Übersicht

Prüft ob an Verbindungsstellen zwischen Linien- und Punktfeatures definierte Verbindungsregeln erfüllt sind. Es kann eine Liste von Regeln definiert werden, welche erlaubte Feature-Kombinationen an einer Verbindungsstelle definieren. Trifft keine der Regeln auf eine Verbindungsstelle zu, wird für diese Verbindungsstelle ein Fehler gemeldet.

Regeln beinhalten Auswahlkriterien für Features, sowie optional Bedingungen für die Anzahl von involvierten Features der jeweiligen Featureklassen (wiederum auch für Unter-Selektionen). Im Fall von Linienfeatures kann dabei auch auf die Richtung der Linie am Verbindungspunkt (eingehend, abgehend) Bezug genommen werden. Wird für eine Verbindungsstelle ein Satz von Regeln gefunden, welche von allen verbundenen Features erfüllt ist (indem die Features sowohl ihre jeweiligen Selektionskriterien, als auch die optionalen Bedingungen für die Anzahl von Features erfüllen), gilt die Verbindungsstelle als korrekt. Andernfalls wird ein Fehler gemeldet.

Als Verbindungsstellen gelten alle Linienendpunkte und Punkte aus der Liste der angegebenen Featureklassen. Für jede dieser Verbindungsstellen werden allfällige weitere an dieser Stelle anstossende Features identifiziert. Auf diese Menge der Features wird anschliessend die Prüfung der Verbindungsregeln angewandt.

Die Regeln werden somit auch auf nicht verknüpfte Linienendpunkte, nicht verknüpfte Punkte, und Linienverknüpfungen ohne verknüpfte Punktobjekte angewandt. Anhand der Definition von Bedingungen zur *Anzahl* verknüpfter Objekte kann an solchen Stellen beispielsweise erkannt werden, dass eine erforderliche Verknüpfung mit anderen Features *fehlt*.

Parameter

Parameter	Typ	Beschreibung
`featureClasses`	`IFeatureClass[]`	Liste von Linien- und Punkt-Featureklassen, an deren Verknüpfungsstellen die Verbindungsregeln überprüft werden sollen.
`Rules`	`String[]`	Liste der Regeln für die einzelnen Featureklassen. Diese Liste ist strukturiert in Gruppen (Tupel) mit je einer Regel pro Featureklasse in `featureClasses`. Im Fall von drei Featureklassen **lines1**, **lines2**, **junctions** und drei Gruppen sieht die Regelliste somit wie folgt aus: 1. Regel für **lines1** in Regelgruppe **1** 2. Regel für **lines2** in Regelgruppe **1** 3. Regel für **junctions** in Regelgruppe **1** 4. Regel für **lines1** in Regelgruppe **2** 5. Regel für **lines2** in Regelgruppe **2** 6. Regel für **junctions** in Regelgruppe **2** 7. Regel für **lines1** in Regelgruppe **3** 8. Regel für **lines2** in Regelgruppe **3**

		9. Regel für **junctions** in Regelgruppe **3** Somit muss die Anzahl einzelner Regeln in dieser Liste einem Vielfachen der Anzahl der in `featureClasses` spezifizierten Featureklassen entsprechen. Dieses Vielfache entspricht der Anzahl der zu verwendenden Gruppen (>= 1). Für jede Verbindungsstelle werden einzelnen Regelgruppen in der Reihenfolge der Liste geprüft. Sind alle Regeln in einer Gruppe erfüllt, gilt die Regelgruppe als erfüllt und die Verbindungsstelle wird als korrekt angesehen (ohne weitere Prüfung der nachfolgenden Regelgruppen). Sind in einer Regelgruppe nicht alle Regeln erfüllt, wird die nachfolgende Regelgruppe geprüft. Wenn *keine* der Regelgruppen erfüllt ist wird für die Verbindungsstelle ein Fehler gemeldet.

Struktur einer Regel

Eine Regel für eine einzelne Featureklasse innerhalb einer Regelgruppe weist die folgenden Elemente auf:

```
<Auswahlbedingung>{;<Variable>:<Zählbedingung>}*{;<Variablen-Prüfbedingung>}
```

Neben einer erforderlichen `<Auswahlbedingung>`, welche die Menge der Features auf welche die Regel zutrifft einschränkt, können optional ein oder mehrere Ausdrücke für die Zuweisung des Resultats einer Zählbedingung zu benannten Variablen (`<Variable>:<Zählbedingung>`) verwendet werden. Einer solchen Variablen wird dabei die *Anzahl der Features* zugewiesen, für welche die Zählbedingung erfüllt ist. Falls Variablen in dieser Art deklariert werden, können die Variablenwerte aller Einzel-Regeln der Regelgruppe in einer `<Variablen-Prüfbedingung>` geprüft werden. Diese Prüfbedingung kann bei einer beliebigen Regel der Regelgruppe angehängt werden. Die einzelnen Bestandteile einer Regel werden dazu mit Strichpunkt voneinander getrennt.

Nur wenn sowohl die Auswahlbedingungen der einzelnen Featureklassen, als auch die Variablen-Prüfbedingung (sofern definiert) erfüllt sind, gilt die Regelgruppe als erfüllt.

Im Folgenden sind die einzelnen Bestandteile einer Regel beschrieben.

Auswahlbedingungen

Mit einer Auswahlbedingung kann mit Attributkriterien eingeschränkt werden, auf welche Features der entsprechenden Featureklasse die Regel zutrifft. Dazu können freie SQL-WHERE-Bedingungen verwendet werden.

Beispiel:

```
TYP IN (2,4,6) AND STATUS = 1
```

→ die Regel trifft nur zu, falls an der Verknüpfungsstelle aus der fraglichen Featureklasse ausschliesslich Features mit Typ 2, 4 oder 6 und Status 1 verknüpft sind.

Alternativ zu einer WHERE-Bedingung können die folgenden Spezialwerte verwendet werden:

Spezialwert	Bedeutung
`true`	Die Regel ist bezüglich der fraglichen Featureklasse **immer erfüllt**. Dies gilt auch wenn keine Features aus dieser Klasse an der Verknüpfungsstelle vorkommen.
`false`	Die Regel ist **nie erfüllt**, *sofern es an der Verknüpfungsstelle Features aus der fraglichen Featureklasse hat*. Die Regel ist nur erfüllt, wenn keine Features aus der fraglichen Klasse an der Verknüpfungsstelle vorkommen.
`null`	Synonym zu `true`

Bei Linien-Featureklassen steht für die Nutzung in der WHERE-Bedingung ein berechnetes Attribut mit Namen `_StartsIn` zur Verfügung. Dieses Feld hat den Wert `true` wenn die verknüpfte Linie an der Verknüpfungsstelle ihren Startpunkt hat, und `false` wenn dort der Endpunkt der Linie liegt. Falls eine Linie an der Verknüpfungsstelle sowohl beginnt als auch endet, wird sie als zwei Linien gezählt, einmal mit `_StartsIn = true` und einmal mit `_StartsIn = false`.

Mit diesem Feld kann eine Regel somit einfach auf nur eingehende oder nur abgehende Linien angewandt werden.

Variablenzuweisungen

Variablenzuweisungen bestehen aus einem Variablennamen, einem Doppelpunkt als Zuweisungssymbol, sowie einem SQL-WHERE-Ausdruck. Dabei wird der Variablen die Anzahl der Features der fraglichen Klasse, für welche der WHERE-Ausdruck erfüllt ist, zugewiesen. Auch hier können die oben genannten Spezialwerte (true, false, null) verwendet werden.

Beispiele:

```
count1:true
```

→ die Variable `count1` erhält die Anzahl aller verknüpften Features der fraglichen Featureklasse.

```
count2: TYP IN (2,4)
```

→ die Variable `count2` erhält die Anzahl der verknüpften Features der fraglichen Featureklasse, für welche das Attribut TYP die Werte 2 oder 4 hat.

Analog zur Auswahlbedingung kann im Fall von Linienfeatures auch in der Variablenzuweisung das berechnete Attribut `_StartsIn` genutzt werden. Somit können zum Beispiel Variablen mit der Anzahl abgehender oder eingehender Linien (ggf. mit Einschränkungen auf weiteren Attributen) zugewiesen werden, welche anschliessend in der Variablen-Prüfbedingung geprüft werden können (s. Beispiel unten).

Variablen-Prüfbedingung

Die Variablenzuweisungen an sich hat noch keinen Einfluss auf die Erfüllung einer Regel. Die Prüfung, ob Variablenwerte gültig sind, erfolgt in einer separaten Prüfbedingung. In dieser Bedingung können alle Variablen, welche in den einzelnen Regeln der Regelgruppe definiert wurden, gemeinsam ausgewertet werden. In einer Regelgruppe kann es maximal *eine* solche Prüfbedingung geben, welche an einer *beliebigen* Regel der Gruppe angehängt werden kann. Die Prüfbedingung sollte alle in der Regelgruppe definierten Variablen nutzen. Umgekehrt können Zuweisungen zu Variablen, welche in der Prüfbedingung nicht genutzt werden, gelöscht werden, da sie keinerlei Wirkung auf die Auswertung der Regel haben.

Beispiele:

Regel für Featureklasse A: `TYP IN (2,4,6);countA_2_4:TYP IN (2,4)`

Regel für Featureklasse B: `true;countB:true;`**`countA_2_4 <= 2 AND countB = 1`**

→ Die Regel ist erfüllt, sofern **maximal 2** Features aus Featureklasse A mit TYP 2 oder 4 mit **genau einem** Feature aus Featureklasse B verbunden ist. Aufgrund der Auswahlbedingung für Featureklasse A dürfen aus Featureklasse A nur Features mit den Typen 2, 4 und 6 an der Verknüpfung beteiligt sein. Für Features mit TYP 2 und 4 gilt die zusätzliche Beschränkung auf die Anzahl, für Features mit TYP 6 ist dies nicht der Fall.

Beispiel für die Verwendung von `_StartsIn` zur Prüfung der Anzahl abfliessender Kanten:

```
true;outCount:(_StartsIn=true AND FLIESSRICHTUNG IN (0,1)) OR
(_StartsIn=false AND FLIESSRICHTUNG=2);outCount=1
```

→ Der Variablen `outCount` wird die Anzahl der abfliessenden Kanten an der Verknüpfungsstelle zugewiesen. In der Prüfbedingung wird anschliessend geprüft, ob an jeder Verknüpfungsstelle exakt eine abfliessende Kante vorliegt. Das Beispiel zeigt auch, wie aufgrund eines Attributs (`FLIESSRICHTUNG`) die Fliessrichtung entgegen der Digitalisierrichtung der Linie interpretiert werden kann (wobei die Werte 0 und 1 als „Fliessrichtung = Digitalisierrichtung“ und der Wert 2 als „Fliessrichtung = entgegen der Digitialisierrichtung“ interpretiert wird.
Die oben aufgeführte Bedingung entspricht der Logik des Tests `QaFlowLogic`. Im Rahmen von `QaLineConnection` kann diese Prüfung weit flexibler erfolgen, beispielweise mit unterschiedlicher Maximal/Minimalanzahl für Ein-/Abflüsse unterschiedlicher Typen, Kombination mit weiteren Zähl- und Auswahlregeln etc.

Bemerkungen

- Es empfiehlt sich, das gesamte Regelwerk für ein Linien-/Punktnetzwerk auf mehrere, fokussierte `QaLineConnection`-Tests aufzuteilen, da sonst die Anzahl der zu definierenden Regeln und Regelgruppen sehr gross und entsprechend unübersichtlich werden kann. Bewährt hat sich beispielweise, mit einzelnen `QaLineConnection`-Tests pro Punktfeatureklasse die Verknüpfungsbedingungen zwischen dieser Punktfeatureklasse und den Linienfeatureklassen zu prüfen. Oft gilt die Bedingung, dass nicht mehrere Punkte aufeinander liegen sollen (im Fall von Geometric Network-Datasets in der Geodatabase gilt diese Regel generell). Deshalb können jeweils die Verknüpfungen zwischen Punkten einer einzelnen Featureklasse mit den Linienfeatureklassen getrennt betrachtet werden. Die Prüfung auf nicht zusammenfallende Punkte kann in einem separaten Test erfolgen (beispielsweise `QaInteriorIntersectsSelf(3)` definiert gegen alle fraglichen Punktfeatureklassen).
- Die in den Regeln verwendeten SQL-WHERE-Ausdrücke werden nicht von der darunterliegenden Datenbank, sondern Client-seitig durch das QA-Framework interpretiert. Beschreibungen der dabei zulässigen Syntax findet sich hier:
  - http://www.csharp-examples.net/dataview-rowfilter/
  - http://msdn.microsoft.com/en-us/library/system.data.datacolumn.expression.aspx
- Die Featureklassen können, müssen aber nicht in einem Geometric Network enthalten sein.

- Eine definierte Spatial Reference ist erforderlich, da die XY-Toleranz ansonsten nicht ermittelt werden kann.
- Alle beteiligten Featureklassen müssen dieselbe Spatial Reference aufweisen.
- Für alle beteiligten Featureklassen können Filter-Bedingungen angegeben werden, welche die Menge der im Test zu berücksichtigenden Features einschränken.

Beispiele

Beispiel 1:

Zu prüfende Bedingung: Punkte in P1 dürfen nie mit Linien in L2 verknüpft sein. Verknüpfungen von P1 und L1 sind erlaubt. Verknüpfungen von L1 und L2 sind ebenfalls erlaubt (sofern nicht ein Punkt aus P1 an derselben Stelle liegt).

Konfiguration

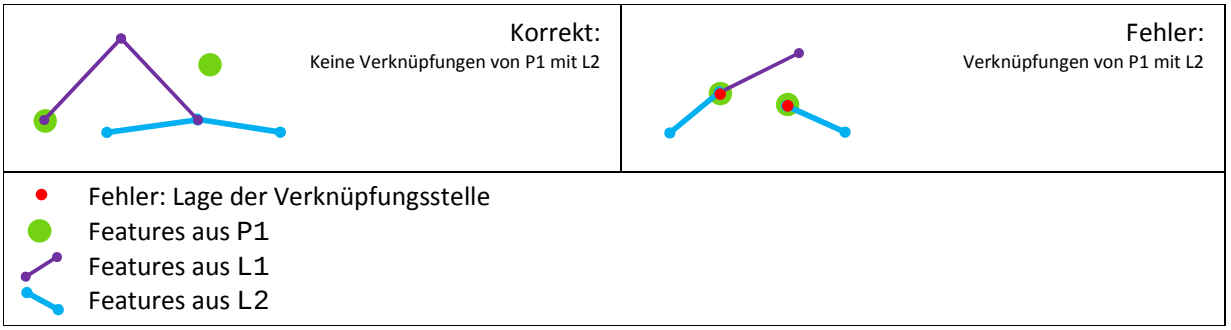
Featureklasse	P1	L1	L2
Regelgruppe 1	`true`	`true`	`false`
Regelgruppe 2	`false`	`true`	`true`

1. Falls an der Verknüpfungsstelle L2 **nicht** involviert ist
   → **korrekt**, Test abgeschlossen

Sonst (L2 ist also involviert):

2. Falls an der Verknüpfungsstelle P1 nicht involviert ist
   → **korrekt**, Test abgeschlossen

Sonst (P1 ist also involviert, und – da Regel 1 nicht erfüllt wurde – auch L2)
   → **Fehler**



Beispiel 2:

Zu prüfende Bedingung: Punkte in P1 dürfen nur mit Linien in L2 mit TYP 1 oder 2 verbunden sein. Verknüpfungen von P1 und L1 sind nicht erlaubt. Sonstige Verknüpfungen von L1 und L2 sind erlaubt.

Konfiguration

Featureklasse	P1	L1	L2
Regelgruppe 1	`true`	`false`	`TYP IN (1, 2)`
Regelgruppe 2	`false`	`true`	`true`

1. Falls an der Verknüpfungsstelle L1 nicht involviert ist, und alle verknüpften Features von L2 vom TYP 1 oder 2 sind (oder es keine verknüpften Features von L2 gibt)
   → **korrekt**, Test abgeschlossen

Sonst (entweder ist L1 involviert, oder ein L2-Feature mit anderem TYP als 1, 2):

2. Falls an der Verknüpfungsstelle P1 nicht involviert ist
   → **korrekt**, Test abgeschlossen

Sonst

→ **Fehler**

Beispiel 3:

Zu prüfende Bedingung: Bei Punkten aus P1 und P2 dürfen jeweils maximal zwei Linien aus L1 verknüpft sein. Punkte aus P1 und P2 dürfen nicht aufeinander liegen.

Konfiguration

Featureklasse	L1	P1	P2
Regelgruppe 1	`true;cL1:true;cL1<=2`	`true`	`false`
Regelgruppe 2	`true;cL1:true;cL1<=2`	`false`	`true`
Regelgruppe 3	`true`	`false`	`false`

1. Falls an der Verknüpfungsstelle P2 nicht involviert ist, und die Anzahl Linien aus L1 kleiner oder gleich 2 ist
   → **korrekt**, Test abgeschlossen

Sonst

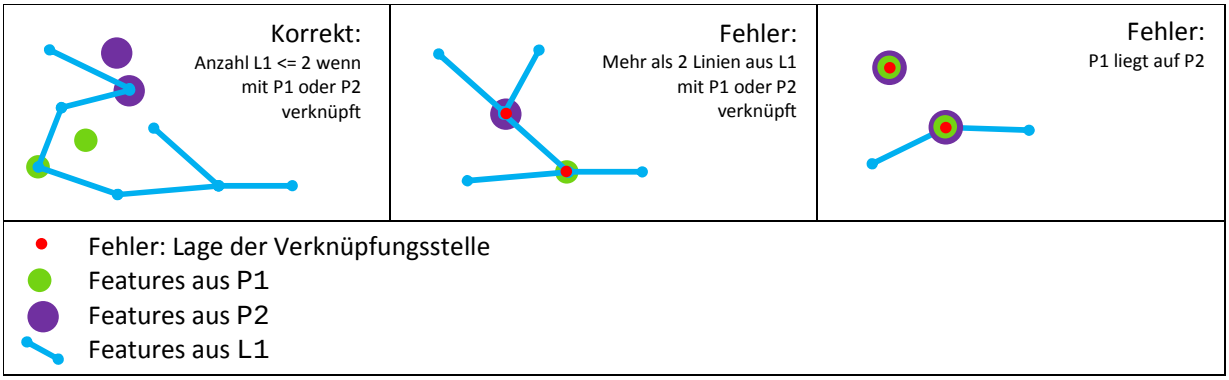
2. Falls an der Verknüpfungsstelle P1 nicht involviert ist, und die Anzahl Linien aus L1 kleiner oder gleich 2 ist
   → **korrekt**, Test abgeschlossen

Sonst

3. Falls an der Verknüpfungsstelle weder P1 noch P2 verknüpft sind (aber beliebige Features aus L1 verknüpft sein dürfen)
   → **korrekt**, Test abgeschlossen

Sonst

→ **Fehler**



Beispiel 4:

Zu prüfende Bedingung:

- Punkte in P1 mit Typ 2, 4 und 6 dürfen nur mit Linien in L2 mit Typ 1 oder 2 verknüpft sein.
- Punkte in P1 mit Typ 5 dürfen mit Linien in L2 mit Typ 1 oder 2, sowie mit allen Linien in L1 verknüpft sein.
- Punkte in P1 mit Typ 1 und 7 dürfen nur mit Linien in L2 mit Typ 1, 2 oder 4 verknüpft sein.
- Punkte in P1 mit Typ 3 dürfen nur mit Linien in L1 verknüpft sein.
- Verknüpfungen zwischen Linien in L1 und L2 ohne Verknüpfung mit P1 sind erlaubt.

Konfiguration

Featureklasse	P1	L1	L2
Regelgruppe 1	`TYP IN (2,4,6)`	`false`	`TYP IN (1,2)`
Regelgruppe 2	`TYP IN (5)`	`true`	`TYP IN (1,2)`
Regelgruppe 3	`TYP IN (1,7)`	`false`	`TYP IN (1,2,4)`
Regelgruppe 4	`TYP IN (3)`	`false`	`true`
Regelgruppe 5	`false`	`true`	`true`

Implementierung

Test Factory: `EsriDE.ProSuite.QA.TestFactories.QaLineConnection`

Assembly: `EsriDE.ProSuite.QA.TestFactories.dll`