

## Gruppe ATMS

### **1. ATMS als Datenstrukturen/Klassen in Java (S.152):**

#### Klasse ATMS-node

- Datum:
  - wird erstellt vom Problem Solver
  - die interne Struktur des Datums ist für die ATMS "egal", sie betrachtet nur das Gesamtdatum
- label:
  - wird von der ATMS berechnet
  - ist Menge von (konsistenten und minimalen) ATMS-environments
- justifications:
  - werden der ATMS vom Problem Solver übermittelt
  - soll als Collection von ATMS-environments dargestellt werden
- type:
  - benötigt 4 boolesche Methoden, die den ATMS-Knoten als premise, assumption, assumed node oder derived node klassifizieren
- Wahrheitsstatus:
  - ist von der Art: true, false, in, out (zudem werden jeweilige Methoden benötigt)
- Kontextbezogene Klassifikation:
  - ein Knoten ist:
    - "necessarily present"
    - "necessarily absent" (hinzufügen des Knotens würde falsum ableitbar machen, S.145 mitte)
    - "currently absent"
  - unklar bisher ist, ob dies überhaupt von uns benötigt wird
- consequents:
  - Liste von justifications
- contradictory:
  - boolescher Wert, ob Datum kontradiktorisch ist oder nicht

#### Klasse ATMS-justification

- consequent: bezeichnet den ATMS-Knoten
- antecedent: Liste von ATMS-Knoten (Vorgänger), kann auch eine leere Liste sein
- informant: nur für Problem Solver wichtig

#### ATMS-environment

- stellt eine Menge von assumptions dar, die einem Knoten letztendlich zugrunde liegen

#### ATMS-context:

- anscheinend ist keine explizite Repräsentation des ATMS-context erforderlich

## Bestandteile ATMS:

Die ATMS besteht aus:

- Menge von Knoten
- einer nogood Datenbank, in der die inkonsistenten ATMS-environments abgelegt sind
- einen ausgezeichneten (distinguished) falsum Knoten

Desweiteren verfügt die ATMS über 3 Basisoperationen, aus denen sich alle anderen Operationen zusammensetzen lassen:

- create ATMS-node
- add justification
- create assumption

Hierbei ist die bislang unbeantwortete Frage aufgetreten, ob aus einer Assumption ein Fakt (premise) werden kann, und wenn ja, dann wie. Wie würde sich diese Umwandlung auf entsprechende environments in der nogood-DB auswirken?

## Methoden der ATMS

- new node: +PS-datum
  - erstellt anhand des PS-datum einen neuen ATMS-Knoten mit leerer justifications-Liste
  - liefert Knoten mit Datum zurück
- new assumption: +PS-datum
  - fügt automatisch eine justification mit sich selbst ein
  - liefert Knoten zurück
- add justification (to node):
  - ist der Knoten gleich falsum, wird die nogood-DB gefüllt
  - update aller label muss vollzogen werden
  - update der consequents muss durchgeführt werden

Außerdem stellt sich die Frage, ob und wie eine Verwaltung im Problem Solver bestehen muss, die eine eindeutige Zuordnung von Datum und Knoten vornimmt, damit gleichbedeutende Daten demselben Knoten zugeordnet werden können.