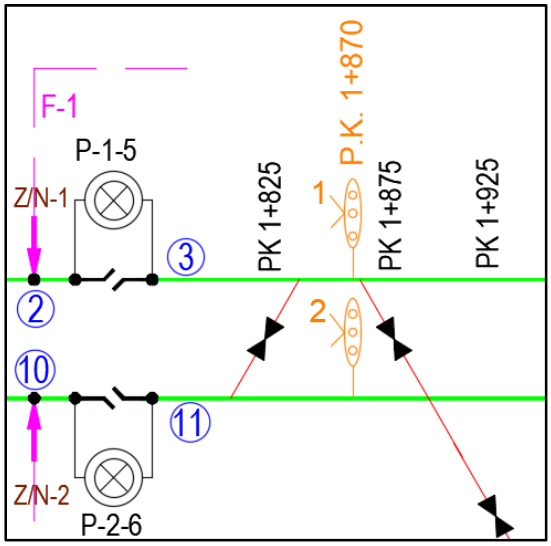

Ilustración 52-Modificación pk señales 1 y 2 de ee1.

```
señal1 = construirSeñal(1.870,(1,1)) 3 PARADA PKD
señal2 = construirSeñal(1.870,(2,2)) 11 PARADA PKD
```

Ilustración 53-Señal 1 y señal 2 de ee1 con pk actualizado.

Realizamos la verificación:

```
*Esquema1> compruebaSeñalSeccionamiento señal1 ee1
True
*Esquema1> compruebaSeñalSeccionamiento señal2 ee1
True
```

Ilustración 54-Verificación señal 1 y señal 2, distancia señal-seccionamiento más próximo en ee1 con pks modificados.

Como vemos, con la corrección realizada en el diseño, la verificación de la propiedad aplicada a las señales 1 y 2 devuelve resultados correctos.

De la misma manera, se podría realizar la modificación del diseño en el resto de las señales para conseguir que sea correcto. Veamos ahora un ejemplo de cómo se haría de forma similar para señales que miran hacia PK creciente, por

ejemplo, las señales 7 y 8. Así, modificamos el diseño y llevamos las señales 7 y 8 del PK 2+400 al PK 2+330.

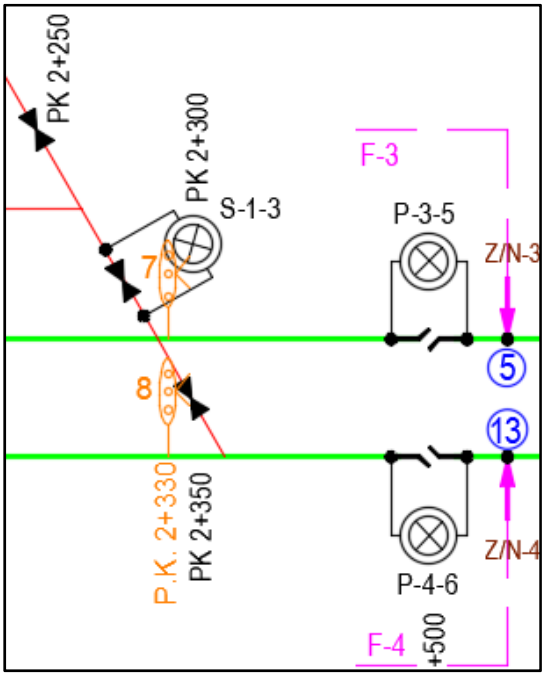

Ilustración 55-Modificación señales de parada 7 y 8 en ee1.

```
señal7 = construirSeñal(2.330,(1,1)) 4 PARADA PKC
señal8 = construirSeñal(2.330,(2,2)) 12 PARADA PKC
```

Ilustración 56-Señal 7 y señal 8 de ee1 con pk actualizado.

```
*Esquema1> compruebaSeñalSeccionamiento señal7 ee1
True
*Esquema1> compruebaSeñalSeccionamiento señal8 ee1
True
```

Ilustración 57-Verificación señales 7 y 8 distancia de parada-seccionamiento más próximo con pk modificado.

Hagamos también algunos ejemplos de verificaciones en el esquema eléctrico *ee2*. Tanto de las señales individualmente:

```
*Esquema2> compruebaSeñalSeccionamiento señal1 ee2
True
*Esquema2> compruebaSeñalSeccionamiento señal2 ee2
True
*Esquema2> compruebaSeñalSeccionamiento señal3 ee2
True
*Esquema2> compruebaSeñalSeccionamiento señal4 ee2
True
```

Ilustración 58-Verificación individual de distancia señales parada-seccionamiento más próximo en ee2

Como de todas las señales de parada del esquema eléctrico:

```
*Esquema2> compruebaSeñalesSeccionamientoEE señalesEE2 ee2
True
```

Ilustración 59-Verificación todas señales parada-seccionamiento más próximo en ee2.

Como vemos el diseño de le esquema eléctrico *ee2* es adecuado y el resultado de la verificación de la propiedad a estudio es correcto.

Consideremos que en el esquema *ee2* la señal1 se encontrase en el punto kilométrico 22+370 en vez de en el 22+770 y que la señal4 se encuentre en el 40+227 en vez de en el 39+927:

```
señal1 = construirSeñal(22.370,(1,1)) 2 PARADA PKD
```

Ilustración 60-Señal 1 con pk actualizado.

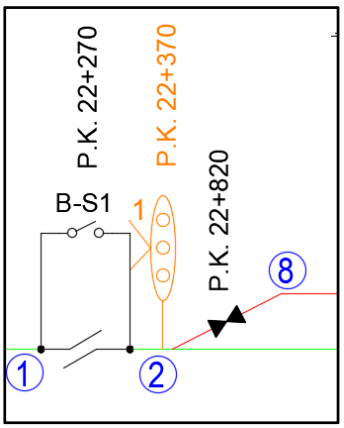

Ilustración 61-Modificación pk señal 1 de ee2.

```
señal4 = construirSeñal(40.227,(1,1)) 6 PARADA PKC
```

Ilustración 62-Señal 4 con pk actualizado.

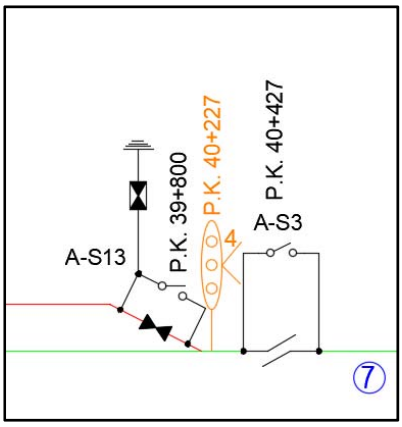

Ilustración 63-Modificación pk señal 4 de ee2.

De esta forma, la verificación de la propiedad en ambas señales no debería cumplirse:

```
*Esquema2> compruebaSeñalSeccionamiento señal1 ee2
False
*Esquema2> compruebaSeñalSeccionamiento señal4 ee2
False
```

Ilustración 64-Verificación señales 1 y 4 distancia de parada-seccionamiento más próximo (con pks modificados).

Y, por tanto, en todo el esquema eléctrico tampoco, ya que el hecho de que hubiese una señal que no cumpla la verificación hace que la comprobación en todo el esquema eléctrico no sea satisfactoria:

```
*Esquema2> compruebaSeñalesSeccionamientoEE señalesEE2 ee2
False
```

Ilustración 65-Verificación todas señales parada-seccionamiento más próximo en ee2 (con pks modificados).

Si se desea se puede consultar la implementación con detalle en el anexo de listado de código (compruebaSeñalSeccionamiento, compruebaSeñalesSeccionamientoEE).

6 - CONCLUSIONES Y LOGROS ALCANZADOS

Con la solución aportada se han alcanzado diversos logros que se plantearon al inicio de la realización del presente Trabajo Fin de Máster. De esta manera, la solución aportada e implementada:

- Resuelve un problema habitual a abordar en todos los proyectos de electrificación ferroviaria, como es la comprobación de requisitos críticos que deben cumplir los diseños.

- Minimiza el tiempo invertido por los técnicos para la verificación de los diseños realizados. De hecho, el tiempo disminuye de manera más que significativa, ya que la realización manual por parte del proyectista de la verificación de todas las propiedades del sistema es un proceso largo y que implica muchas horas de trabajo.

- Minimiza la probabilidad de error. Al desarrollar una herramienta software que aporta la realización de las tareas del experto de forma automática se minimizan los errores, especialmente debido a que en la actualidad el técnico debe invertir varias horas seguidas en ello y como sabemos eso también aumenta la posibilidad de error.

- Mayor seguridad en un sistema crítico. Como consecuencia del punto anterior, que hace hincapié en el hecho de que la solución implementada minimiza futuros errores y, unido a que se aporta una especificación matemática de las propiedades a verificar, se dota a la herramienta de la solidez, robustez y las garantías necesarias a la hora de abordar problemas dentro del ámbito de los sistemas críticos.

- Mayor poder de reacción y en menor tiempo a nuevas condiciones de partida, cambios o variación de requisitos del cliente. Este aspecto

aporta valiosos beneficios no sólo a nivel económico (puesto que menos horas invertidas derivan en menor coste), sino a efectos de satisfacción del cliente.

- Dentro del ámbito ferroviario es cada vez más común el hecho de que todas las técnicas ferroviarias involucradas en un proyecto trabajen de forma simultánea, de esta forma es de vital importancia tener herramientas a las que se le puedan añadir funcionalidades para ese cometido, sin riesgo de que se produzcan cambios en los escenarios compartidos cuando esas otras técnicas introduzcan sus datos e información de partida.

- Software fácilmente mantenible que permite añadir funcionalidades en tiempos cortos de desarrollo. El hecho de que se parta de la especificación de las propiedades y la utilización de programación funcional que encaja muy bien con el dominio del problema, hace que añadir funcionalidades sea un proceso rápido. Esto es importante dentro del ámbito ferroviario, ya que una ligera variación en una norma implica tener que añadir una funcionalidad nueva o modificar otra existente y, por ello, el hecho de que estas acciones se puedan realizar de forma solvente en el software realizado aporta gran valor.

Como conclusiones me gustaría exponer que ha sido muy enriquecedor realizar tareas como:

- Analizar qué recursos utilizar para realizar la solución: paradigma de programación, recopilación de normativa necesaria, metodología de trabajo, forma de especificación que mejor se adapte al dominio y a las necesidades del problema, etc. que me ha permitido poner en práctica también un rol analítico aparte del de desarrollador.

- Abordar un problema desde un punto de vista transversal a dos ámbitos como el de la Ingeniería Informática y la Ingeniería Ferroviaria con las particularidades de ambas, de tal manera que se ha convergido a la solución planteada en el presente trabajo. Así, ha habido tareas a realizar que han resultado vitales para el buen devenir del producto final como la toma de requisitos, consulta de bibliografía, estudio de diversos casos relacionados con el problema para que la solución cubra todas las alternativas posibles etc.

Por último, me gustaría destacar que me ha resultado especialmente interesante la realización del trabajo, ya que mezcla el sector profesional del que formo parte con los estudios que he realizado, aportando además una solución práctica a un problema real.

7 - FUTUROS TRABAJOS

Gracias a los logros alcanzados se pueden plantear futuros trabajos que sigan aportando valor dentro de esta línea de conocimiento. Entre ellos son destacables:

- Conexión de la solución software creada con programas CAD. De esta forma la entrada y salida de la información se podrá utilizar procedente de los planos de manera directa.

- Trabajar en la integración con los trabajos de otras técnicas ferroviarias (vía, obra, civil, telecomunicaciones, protecciones de seguridad, etc ..).

- Una vez que hemos conseguido una herramienta de verificación de diseños de esquemas eléctricos se podrán considerar trabajos de optimización en los mismos. De esta manera, se podría llegar al diseño del esquema eléctrico más óptimo partiendo de las condiciones específicas de cada caso y las directrices marcadas por el cliente.

8 - BIBLIOGRAFÍA

- J O'Donnel y C. Hall, Discrete Mathematics Using a Computer, Springer, 2006.
- J. L. Gross y J. Yellen, Graph Theory and Its Applications, CRC Press, 2005.
- F.J., González Fernández y J. Fuentes Losa, Ingeniería Ferroviaria, UNED, 2010.
- F. Kiessling, R. Puschmann, A.Schmieder y T. Vega, Líneas de contacto para ferrocarriles electrificados, Siemens, 2008.
- M. Huth y M. Ryan, Logic in Computer Science, Cambridge University Press, 2004.
- M. Ben-Ari, Mathematical Logic for Computer Science, Springer, 2012.
- R.Bird, Pearls of functional algorithm design, Cambridge University Press, 2010.
- G.Hutton, Programming in Haskell, Cambridge University Press, 2016.
- J. Pachl, Railway Operation and Control, VTD Rail Publishing, 2002.
- B. O'Sullivan, J. Goerzen y D. Stewart, Real World Haskell, O'Reilly, 2009.
- V.S. Alagar y K. Periyasamy, Specification of Software Systems, Springer 2011.
- Normativa Adif: http://descargas.adif.es/ade/u18/GCN/NormativaTecnica.nsf (última consulta 4 de Mayo de 2018)

9 – LISTADO DE SIGLAS, ABREVIATURAS Y ACRÓNIMOS

AI: Aislador de sección.

CAD: Computer-aided design.

GHC: Glasgow Haskell Compiler.

LAAT: Línea aérea de alta tensión.

PK: Punto kilométrico.

SA: Aislador de sección + seccionador.

SS: Seccionamiento de lámina de aire + seccionador.

10- ANEXOS

10.1 - PLAN DE ACTIVIDADES

	ACTIVIDADES DEL TFM	2017		2018						
		Nov	Dic	Ene	Feb	Mar	Abr	May	Jun	Jul
1	**TRABAJOS PREVIOS**									
	Elaboración de Propuesta de TFM									
	Primeros contactos con futuro director del TFM									
	Aceptación de dirección de TFM									
2	**ESTUDIO DEL PROBLEMA**									
	Intorducción en el dominio del problema									
	Estudio de las propiedades a verificar									
	Especificación de propiedades									
3	**SOLUCIÓN AL PROBLEMA**									
	Análisis de recursos y tecnologías a utilizar									
	Preparación de ejemplos relevantes									
	Implementación de la solución									
4	**REDACCIÓN DE MEMORIA**									
	Definición de estructura del documento									
	Redacción del documento									
5	**EXPOSICIÓN DEL TFM**									
	Elaboración de Presentación									
	Presentación									

10.2 - LISTADO DE CÓDIGO

```
module Ubicacion
    (
     TipoPk,
     TipoIdVia,
     TipoParVias,
     TipoUbicacionElemento,
     pkElemento,
     parViasElemento,
     obtenerVia,
     esDesvio,
     longitudEntreDosPks
    ) where


type TipoPk = Float

type TipoIdVia = Int

type TipoParVias = (Int,Int)

-- En desvío los dos Ids de vía distintos
-- En vía los dos Ids de vía iguales
type TipoUbicacionElemento = (TipoPk, TipoParVias)


pkElemento:: TipoUbicacionElemento -> TipoPk
pkElemento (pkelto, _) = pkelto

parViasElemento:: TipoUbicacionElemento -> TipoParVias
parViasElemento (_, parviasid) = parviasid

obtenerVia:: TipoParVias-> TipoIdVia
obtenerVia(v1,v2) = if esVia(v1,v2) then
                          v1
                    else
                          min v1 v2

esDesvio:: TipoParVias -> Bool
esDesvio (v1,v2) = v1/=v2

esVia:: TipoParVias -> Bool
esVia (v1,v2) = v1==v2

longitudEntreDosPks:: TipoPk -> TipoPk -> Float
longitudEntreDosPks pk1 pk2 = (abs(pk1 - pk2)) * 1000
```

```
module Configuracion
    (
     TipoEquipo (..),
     TipoConfiguracion (..),
     equipo,
     estadoConfiguracion,
     ubicacionEquipo,
     pasaCorriente,
     cerrarSeccionador,
     abrirSeccionador,
     hayAisladorSeccion,
     haySeccionador,
     haySeccionadorCerradoEnConfiguraciones,
     hayAisladorSeccionEnConfiguraciones
    ) where

import Ubicacion

-- El equipo electrico puede ser:
--              . seccionador + aislador de sección
--              . seccionador + seccionamiento de lámina de aire
--              . Aislador de sección
data TipoEquipo = SA | SS | AI
                  deriving (Eq, Show)
-- Aislador de sección sin seccionador (AI) siempre cerrado
-- Si hay seccionador puede estar abierto o cerrado
-- Si está cerrado => True
-- Si está abierto => False

type TipoConfiguracion = (TipoEquipo,Bool, TipoUbicacionElemento)

cquipo :: TipoConfiguracion -> TipoEquipo
equipo (e,_,_) = e

estadoConfiguracion :: TipoConfiguracion -> Bool
estadoConfiguracion (_,s,_) = s

ubicacionEquipo ::  TipoConfiguracion ->  TipoUbicacionElemento
ubicacionEquipo  (_,_,ubieq) = ubieq

pasaCorriente :: TipoConfiguracion -> Bool
pasaCorriente confi = estadoConfiguracion(confi) == True

cerrarSeccionador :: TipoConfiguracion -> TipoConfiguracion
cerrarSeccionador (e,s,ubieq) = if (e==SA || e==SS) then
                                (e,True,ubieq)
                            else
                                (e,False,ubieq)

abrirSeccionador :: TipoConfiguracion -> TipoConfiguracion
abrirSeccionador (e,s,ubieq) = (e,False,ubieq)

haySeccionador:: TipoEquipo -> Bool
haySeccionador eq = (eq == SS || eq == SA)

hayAisladorSeccion:: TipoEquipo -> Bool
hayAisladorSeccion eq = (eq == AI)

haySeccionamientoLaminaAire:: TipoEquipo -> Bool
haySeccionamientoLaminaAire eq = (eq == SS)

haySeccionadorCerradoEnConfiguraciones:: [TipoConfiguracion] -> Bool
haySeccionadorCerradoEnConfiguraciones [] = False
haySeccionadorCerradoEnConfiguraciones (x:xs) = (haySeccionador (equipo x) &&
                                                 estadoConfiguracion x == True) ||
                                                 haySeccionadorCerradoEnConfiguraciones xs
```

```
hayAisladorSeccionEnConfiguraciones:: [TipoConfiguracion] -> Bool
hayAisladorSeccionEnConfiguraciones [] = False
hayAisladorSeccionEnConfiguraciones (x:xs) = hayAisladorSeccion (equipo x) ||
                                             hayAisladorSeccionEnConfiguraciones xs
```

```
module SectorElectrico
    (
    TipoIdSector,
    TipoZona(..),
    TipoSectorElectrico,
    identificadorSector,
    zonaSector,
    ubicacionSector,
    pkInicial,
    pkFinal,
    parViasInicial,
    parViasFinal,
    longitudUbicacion,
    longitudSector,
    esZn
    ) where


import Ubicacion
import Data.Array


type TipoIdSector v = v

data TipoZona = ZN | ZS
                  deriving (Eq, Show)


type TipoUbicacionSector = (TipoUbicacionElemento,TipoUbicacionElemento)

type TipoSectorElectrico v = (TipoIdSector v,TipoZona,TipoUbicacionSector)


identificadorSector :: (Ix v) => TipoSectorElectrico v -> TipoIdSector v
identificadorSector(idsector,_ ,_) = idsector

zonaSector:: TipoSectorElectrico v -> TipoZona
zonaSector (_,z,_) = z

ubicacionSector:: TipoSectorElectrico v -> TipoUbicacionSector
ubicacionSector (_,_,u) = u

pkInicial :: TipoUbicacionSector -> TipoPk
pkInicial  ((pkin,_), _) = pkin;

pkFinal :: TipoUbicacionSector -> TipoPk
pkFinal (_, (pkfin,_)) = pkfin;

parViasInicial:: TipoUbicacionSector -> TipoParVias
parViasInicial(ubielto1,_) = parViasElemento ubielto1

parViasFinal:: TipoUbicacionSector -> TipoParVias
parViasFinal(_,ubielto2) = parViasElemento ubielto2

longitudUbicacion :: TipoUbicacionSector -> Float
longitudUbicacion ((pkin,_), (pkfin,_)) = longitudEntreDosPks pkfin pkin;

esZn:: TipoSectorElectrico v -> Bool
esZn sect = zonaSector sect == ZN

longitudSector:: TipoSectorElectrico v -> Float
longitudSector sect = longitudUbicacion (ubicacionSector sect)
```

```
module Señal
   (
    TipologiaSeñal(..),
    TipoOrientacion (..),
    TipoSeñal,
    TipoSeñalesEE,
    construirSeñal,
    ubicacionSeñal,
    idSectorSeñal ,
    tipologiaSeñal ,
    orientacionSeñal
   ) where


import SectorElectrico (TipoIdSector)
import Ubicacion
import Data.Array


-- Parada ==> Señal de parada (puede detener al tren)
-- Proximidad ==> Señal de proximidad (solo indicadoras de proximidad a señal de parada)
data TipologiaSeñal = PARADA | PROXIMIDAD
                          deriving (Eq, Show)

-- orientación de la señal
-- PKC señal orientada a Pk creciente
-- PKD señal orientada a Pk decreciente
data TipoOrientacion = PKC | PKD
                           deriving (Eq, Show)

type TipoSeñal v = (TipoUbicacionElemento,TipoIdSector v,TipologiaSeñal, TipoOrientacion)

type TipoSeñalesEE v = [TipoSeñal v]


construirSeñal :: (Ix v) -> TipoUbicacionElemento -> TipoIdSector v -> TipologiaSeñal
                                                    -> TipoOrientacion -> TipoSeñal v
construirSeñal ubsen idse tipsen orientsen = (ubsen, idse, tipsen, orientsen)

ubicacionSeñal :: (Ix v) => TipoSeñal v -> TipoUbicacionElemento
ubicacionSeñal (ubicsig,_,_,_) = ubicsig

idSectorSeñal  :: (Ix v) => TipoSeñal v -> TipoIdSector v
idSectorSeñal  (_,sectsig,_,_) = sectsig

tipologiaSeñal  :: (Ix v) => TipoSeñal v -> TipologiaSeñal
tipologiaSeñal  (_,_,sigtipol,_) = sigtipol

orientacionSeñal  :: (Ix v) => TipoSeñal v -> TipoOrientacion
orientacionSeñal  (_,_,_,o) = o
```

```haskell
module SectoresVp
    (TipoSectoresVp,
     construirSectoresVp,
     reiniciarSectoresVp,
     estaEnViaPrincipal,
     estaEnViaSecundaria
    ) where


import SectorElectrico (TipoIdSector)
import Ubicacion
import Data.Array


type TipoSectoresVp v = [TipoIdSector v]


construirSectoresVp :: (Ix v) => [TipoIdSector v] -> TipoSectoresVp v
construirSectoresVp lsvp = lsvp

reiniciarSectoresVp :: (Ix v) => TipoSectoresVp v -> TipoSectoresVp v
reiniciarSectoresVp _ = []

-- Todo el sector eléctrico o parte del mismo se encuentra en Vía principal
estaEnViaPrincipal :: (Ix v) => v -> TipoSectoresVp v -> Bool
estaEnViaPrincipal s lvp = elem s lvp

-- No hay parte del sector eléctrico en Vía Principal
estaEnViaSecundaria :: (Ix v) => v -> TipoSectoresVp v -> Bool
estaEnViaSecundaria s lvp = not (elem s lvp)
```

```
module Electrificacion
    (TipoFase (..),
     TipoFuente,
     TipoSectoresElectricos,
     TipoSectoresCaidos,
     TipoSectoresFuente,
     TipoAlimentacion(..),
     TipoElectrificacion (..),
     creaElectrificacion,
     sectoresElectricos,
     alimentacion,
     sectoresFuente,
     sectoresCaidos,
     esSectorCaido,
     esSectorFuente,
     sectorElectricoxId,
     esContinua,
     esAlterna,
     sacarFuenteDeId,
     mismaFase,
     sacarNumeracionFuentes,
     hayFuente,
     sinCaidos,
     anadirCaido,
     cumpleDistanciaZn,
     cumpleDistanciaTodasZn
     ) where


import SectorElectrico
import Data.Array
import Data.List


-- CC ==> Corriente continua
-- CA ==> Corriente Alterna
data TipoAlimentacion =  CC | CA
                      deriving (Eq, Show)


-- N ==> No hay fase (corriente continua)
-- R ==> Fase R de corriente alterna
-- S ==> Fase S de corriente alterna
-- T ==> Fase T de corriente alterna
data TipoFase = N | R | S | T
                        deriving (Eq, Show)


type TipoFuente v = (TipoIdSector v, TipoFase)

type TipoSectoresCaidos v = [TipoIdSector v]

type TipoSectoresFuente v = [TipoFuente v]

type TipoSectoresElectricos v = [TipoSectorElectrico v]

type TipoElectrificacion v = (TipoSectoresElectricos v,TipoAlimentacion,
                              TipoSectoresFuente v, TipoSectoresCaidos v)


fase :: (Ix v) => TipoFuente v -> TipoFase
fase (_,f) = f

idSectorFuente :: (Ix v) => TipoFuente v -> TipoIdSector v
idSectorFuente(ids,_) = ids

creaElectrificacion :: (Ix v) => (TipoSectoresElectricos v, TipoAlimentacion,
         TipoSectoresFuente v, TipoSectoresCaidos v) -> TipoElectrificacion v
creaElectrificacion (sects, a,lf,lc) = (sects, a, lf, lc)

sectoresElectricos :: (Ix v) => TipoElectrificacion v -> TipoSectoresElectricos v
```

```
sectoresElectricos (sects,_,_,_) = sects

alimentacion :: (Ix v) => TipoElectrificacion v -> TipoAlimentacion
alimentacion (_,a,_,_) = a

sectoresFuente :: (Ix v) => TipoElectrificacion v -> TipoSectoresFuente v
sectoresFuente (_,_,sf,_) = sf

sectoresCaidos :: (Ix v) => TipoElectrificacion v -> TipoSectoresCaidos v
sectoresCaidos (_,_,_,sc) = sc

-- PRE: hay un sector eléctrico con ese id en el EE
sectorElectricoxId:: (Ix v) => TipoSectoresElectricos v -> TipoIdSector v
                                                        -> TipoSectorElectrico v
sectorElectricoxId (x:xs) ids = if identificadorSector(x) == ids then
                                                          x
                                else
                                                        sectorElectricoxId xs ids

esContinua :: (Ix v) => TipoElectrificacion v -> Bool
esContinua (_,a,_,_) = (a == CC)

esAlterna :: (Ix v) => TipoElectrificacion v -> Bool
esAlterna (_,a,_,_) = (a == CA)

esSectorCaido:: (Ix v) => v -> TipoSectoresCaidos v -> Bool
esSectorCaido s lsc = elem s lsc

anadirCaido:: (Ix v) => TipoIdSector v -> TipoSectoresCaidos v -> TipoSectoresCaidos v
anadirCaido s lsc = nub (s:lsc)

sacarFuenteDeId :: (Ix v) => TipoIdSector v -> TipoSectoresFuente v -> TipoFase
sacarFuenteDeId ids (x:xs) = if (ids == idSectorFuente x) then
                                 fase x
                             else
                                 sacarFuenteDeId ids xs

-- comprueba en una lista de identificadores de fuentes que todos tienen la misma fase
mismaFase:: (Ix v) => [TipoIdSector v] -> TipoSectoresFuente v -> Bool
mismaFase [] _ = True
mismaFase [x] _ = True
mismaFase (x:(y:ys)) lf = ((sacarFuenteDeId y lf) == f) && mismaFase (y:ys) lf
                    where f = sacarFuenteDeId x lf

sacarNumeracionFuentes:: (Ix v) => TipoSectoresFuente v -> [TipoIdSector v]
sacarNumeracionFuentes [] = []
sacarNumeracionFuentes (x:xs) = [idSectorFuente(x)] ++ sacarNumeracionFuentes xs


esSectorFuente:: (Ix v) => TipoIdSector v -> TipoSectoresFuente v  -> Bool
esSectorFuente s lsf = elem s (sacarNumeracionFuentes lsf)

-- Quita de una lista de sectores los que están caidos
sinCaidos:: (Ix v) => [TipoIdSector v] -> [TipoIdSector v] -> [TipoIdSector v]
sinCaidos [] _ = []
sinCaidos (x:xs) lsc | esSectorCaido x lsc = sinCaidos xs lsc
                     | otherwise = (x:sinCaidos xs lsc)

-- Ver si en una lista de sectores hay conexión a fuente
hayFuente:: (Ix v) => [TipoIdSector v] -> TipoElectrificacion v -> Bool
hayFuente [] _ = False
hayFuente (x:xs) electr = (esSectorFuente x y &&
                          (not (esSectorCaido x (sectoresCaidos electr)))) ||
                           hayFuente xs electr
                           where y = sectoresFuente electr

-- Comprobar que un sector que es Zona Neutra cumple normativa de seguridad de longitud
-- PRE: esZn (szn)
cumpleDistanciaZn:: TipoSectorElectrico v -> TipoAlimentacion -> Bool
cumpleDistanciaZn szn alim | alim == CC = longitudSector szn > 160
cumpleDistanciaZn szn alim | alim == CA = longitudSector szn > 402
```

```
-- Comprobar que todos los sectores eléctricos que son Zona Neutra de toda la electrificación
-- cumplen normativa de seguridad de longitud
cumpleDistanciaTodasZn:: TipoElectrificacion v -> Bool
cumpleDistanciaTodasZn ([], alim, _, _) = True
cumpleDistanciaTodasZn ((x:xs), alim, lsf, lsc) =
        if (esZn x) then
                    (cumpleDistanciaZn x alim) && (cumpleDistanciaTodasZn (xs, alim,lsf,lsc))
        else
                    (cumpleDistanciaTodasZn (xs, alim,lsf,lsc))
```

```
module GrafoEE
    (
     Grafo (..),
     TipoRecorrido(..),
     TipoRecorridoEE,
     creaGrafo,
     configuracionSector,
     nodos,
     aristasD,
     aristasND,
     adyacentesConfiguraciones,
     adyacentesTrazado,
     sectoresAccesiblesEE,
     aislaSectores,
     adyacentesSeccionador
    ) where


import Data.Array
import Configuracion
import Electrificacion
import Ubicacion


type Grafo v = Array v [([TipoConfiguracion],v)]

-- Recorrido del EE según trazado, Seccionadores o eléctrico
data TipoRecorrido = RTRA | RSEC | RELE
                     deriving (Eq, Show)

-- Booleano inluir nodo inicial
type TipoRecorridoEE = (TipoRecorrido,Bool)


creaGrafo :: (Ix v) => (v,v) -> [([TipoConfiguracion],v,v)] -> Grafo v
creaGrafo cs vs =
    accumArray
     (\xs x -> xs++[x]) [] cs
     ([(x2,(e,x1))|(e,x1,x2) <- vs, x1 /= x2] ++
      [(x1,(e,x2)) | (e,x1,x2) <- vs])

-- configuracion del equipo que une los vértices (sectores) v1 y v2
-- en grafo g
configuracionSector :: (Ix v) => v -> v -> (Grafo v) -> [TipoConfiguracion]
configuracionSector x y g = head [e | (e,a) <- g!x , a == y]

-- elementos del Grafo Esquema Eléctrico
elementos :: (Ix v) => (Grafo v) -> [([[TipoConfiguracion], v)]]
elementos g = elems g

-- Construye Grafo con una lista de Elementos
construyeEE :: (Ix v) => (v,v) -> [([[TipoConfiguracion], v)]] -> Grafo v
construyeEE (s,i) leltos = listArray (s,i) leltos

-- nodos del grafo g.
nodos :: (Ix v) => (Grafo v) -> [v]
nodos g = indices g

-- aristas del grafo dirigido g
aristasD :: (Ix v) => (Grafo v) -> [([TipoConfiguracion],v,v)]
aristasD g = [(e,v1,v2) | v1 <- nodos g , (e,v2) <- g!v1]

-- aristas del grafo no dirigido g
aristasND :: (Ix v) => (Grafo v) -> [([TipoConfiguracion],v,v)]
aristasND g =
    [(e,v1,v2) | v1 <- nodos g , (e,v2) <- g!v1 , v1 < v2]

adyacentesNodos :: (Ix v) => (Grafo v) -> v -> (([TipoConfiguracion], v) -> Bool) -> [v]
adyacentesNodos g v f = map (\(e,a) -> a) ladys
                            where ladys = (filter f (g!v))
```

```
funAdyacenciaSeccionador :: ([TipoConfiguracion], v) -> Bool
funAdyacenciaSeccionador ([],_) = False
funAdyacenciaSeccionador ((x:xs),idv) = (equipo x/=AI) || funAdyacenciaSeccionador(xs, idv)

funAdyacenciaElectrica :: ([TipoConfiguracion], v) -> Bool
funAdyacenciaElectrica  ([],_) = False
funAdyacenciaElectrica ((x:xs),idv)= ((equipo x /=AI) && (estadoConfiguracion x == True)) ||
                                     funAdyacenciaElectrica(xs,idv)

funAdyacenciaTrazado ::([TipoConfiguracion], v) -> Bool
funAdyacenciaTrazado _ = True

adyacentesTrazado :: (Ix v) => (Grafo v) -> v -> [v]
adyacentesTrazado g v = adyacentesNodos g v funAdyacenciaTrazado

adyacentesSeccionador :: (Ix v) => (Grafo v) -> v -> [v]
adyacentesSeccionador g v = adyacentesNodos g v funAdyacenciaSeccionador

adyacentesElectrico :: (Ix v) => (Grafo v) -> v -> [v]
adyacentesElectrico g v = adyacentesNodos g v funAdyacenciaElectrica

-- verifica si a es una arista del grafo g
aristaEn :: (Ix v) => (Grafo v) -> (v,v) -> Bool
aristaEn g (x,y) = elem y (adyacentesTrazado g x)

-- Configuraciones adyacentes a un sector
adyacentesConfiguraciones :: (Ix v) => (Grafo v) -> v -> [([TipoConfiguracion],v)]
adyacentesConfiguraciones g ids = g!ids

seccionamientosMismaVia :: (Ix v) => [(TipoConfiguracion,v)] -> TipoIdVia
                                     -> [(TipoConfiguracion,v)]
seccionamientosMismaVia lcs via =
    filter(\(confi,ids) -> (equipo confi /=SS) &&
                           (obtenerVia (parViasElemento (ubicacionEquipo confi)) /= via))
                           lcs

ubicacionSeccionamientosMismaVia :: (Ix v) => [(TipoConfiguracion,v)] -> TipoIdVia
                                              -> [(TipoPk)]
ubicacionSeccionamientosMismaVia lcs via =
     map (\((e,p,(pke,ve)),ids) -> pke) lcfids
     where lcfids = filter(\(confi,ids) ->
                           (equipo confi /=SS) &&
                           (obtenerVia (parViasElemento (ubicacionEquipo confi)) /= via))
                           lcs

pkMayoresSeccionamientosMismaVia :: (Ix v) => [(TipoConfiguracion,v)]
                                              -> TipoUbicacionElemento -> [(TipoPk)]
pkMayoresSeccionamientosMismaVia lcs (pks,(via1,via2)) =
     map (\((e,p,(pke,ve)),ids) -> pke) lcfids
     where lcfids = filter(\(confi,ids) ->
                       (equipo confi /=SS) &&
                       (pkElemento (ubicacionEquipo confi) < pks) &&
                       (obtenerVia (parViasElemento (ubicacionEquipo confi)) /=
                                                    obtenerVia(via1,via2)))
                        lcs

cambiaUnion2Sectores:: [(TipoConfiguracion)] -> Bool -> [(TipoConfiguracion)]
cambiaUnion2Sectores [] _ = []
cambiaUnion2Sectores ((eqpo,st,ubieq):xs) estadonew =
        if (eqpo/=AI) then
             if (estadonew == False)
             then
                       (eqpo,estadonew,ubieq) : cambiaUnion2Sectores xs estadonew
             else
                       (eqpo,estadonew,ubieq) :
                       xs
        else
             (eqpo,st,ubieq) : cambiaUnion2Sectores xs estadonew

cambiaUnionSectores:: (Eq v) => [([TipoConfiguracion],v,v)] -> v -> v -> Bool
                                -> [([TipoConfiguracion],v,v)]
```

```
cambiaUnionSectores [] _ _ _= []
cambiaUnionSectores ((lconfis, va1,va2):xs) v1 v2 estadonew =
        if (v1==va1 && v2==va2) || (v1==va2 && v2==va1) then
            ((cambiaUnion2Sectores lconfis estadonew),va1,va2) :
                                    (cambiaUnionSectores xs v1 v2 estadonew)
        else
            (lconfis, va1,va2) : (cambiaUnionSectores xs v1 v2 estadonew)

rangoVertices :: (Ix v) => (Grafo v) -> (v,v)
rangoVertices g = bounds g

comunicaSectores :: (Ix v) => (Grafo v) -> v -> v -> (Grafo v)
comunicaSectores g v1 v2 = creaGrafo (rangoVertices g)
                                    (cambiaUnionSectores (aristasND g) v1 v2 True)

aislaSectores :: (Ix v) => (Grafo v) -> v -> v -> (Grafo v)
aislaSectores g v1 v2 = creaGrafo (rangoVertices g)
                                  (cambiaUnionSectores (aristasND g) v1 v2 False)

sectoresAccesibles:: (Ix v) => v -> Grafo v -> ((Grafo v) -> v -> [v]) -> [v] -> [v]
sectoresAccesibles i g fady lsc = rp [i] []
    where
      rp [] vis    = vis
      rp (c:cs) vis
          | c `elem` vis = rp cs vis
          | otherwise    = rp ((sinCaidos (fady g c) lsc) ++cs)
                              (vis++[c])

sectoresAccesiblesEE:: (Ix v) => v -> Grafo v -> [v] -> TipoRecorridoEE -> [v]
sectoresAccesiblesEE i g lsc (recor,cp) | (recor==RTRA && cp) = (sectoresAccesibles i g
adyacentesTrazado lsc)
                                        | (recor==RSEC && cp) = (sectoresAccesibles i g
                                        adyacentesSeccionador lsc)
                                        | (recor==RELE && cp) = (sectoresAccesibles i g
                                        adyacentesElectrico lsc)
                                        | (not cp) = filter (\n -> n /= i)
                                        (sectoresAccesiblesEE i g lsc (recor,True))
```

```
module EsquemaElectrico
    (
        conexionesEE,
        electrificacionEE,
        sectoresEnVp,
        crearEsquemaElectrico,
        esPosibleConexionElectrica,
        hayConexionElectrica,
        aislarSector,
        aislarSectorEE,
        esSectorAlimentadoSinEstado,
        esSectorAlimentadoConEstado,
        sectoresBienAlimentadosEE,
        sectoresNoCaidosBienAlimentadosEE,
        todosSectoresNoCaidosBienAlimentadosEE,
        esSectorBienAlimentadoConCaidaSinEstado,
        esSectorBienAlimentadoConCaidaConEstado,
        sectoresVpAlimentadosConCaidaSectorVs,
        puedeLlegarTren,
        fuentesDeSector,
        fuentesDeSectorConEstado,
        esSectorConUnicaFuente,
        esSectorConUnicaFuenteConEstado,
        todosSectoresConSoloUnTipoDeFase,
        todosSectoresConSoloUnTipoDeFaseEE,
        compruebaSeñalSeccionamiento,
        compruebaSeñalesSeccionamientoEE,
        cumpleDistanciaZnEE,
        cumpleDistanciaTodasZnEE,
        idSectorContiguoPkDado,
        compruebaFasesSectoresExtremosZn,
        pkSeccionamientoMasProximo,
        pksSeccionamientosMismaViaSentido,
        cumpleNormaDistanciaSeñalPkSeccionamiento,
    ) where


import Electrificacion
import GrafoEE
import Configuracion
import SectorElectrico
import SectoresVp
import Ubicacion
import Señal
import Data.Array


type TipoEsquemaElectrico v = (TipoSectoresVp v, TipoElectrificacion v, Grafo v)

crearEsquemaElectrico:: (Ix v) => TipoSectoresVp v -> TipoElectrificacion v -> Grafo v
                                                    -> TipoEsquemaElectrico v
crearEsquemaElectrico svp electr gee = (svp,electr,gee)

sectoresEnVp :: TipoEsquemaElectrico v -> TipoSectoresVp v
sectoresEnVp (svp, _,_) = svp

electrificacionEE:: TipoEsquemaElectrico v -> TipoElectrificacion v
electrificacionEE (_,electr, _) = electr

conexionesEE::  TipoEsquemaElectrico v ->   Grafo v
conexionesEE ( _,_, conex) = conex

-- Ver si sería posible conectar eléctricamente los dos sectores dados
esPosibleConexionElectrica:: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
esPosibleConexionElectrica s1 s2 (_,electr,gee) =
                              elem s2 (sectoresAccesiblesEE s1
                                       gee (sectoresCaidos electr) (RSEC, False))

-- Comprueba si con la configuración actual habría conexión eléctrica
-- entre los dos sectores dados
hayConexionElectrica:: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
```

```
hayConexionElectrica s1 s2 (_,electr,gee) =
                                elem s2 (sectoresAccesiblesEE s1
                                            gee (sectoresCaidos electr) (RELE, False))

aislarConAdysSeccionador:: (Ix v) => v -> [v] -> Grafo v -> Grafo v
aislarConAdysSeccionador _ [] ge = ge
aislarConAdysSeccionador s1 (x:xs) ge = aislarConAdysSeccionador s1 xs (aislaSectores ge s1 x)

-- AISLAR SECTOR INDICADO
-- Todo seccionador que comunica con sector adyacente pasará a posición de abierto
aislarSector:: (Ix v) => v -> Grafo v -> Grafo v
aislarSector s ge = aislarConAdysSeccionador s (adyacentesSeccionador ge s) ge

-- Operativa a realizar ante sector caido
aislarSectorEE:: (Ix v) => v -> TipoEsquemaElectrico v -> TipoEsquemaElectrico v
aislarSectorEE s (svp,(secta,alim,lf,lsc),gee) = (svp,
                                                  (secta,alim,lf, anadirCaido s lsc),
                                                   aislarSector s gee)

-- Comprobar que una lista de sectores dados están correctamente alimentados
-- Para ello Comprobar que todos tienen un posible camino eléctrico a un fuente
-- si en la lista suministrada hay un sector caido también devuelve falso
sectoresBienAlimentadosEE:: (Ix v) => TipoEsquemaElectrico v -> [v] -> Bool
sectoresBienAlimentadosEE _ [] = True
sectoresBienAlimentadosEE (svp,electr,gee) (x:xs)
          | (esSectorCaido x (sectoresCaidos electr)) = False
          | otherwise = if (esSectorFuente x (sectoresFuente electr)) then
                              (sectoresBienAlimentadosEE (svp,electr,gee) xs)
                        else
                               esSectorAlimentadoSinEstado x (svp,electr,gee) &&
                               (sectoresBienAlimentadosEE (svp,electr,gee) xs)

-- Comprobar que una lista de sectores (quitando los caidos) dados
-- están correctamente alimentados.
-- Para ello Comprobar que todos tienen un posible camino eléctrico a un fuente
sectoresNoCaidosBienAlimentadosEE:: (Ix v) => TipoEsquemaElectrico v -> [v] -> Bool
sectoresNoCaidosBienAlimentadosEE (svp,electr,gee) ls =
            sectoresBienAlimentadosEE (svp,electr,gee) (sinCaidos ls (sectoresCaidos electr))

-- COMPROBAR QUE TODOS LOS SECTORES DE UN ESQUEMA ELÉCTRICO QUE NO ESTÉN CAIDOS
-- ESTÁN CORRECTAMENTE ALIMENTADOS
todosSectoresNoCaidosBienAlimentadosEE:: (Ix v) => TipoEsquemaElectrico v -> Bool
todosSectoresNoCaidosBienAlimentadosEE esq  =
            sectoresNoCaidosBienAlimentadosEE esq (nodos (conexionesEE esq))

esSectorAlimentado:: (Ix v) => v -> TipoEsquemaElectrico v -> Bool -> Bool
esSectorAlimentado s (svp,electr,gee) estsec =
           (not (esSectorCaido s (sectoresCaidos electr))) &&
           ((esSectorFuente s (sectoresFuente electr)) ||
           (hayFuente (sectoresAccesiblesEE s gee (sectoresCaidos electr) (r, False)) electr))
                where r | estsec == True = RELE
                        | otherwise = RSEC

-- Comprueba si un sector está alimentado
-- (camino eléctrico a un fuente) (sin tener en cuenta estado)
esSectorAlimentadoSinEstado:: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
esSectorAlimentadoSinEstado s esq = esSectorAlimentado s esq False

-- Comprueba si un sector está alimentado
-- (camino eléctrico a un fuente) (teniendo en cuenta estado)
esSectorAlimentadoConEstado:: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
esSectorAlimentadoConEstado s esq = esSectorAlimentado s esq True

esSectorBienAlimentadoConCaida :: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool -> Bool
esSectorBienAlimentadoConCaida s1 s2 esq estsec =
            (esSectorFuente s1 (sectoresFuente (electrificacionEE esq))) ||
            (hayFuente (sectoresAccesiblesEE s1
                       geeact (sectoresCaidos electract) (r, False)) electract)
                  where {(svact, electract, geeact) = aislarSectorEE s2 esq
                                    ; r | estsec == True = RELE
                                        | otherwise = RSEC}
```

```
-- Si cae s2 comprobar que s1 sigue alimentado correctamente (sin tener en cuenta estado)
esSectorBienAlimentadoConCaidaSinEstado :: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
esSectorBienAlimentadoConCaidaSinEstado s1 s2 esq =
                                          esSectorBienAlimentadoConCaida s1 s2 esq False

esSectorBienAlimentadoConCaidaConEstado :: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
esSectorBienAlimentadoConCaidaConEstado s1 s2 esq =
                                          esSectorBienAlimentadoConCaida s1 s2 esq True

-- Aplicar a una lista de sectores que ante la caida de un sector
-- ninguno de los de los de la lista introducida cae en consecuencia
sectoresVpAlimentadosConCaidaSectorVs :: (Ix v) => [v] -> v -> TipoEsquemaElectrico v -> Bool
                                                                                    -> Bool
sectoresVpAlimentadosConCaidaSectorVs [] _ esq _ = True
sectoresVpAlimentadosConCaidaSectorVs (x:xs) svs esq estsec =
    bienAlimentadoS &&
   (sectoresVpAlimentadosConCaidaSectorVs xs svs esq estsec)
      where bienAlimentadoS | estsec == True = (esSectorBienAlimentadoConCaidaConEstado x svs
                                                                                      esq)
                            | otherwise = (esSectorBienAlimentadoConCaidaSinEstado x svs
                                                                                   esq)

sectoresVpAlimentadosConCaidaSectorVsSinEstado :: (Ix v) => [v] -> v -> TipoEsquemaElectrico v
                                                                                    -> Bool
sectoresVpAlimentadosConCaidaSectorVsSinEstado ls svs esq =
                                        sectoresVpAlimentadosConCaidaSectorVs ls svs esq False

sectoresVpAlimentadosConCaidaSectorVsConEstado :: (Ix v) => [v] -> v -> TipoEsquemaElectrico v
                                                                                    -> Bool
sectoresVpAlimentadosConCaidaSectorVsConEstado ls svs esq =
                                        sectoresVpAlimentadosConCaidaSectorVs ls svs esq True


pasaTrenAisladorSeccion:: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
pasaTrenAisladorSeccion s1 s2 esq =
          if (esSectorAlimentadoConEstado s1 esq) && (esSectorAlimentadoConEstado s2 esq) then
                                        True
          else
                                        False

pasaTrenSeccionador:: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
pasaTrenSeccionador s1 s2 esq | (esSectorAlimentadoConEstado s1 esq) &&
                                  (esSectorAlimentadoConEstado s2 esq) = True
                              | not (esSectorAlimentadoConEstado s1 esq) &&
                                not (esSectorAlimentadoConEstado s2 esq) = False
                              | otherwise = if estadocerrado then
                                                  True
                                            else
                                                  False
                                where estadocerrado = haySeccionadorCerradoEnConfiguraciones
                                                (configuracionSector s1 s2 (conexionesEE esq))

pasaTrenSectoresContiguosConfiguraciones:: (Ix v) => v -> v -> TipoEsquemaElectrico v
                                                      ->[TipoConfiguracion] -> Bool
pasaTrenSectoresContiguosConfiguraciones _ _ _ [] = False
pasaTrenSectoresContiguosConfiguraciones s1 s2 esq (x:xs) =
  if (hayAisladorSeccion (equipo x)) then
   (pasaTrenAisladorSeccion s1 s2 esq) ||
   pasaTrenSectoresContiguosConfiguraciones s1 s2 esq xs
  else
   (pasaTrenSeccionador s1 s2 esq) || pasaTrenSectoresContiguosConfiguraciones s1 s2 esq xs

pasaTrenSectoresContiguos:: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
pasaTrenSectoresContiguos s1 s2 esq = pasaTrenSectoresContiguosConfiguraciones s1 s2 esq
                                        (configuracionSector s1 s2 (conexionesEE esq))

-- De todos los sectores por lo que pasaría un recorrido eléctrico sin considerar estado
-- que le llegue al sector nos quedamos con los fuente
fuentesDeSector :: (Ix v) => v -> TipoEsquemaElectrico v -> [v]
fuentesDeSector s esq = (filter (\n ->
```

```
        (esSectorFuente n (sectoresFuente (electrificacionEE esq))))
          (sectoresAccesiblesEE (s) (conexionesEE esq) (sectoresCaidos(electrificacionEE esq))
                                (RSEC, True)))

-- De todos los sectores por lo que pasaría un recorrido eléctrico considerando estado
-- que le llegue al sector nos quedamos con los fuente
fuentesDeSectorConEstado :: (Ix v) => v -> TipoEsquemaElectrico v -> [v]
fuentesDeSectorConEstado s esq =
   (filter (\n -> (esSectorFuente n (sectoresFuente (electrificacionEE esq))))
                  (sectoresAccesiblesEE (s) (conexionesEE esq)
                                        (sectoresCaidos(electrificacionEE esq)) (RELE, True)))

-- Verifica que un sector está alimentado siempre por la misma fuente (sin estado)
esSectorConUnicaFuente :: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
esSectorConUnicaFuente s esq =
                  (mismaFase (fuentesDeSector s esq)) (sectoresFuente (electrificacionEE esq))

-- Verifica que un sector está alimentado siempre por la misma fuente (con estado)
esSectorConUnicaFuenteConEstado :: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
esSectorConUnicaFuenteConEstado s esq =  mismaFase (fuentesDeSectorConEstado s esq)
                                                   (sectoresFuente (electrificacionEE esq))

todosSectoresConSoloUnTipoDeFase :: (Ix v) => [v] -> TipoEsquemaElectrico v -> Bool
todosSectoresConSoloUnTipoDeFase [] _ = True
todosSectoresConSoloUnTipoDeFase (x:xs) esq = (esSectorConUnicaFuenteConEstado x esq) &&
                                              todosSectoresConSoloUnTipoDeFase xs esq

-- VERIFICA QUE NO HAY NINGUN SECTOR EN EL EE DE ALTERNA ALIMENTADO CON DOS FASES DISTINTAS
todosSectoresConSoloUnTipoDeFaseEE :: (Ix v) => TipoEsquemaElectrico v -> Bool
todosSectoresConSoloUnTipoDeFaseEE esq =
                                todosSectoresConSoloUnTipoDeFase (nodos(conexionesEE esq)) esq

-- Nos dice si coincide la fase de un sector dado con la de alguno de una lista dada
-- (sólo se miran los que estén alimentados).
-- En este caso sólo tiene sentido buscar fuentes considerando el estado de los seccionadores
coincideFase:: (Ix v) => v -> [v] -> TipoEsquemaElectrico v -> Bool
coincideFase _ [] _ = False
coincideFase s (x:xs) esq =
    if (esSectorAlimentadoSinEstado s esq) then
          (head (fuentesDeSectorConEstado s esq) == head (fuentesDeSectorConEstado x esq)) ||
                                                    (coincideFase s xs esq)
    else
           (coincideFase s xs esq)

-- Nos dice si coincide la fase de un sector dado con la de alguno de sus adyacentes
-- Se considera adyacencia Trazado ==> SectoresAccesiblesTrazado
coincideFaseContiguos :: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
coincideFaseContiguos s esq =
       coincideFase s
                  (sectoresAccesiblesEE s
                                        (conexionesEE esq)
                                         (sectoresCaidos(electrificacionEE esq))
                                         (RTRA, False))
                  esq


-- nodos adyacentes en cuanto a circulacion de trenes
adyacentesNodosCirculacion :: (Ix v) => TipoEsquemaElectrico v -> v  -> [v]
adyacentesNodosCirculacion esq v =
     map (\(e,a) -> a) ladys
        where ladys = filter (\(e1,a1) -> pasaTrenSectoresContiguos v a1 esq)
                             ((conexionesEE esq)!v)

sectoresAccesiblesCirculacion :: (Ix v) => v -> TipoEsquemaElectrico v -> [v]
sectoresAccesiblesCirculacion i esq = rp [i] []
    where
      rp [] vis    = vis
      rp (c:cs) vis
          | c `elem` vis = rp cs vis
          | otherwise    = rp ((sinCaidos (adyacentesNodosCirculacion esq c)
                                          (sectoresCaidos (electrificacionEE esq))) ++cs)
```

```
                                (vis++[c])

sectoresLlegaTrenCirculando:: (Ix v) => v -> TipoEsquemaElectrico v -> [v]
sectoresLlegaTrenCirculando sinicial esq = filter (\n -> n /= sinicial)
                                                  (sectoresAccesiblesCirculacion sinicial esq)

-- COMPROBAR QUE ANTE LA CAIDA DE UN SECTOR EL TREN PUEDE ATRAVESAR LA ESTACIÓN.
-- Comprueba si con la configuración actual un tren puede llegar
-- de un sector incial a otro final
puedeLlegarTren :: (Ix v) => v -> v -> TipoEsquemaElectrico v -> Bool
puedeLlegarTren s1 s2 esq = elem s2 (sectoresLlegaTrenCirculando s1 esq)

-- Saca el grupo de sectores alimentados por una fuente
grupoSectores:: (Ix v) => v -> TipoEsquemaElectrico v -> [v]
grupoSectores sf esq = sectoresAccesiblesEE sf (conexionesEE esq)
                                               (sectoresCaidos(electrificacionEE esq))
                                               (RELE, True)

encontradoSectorPorPkyVia:: [TipoConfiguracion] -> TipoPk -> TipoIdVia -> Bool
encontradoSectorPorPkyVia [] _ _ = False
encontradoSectorPorPkyVia ((e1,p1,(pk1,pv1)):xs) pkentr ventr =
          if (pk1 == pkentr) && (obtenerVia pv1 == ventr) then
                                True
          else
                                encontradoSectorPorPkyVia xs pkentr ventr

idSectorContiguoPkDado:: (Ix v) => [([TipoConfiguracion],v)] -> TipoPk -> TipoIdVia -> v
idSectorContiguoPkDado ((l,idsres):xs) pkentr viaentr =
          if encontradoSectorPorPkyVia l pkentr viaentr
          then
                                  idsres
          else

                  idSectorContiguoPkDado xs pkentr viaentr

-- VERIFICA QUE LOS SECTORES SITUADOS A AMBOS LADOS DE ZONA NEUTRA DE ALTERNA
-- TIENEN DISTINTA FASE
compruebaFasesSectoresExtremosZn:: (Ix v) =>  v -> TipoEsquemaElectrico v -> Bool
compruebaFasesSectoresExtremosZn idszn esq = not (mismaFase [s1,s2]
(sectoresFuente(electrificacionEE esq)))
    where {s1 = idSectorContiguoPkDado (adyacentesConfiguraciones (conexionesEE esq) idszn)
           (pkInicial (ubicacionSector(sectorElectricoxId
           (sectoresElectricos(electrificacionEE esq))idszn)))
           (obtenerVia(parViasInicial(ubicacionSector(sectorElectricoxId
           (sectoresElectricos(electrificacionEE
           esq))idszn))))
            ;s2 = idSectorContiguoPkDado (adyacentesConfiguraciones (conexionesEE esq) idszn)
            (pkFinal (ubicacionSector(sectorElectricoxId
            (sectoresElectricos(electrificacionEE esq))idszn)))
            (obtenerVia(parViasFinal(ubicacionSector(sectorElectricoxId
            (sectoresElectricos(electrificacionEE esq))idszn)))) }


-- ZONAS NEUTRAS COMPROBACIÓN DE LONGITUD
-- Comprueba que una zona neutra de un esquema eléctrico de alterna
-- cumple la norma de longitud mínima
cumpleDistanciaZnEE :: (Ix v) => v -> TipoEsquemaElectrico v -> Bool
cumpleDistanciaZnEE ids esq =
            cumpleDistanciaZn ((sectorElectricoxId (sectoresElectricos (electr)) (ids)))
                              (alimentacion(electr))
            where electr = electrificacionEE (esq)

cumpleDistanciaTodasZnEE :: (Ix v) => TipoEsquemaElectrico v -> Bool
cumpleDistanciaTodasZnEE esq = cumpleDistanciaTodasZn (electrificacionEE esq)

pksSeccionamientosMismaViaSentido :: (Ix v) => [(TipoConfiguracion,v)] -> TipoSeñal v
                                                                       -> [(TipoPk)]
pksSeccionamientosMismaViaSentido lcs señ =
  if (orientacionSeñal(señ) == PKD) then
    map (\((e,p,(pke,parve)),ids) -> pke) (filter(\(confi,ids) ->
                  (equipo confi ==SS) &&
```

```
                    (pkElemento (ubicacionEquipo confi) < pkElemento (ubicacionSeñal señ)) &&
                    (obtenerVia (parViasElemento (ubicacionEquipo confi)) ==
                                obtenerVia (parViasElemento (ubicacionSeñal señ)))) lcs)
  else
    map (\((e,p,(pke,parve)),ids) -> pke) (filter(\(confi,ids) -> (equipo confi ==SS) &&
                    (pkElemento (ubicacionEquipo confi) > pkElemento (ubicacionSeñal señ)) &&
                    (obtenerVia (parViasElemento (ubicacionEquipo confi)) ==
                                obtenerVia (parViasElemento (ubicacionSeñal señ)))) lcs)

pkSeccionamientoMasProximo :: [TipoPk] -> TipoOrientacion -> TipoPk
pkSeccionamientoMasProximo [] _ = -1
pkSeccionamientoMasProximo lpks orient | (orient == PKC) = maximum (lpks)
                                       | (orient == PKD) = minimum (lpks)

cumpleNormaDistanciaSeñalPkSeccionamiento:: TipoPk -> TipoPk -> TipoAlimentacion -> Bool
cumpleNormaDistanciaSeñalPkSeccionamiento _ (-1) _ = True
cumpleNormaDistanciaSeñalPkSeccionamiento pkseñ pksecc alim
                                       | (alim == CC) = ((abs (pkseñ - pksecc))*1000) > 160
cumpleNormaDistanciaSeñalPkSeccionamiento pkseñ pksecc alim
                                       | (alim == CA) = ((abs (pkseñ - pksecc))*1000) > 402


aplanarListaConfiguracionIdSector :: ([TipoConfiguracion],v) -> [(TipoConfiguracion,v)]
aplanarListaConfiguracionIdSector ([],v) = []
aplanarListaConfiguracionIdSector ((x:xs),v) = (x,v): aplanarListaConfiguracionIdSector (xs,v)

compruebaSeñalSeccionamiento :: (Ix v) => TipoSeñal v -> TipoEsquemaElectrico v -> Bool
compruebaSeñalSeccionamiento señ esq =
   cumpleNormaDistanciaSeñalPkSeccionamiento (pkElemento (ubicacionSeñal señ))
       pkseccmasprox (alimentacion (electrificacionEE esq))
         where pkseccmasprox = pkSeccionamientoMasProximo(pksSeccionamientosMismaViaSentido
                               (concat(map (\elto -> aplanarListaConfiguracionIdSector (elto))
                               (adyacentesConfiguraciones (conexionesEE esq)
                                                          (idSectorSeñal señ)))) señ )
                                (orientacionSeñal señ)

-- COMPRUEBA DISTANCIA DE SEÑAL DE DETENCIÓN
-- CON SECCIONAMIENTO DE LÁMINA DE AIRE MÁS PRÓXIMO EN SU VIA.
compruebaSeñalesSeccionamientoEE :: (Ix v) => [TipoSeñal v] -> TipoEsquemaElectrico v -> Bool
compruebaSeñalesSeccionamientoEE [] _ = True
compruebaSeñalesSeccionamientoEE (ps:xs) esq
           | (tipologiaSeñal ps == PARADA) = (compruebaSeñalSeccionamiento ps esq) &&
                                             (compruebaSeñalesSeccionamientoEE xs esq)
           | otherwise = (compruebaSeñalesSeccionamientoEE xs esq)
```