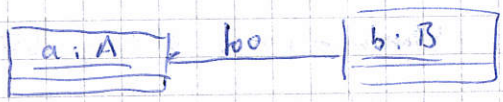
Unchangeable Graph Nodes

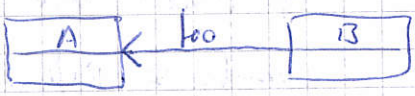
→ Idee:
- Alle Objekte, die in den Regeln oder für die Erreichbarkeitsanalyse genutzt, aber nie verändert werden, werden gekennzeichnet verwaltet.
- Diese Objekte sind dann im Graph enthalten, werden aber nicht kopiert, d.h. alle erreichbaren Graphen teilen sich dieselbe Menge von unveränderlich Knoten.

- Bedingungen für unveränderliche Knoten:

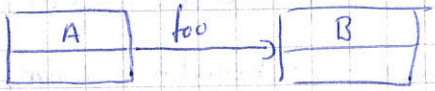
1. Sie sind in den initial Graph Objects enthalten.
2. Es gibt keine Regel, die ein Objekt der zugehörigen Klasse oder einer Oberklasse erzeugt oder löscht.
3. Existiert eine Link Variable, die erzeugt oder gelöscht wird, deren Source oder Target Node ein unchangeable object binden kann, dann muss die zugehörige Referenz unidirektional sein (d.h. kein Opposite besitzen), wobei die Referenz auf die Klasse des unchangeable objects zeigen muss: Bsp: Objekte von Typ A sind unchangeable

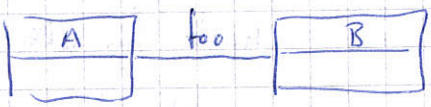


im Story Pattern

ist ok falls 

im Klassendiag

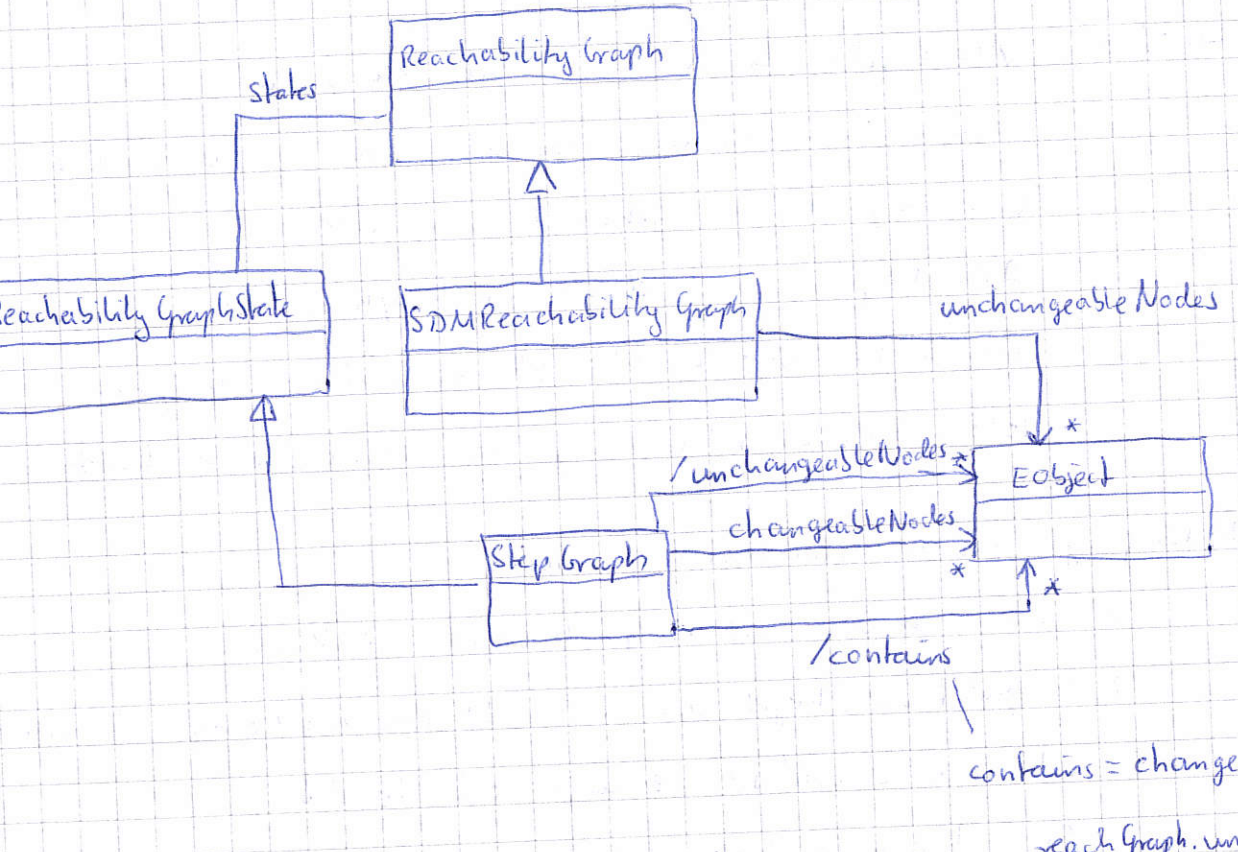
ist nicht ok falls 

oder 

im Klassendiag.

4. Die Referenzen, die nach 3. auf ein unchangeable object zeigen, dürfen keine Containments sein.

Meta-Modell



Anwendung:

copyState: - Kopiert nur die changeable Nodes
- Referenzen von changeable Nodes auf unchangeable Nodes werden so kopiert, dass anschließend ~~nur~~ Original und Kopie auf denselben unchangeable Node zeigen.

Isomorphism: - ~~Berechne~~ prüfe changeable Nodes auf Inhalts-gleichheit wie bisher auch
- prüfe bei Referenzen auf unchangeable Nodes, ob diese auf das gleiche Objekt zeigen (gleiche Referenzen)

Hashing: - berechne einmalig zu Beginn einen Level 0 Hash für alle unchangeable Nodes
- hashFunction berechnet dann Hashwerte für alle changeable Nodes
- referenziert ein changeable Node einen unchangeable Node, wird bei Level 1 und allen höheren Hashes jeweils der Level 0 Hash des unchangeable Nodes eingerechnet.

For Each Transformation:

- Es werden zwei Mengen von Classes unterschieden
  1. Classes, die zu unchangeable Nodes gehören (entweder sind alle Objekte einer Klasse unchangeable oder keine, da sonst nicht garantiert werden kann, dass die Regeln niemals ein unchangeable Object ändern)
  2. ~~Class~~ Alle sonstigen Classes.

- Objektvariables, die über eine Klasse aus Menge 1. gehypt sind, werden über derived Referenz „unchangeable Node" von step ausgehend gebunden → Object Variable Typ A
- Objektvariables, die über eine Klasse aus Menge 2. gehypt sind, werden über Referenz „changeable Nodes" von step ausgehend gebunden. → Object Variable Typ B.
- In den Activity Nodes nach dem Copy State - Aufruf werden alle ObjectVariables vom Typ B neu über die Map Index der GT Transition / SDM Transition neu gebunden (Objekte des kopierten Graphen finden aktuellen Match rausfinden)
- In den Activity Node nach dem Copy State - Aufruf werden alle Objektvariables vom Typ A als gebundene Objekte weiterverwendet.
- Wird in eine Regel ein Objekt erzeugt oder gelöscht ist es vom Typ B und wird dementsprechend aus der changeable Nodes Referenz von step entfernt.

Gesamtablauf Reachability Analysis: