Einführen von *equals()* auf ConflictReason Klasse.

Umsetzung:

- Methode generiert
- *equals()*-Methode auf *Set*-interface scheint bereits laut Doku (JavaDoc) ausreichend zu sein.

E:rgebnis:

- Vor Einführung:1688 CRs
- Nach Einführung: 1752 CRs

Wie kann das sein? Wie kann sich die Ergebniszahl dadurch erhöhen????

Ergebnis der Auswertung der erzeugten CRs:

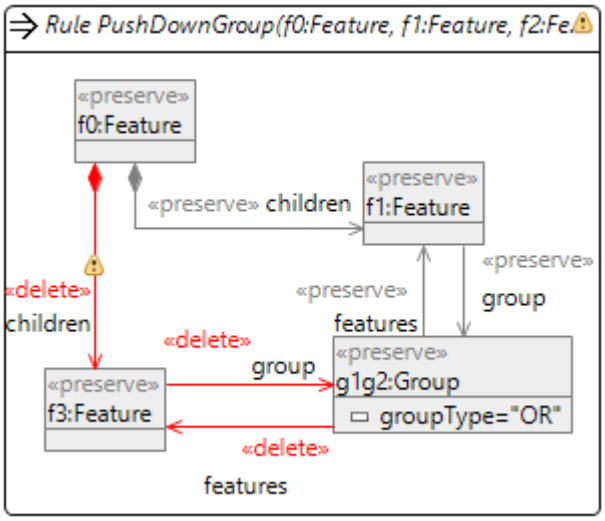
```
Anzahl der Einträge im 'amoutOfMCRs': 4
Die Analyse ergibt 16 Confl.Reason die aus 1 MinimalConflictReason bestehen
Die Analyse ergibt 120 Confl.Reason die aus 2 MinimalConflictReason bestehen
Die Analyse ergibt 640 Confl.Reason die aus 3 MinimalConflictReason bestehen
Die Analyse ergibt 976 Confl.Reason die aus 4 MinimalConflictReason bestehen
```

➔ Der ersten Eintrag (1-16) repräsentiert die MCRs selbst. → Gut!

➔ Dass es keine CRs gibt, die aus mehr als 4 MCRs besteht ist ebenfalls gut.

- Bei 4 MCRs sollte es sich immer um das löschende Muster handeln. Vermutung: „Eine valide Kombination der 4 MCRs muss auf Seite der 1.Regel immer zum gleichen Muster führen.“ Andernfalls kann die 1.Regel nicht ge-matcht werden. Für die zweite Regel fällt in dem Beispiel auf, dass der löschende „Ring“ der ersten Regel in der zweiten Regel mehrmals auftaucht. Mutmaßlich sechs mal. Dass kein vollständiger „Ring“ in der zweiten Regel vorhanden ist, ist durch die gemeinsamen MCRs und deren durch die erste Regel gegebenes Muster, ausgeschlossen.

Beispiel der halben PushDownGroup Regel



Die Anzahl der CRs und deren Zusammensetzung aus MCRs ist sehr plausibel

```
Anzahl der Einträge im 'amoutOfMCRs': 2
Die Analyse ergibt 4 Confl.Reason die aus 1 MinimalConflcitReason bestehen
Die Analyse ergibt 6 Confl.Reason die aus 2 MinimalConflcitReason bestehen
```

Die ess. CPA weist ebenso 10 de-use-Conflicts auf.

[update: durch die Fehlerbehebung sind es jetzt beim granularen Ansatz nur noch 6 (4+2) Ergebnisse. Siehe „Na toll, jetzt sind die Ergebnisse zu wenig.“]

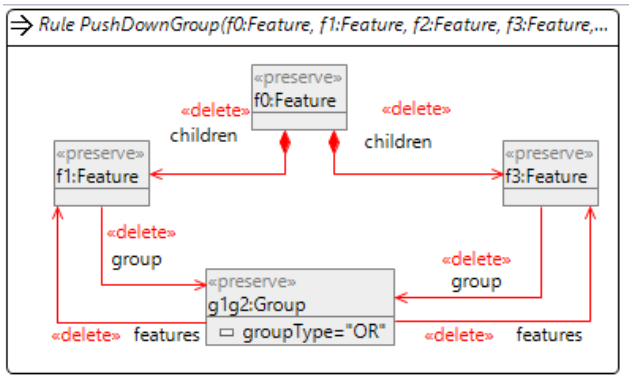
Kleine Zwischenerkenntnis: Ein Knoten aus Regel 1 darf nur einem Knoten aus Regel 2 zugeordnet sein über die MCRs bzw den CR.

D.H. setzt sich ein CR aus mehreren MCRs zusammen, sodass die MCRs einen Knoten einer Regel auf mehrere Knoten der anderen Regel abbilden, dann handelt es sich um kein gültiges CR!!!

Siehe Beispiel „HalfPushDownGroupCircle“: Diese Regel lässt sich mit vier MCRs nur zweimal auf sich selbst abbilden. Einmal mit einer komplett identischen Überlappung und einmal mit einer Spiegel an der vertikalen Schnittachse.

- ➔ Dies muss bei der Methode „findCommonNodesAndJoinToNewConflictReason(…)“ berücksichtigt werden!
- ➔ JUHU!!! Großer Durchbruch mit *(sameImageInRule1 ^ sameImageInRule2)* erreicht!

[HalfPushDownGroupCircle.henshin_diagram] Beispielregel:

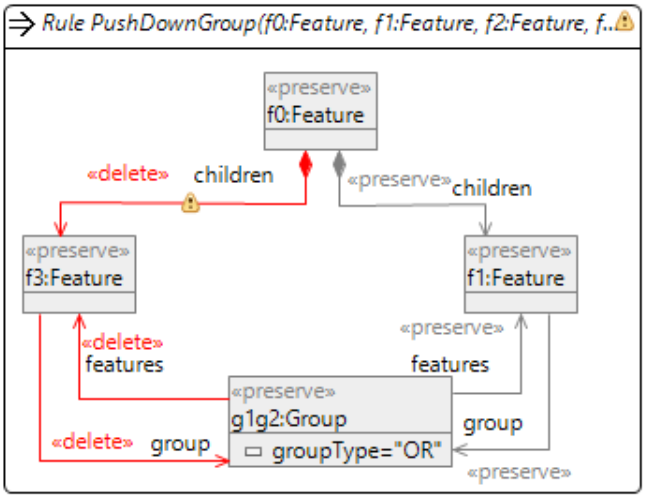


Na toll, jetzt sind die Ergebnisse zu wenig.

Mutmaßlich hängt das mit der Erkenntnis zusammen, dass im Spezialfall gegenläufiger zu löschender Referenzen zwischen den gleichen Knoten die Anzahl der MCR < Anzahl der CA ist.

Dies lässt sich auch im Beispiel der „halfPushDownGroup“ erkennen. Die vier ess.CPA Ergebnisse die nur eine der beiden gegenläufigen Referenzen berücksichtigen sind in vereinigter Form in den beiden Ergebnissen welche die gegenläufigen Referenzen beide berücksichtigen enthalten.

Regel **HalfPushDownGroup**:



Übersicht der **HalfPushDownGroup** Ergebnisse der ess. CPA: