Aufgabe 2: HRU-Beispiel Fernuni

Systemsicherheit

Übungsblatt 2
Stephan Beyer
Team 2

Systemsicherheit SS'08
Technische Universität Ilmenau

5. Mai 2008

Übungsblatt 2, Aufgabe 2

Ändern Sie das Autorisierungsschema des HRU-Modells des *Fernuni-Szenariums* aus der Vorlesung so ab, dass

1. jeder Kursteilnehmer seine Hausaufgaben *nur einmal* abgeben und die Musterlösung *nur einmal* abholen kann,
2. die *dynamische Aufnahme* und das *Löschen von Kursteilnehmern* während eines laufenden Kurses möglich wird.

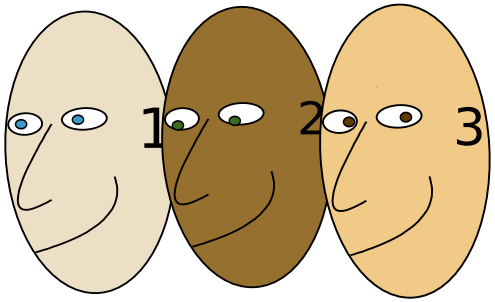
Zu welcher *Safety-Entscheidbarkeitsvariante* gehören die entstehenden Modelle?
Geben Sie exemplarisch für einen Studenten den für jeden Studenten durch die Zugriffsmatrix repräsentierten, aus drei Zuständen bestehenden Automaten und die entsprechenden Zustandsübergänge in Form eines *Zustandsgraphen* an.

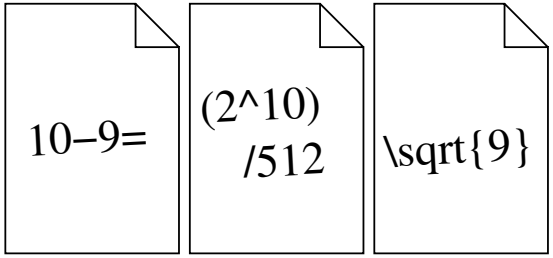
Vorlesungsbeispiel Fernuni

HRU-Modell (Q, E, δ, q_0)

Zustandsmenge $Q = S \times O \times M$

- Subjektmenge $S = \{s_1, s_2, s_3\}$ (Kursteilnehmer)
- Objektmenge $O = \{o_1, o_2, o_3\}$ (Hausaufgaben)
- Menge $M = \{m \mid m: \ S \times O \rightarrow \mathcal{P}(R)\}$ aller Zugriffsmatrizen über Rechtemenge $R = \{PutHomeworkRight, GetSampleRight\}$





M	o_1	o_2	o_3
s_1	...	...	...
s_2	...	...	...
s_3	...	...	...

Vorlesungsbeispiel Fernuni

HRU-Modell (Q, E, δ, q_0)

Eingabemenge $E = A \times X$

- anwendungsspezifische Operationen:
  $A = \{PutHomework, GetSample\}$
- beteiligte Subjekte und Objekte: $X = (S \cup O)^k$

Schreibweise:

- *PutHomework*(IN Kursteilnehmer, IN Hausaufgabe)
- *GetSample*(IN Kursteilnehmer, OUT Musterlösung)

Vorlesungsbeispiel Fernuni

HRU-Modell (Q, E, δ, q_0)

Zustandsübergangsfunktion $\delta : \; Q \times E \rightarrow Q$

Beispielsweise mathematisch formal, als Graph, oder...

Vorlesungsbeispiel Fernuni

. . . mittels Autorisierungsschema:

„Nach Abgabe der Hausaufgaben darf die Musterlösung gelesen werden.“

Regel für *PutHomeWork*

$\delta(q, (PutHomework, (s, o))) :=$

if $PutHomeworkRight \in M(s, o)$ **then**

enter $GetSampleRight$ **into** $M(s, o)$

„Nach Lesen der Musterlösung darf keine Hausaufgabe mehr abgegeben werden.“

Regel für *GetSample*

$\delta(q, (GetSample, (s, o))) :=$

if $GetSampleRight \in M(s, o)$ **then**

delete $PutHomeworkRight$ **from** $M(s, o)$

Vorlesungsbeispiel Fernuni

HRU-Modell (Q, E, δ, q_0)

Startzustand $q_0 \in Q$, formal:

$$
\begin{aligned}
q_0 \quad = \quad (\quad & \{s_1, s_2, s_3\}, \quad \{o_1, o_2, o_3\}, \\
& \{((s_1, o_1), \{PutHomeworkRight\}), ((s_1, o_2), \emptyset), ((s_1, o_3), \emptyset), \\
& ((s_2, o_1), \emptyset), ((s_2, o_2), \{PutHomeworkRight\}), ((s_2, o_3), \emptyset), \\
& ((s_3, o_1), \emptyset), ((s_3, o_2), \emptyset), ((s_3, o_3), \{PutHomeworkRight\}) \ \})
\end{aligned}
$$

oder als Matrix:

M	o_1	o_2	o_3
s_1	*PutHomeworkRight*		
s_2		*PutHomeworkRight*	
s_3			*PutHomeworkRight*

Anforderung 1

Jeder Kursteilnehmer kann seine Hausaufgaben *nur einmal* abgeben und die Musterlösung *nur einmal* abholen.

Regel für *PutHomeWork*

$\delta(q, (PutHomework, (s, o))) :=$

if $PutHomeworkRight \in M(s, o)$ **then**

enter $GetSampleRight$ **into** $M(s, o)$

delete $PutHomeworkRight$ **from** $M(s, o)$

Regel für *GetSample*

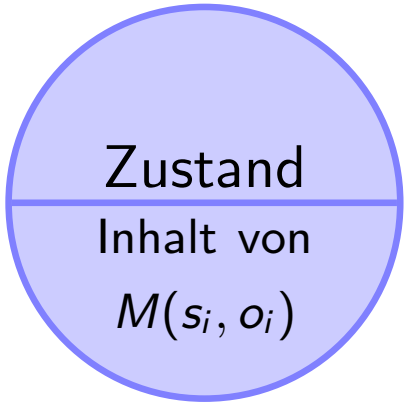
$\delta(q, (GetSample, (s, o))) :=$

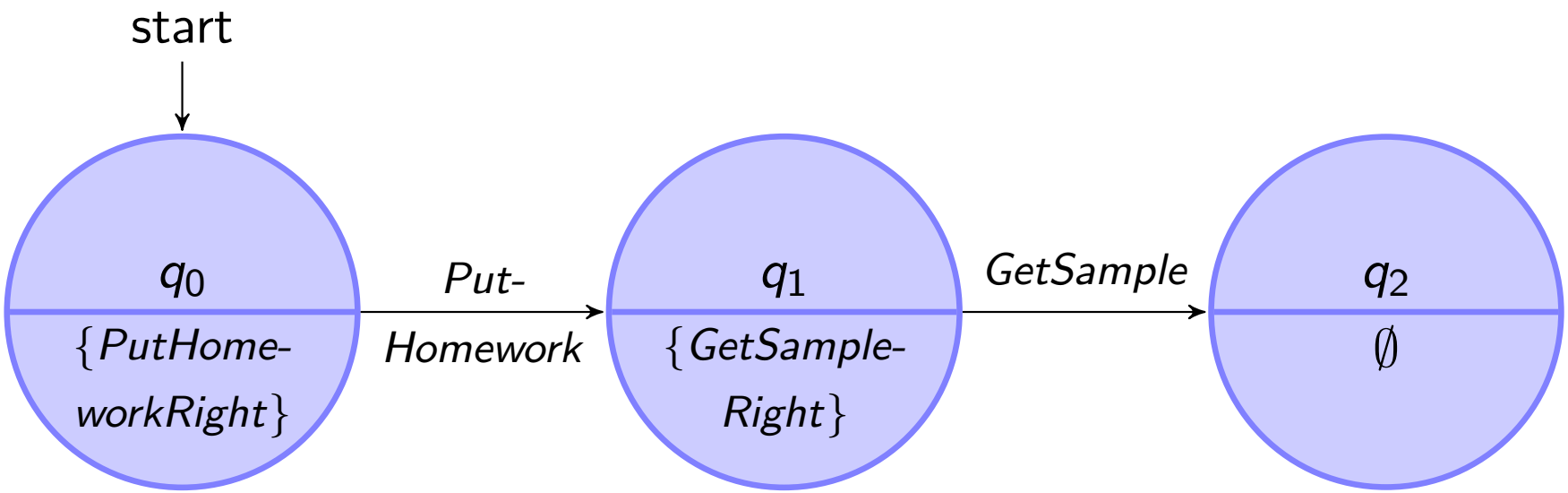
if $GetSampleRight \in M(s, o)$ **then**

delete $GetSampleRight$ **from** $M(s, o)$

Zustandsgraph

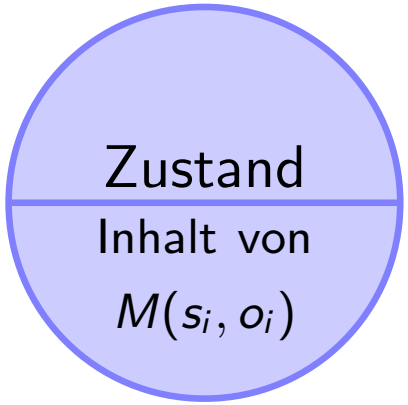
„Geben Sie exemplarisch für einen Studenten den für jeden Studenten durch die Zugriffsmatrix repräsentierten, aus drei Zuständen bestehenden Automaten und die entsprechenden Zustandsübergänge in Form eines Zustandsgraphen an.“

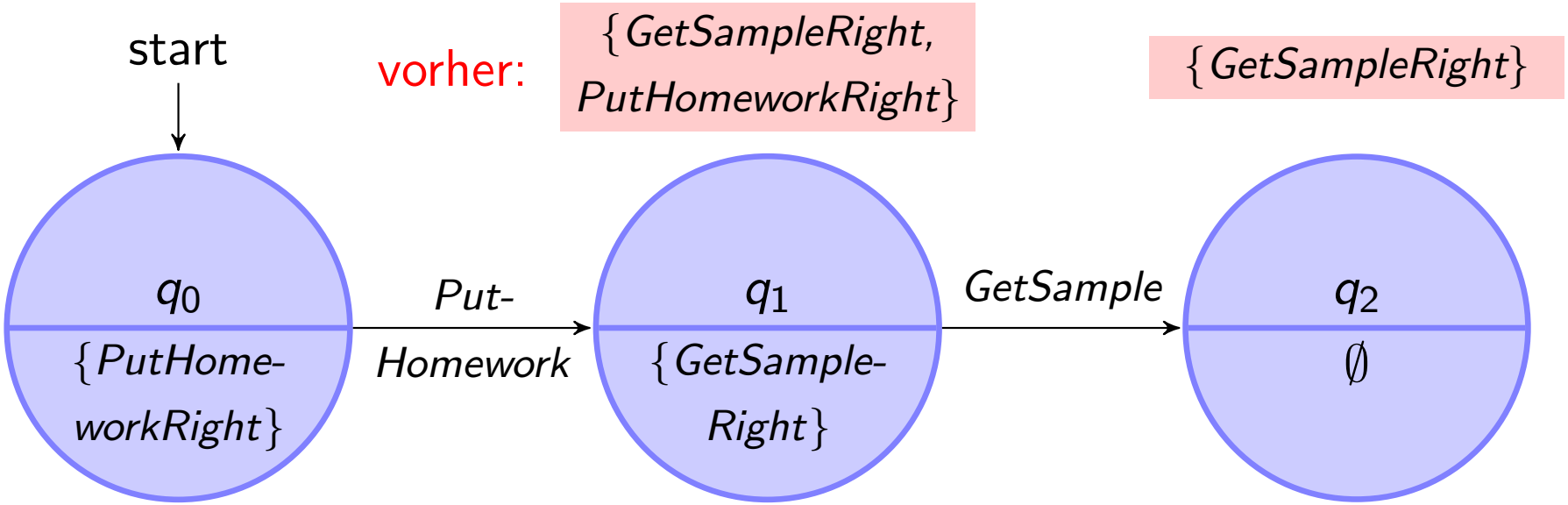




Zustandsgraph

„Geben Sie exemplarisch für einen Studenten den für jeden Studenten durch die Zugriffsmatrix repräsentierten, aus drei Zuständen bestehenden Automaten und die entsprechenden Zustandsübergänge in Form eines Zustandsgraphen an.“





HRU-Safety des Modells

allgemeine Klassifikation

Regel für *PutHomeWork*

$\delta(q, (PutHomework, (s, o))) :=$

if $PutHomeworkRight \in M(s, o)$ **then**

enter $GetSampleRight$ **into** $M(s, o)$

delete $PutHomeworkRight$ **from** $M(s, o)$

- monokonditional
- bioperational
- nicht monoton
- statisch

Regel für *GetSample*

$\delta(q, (GetSample, (s, o))) :=$

if $GetSampleRight \in M(s, o)$ **then**

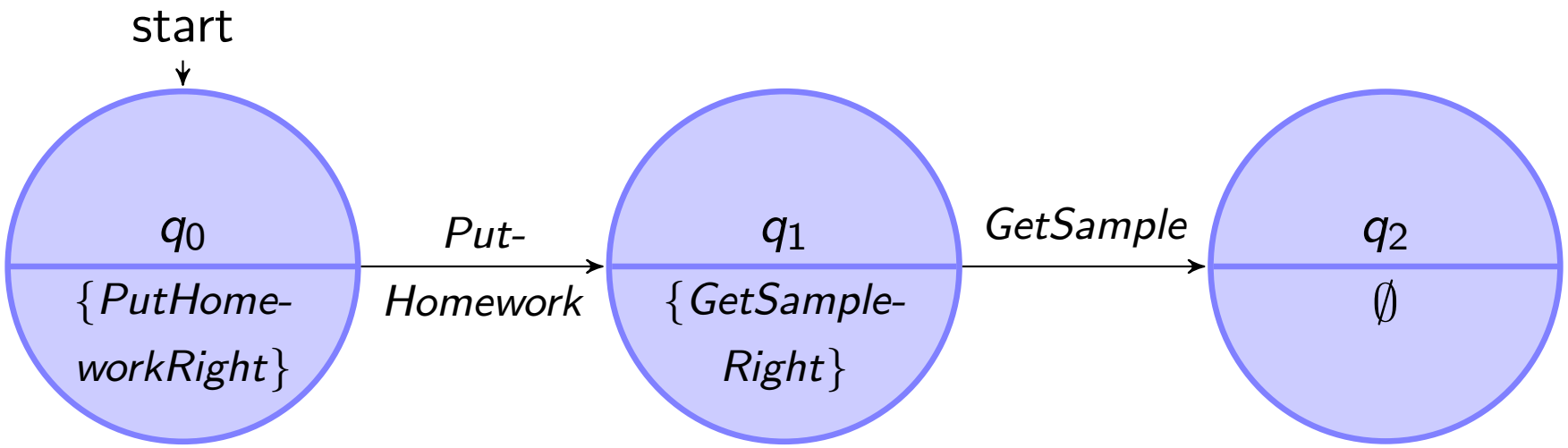
delete $GetSampleRight$ **from** $M(s, o)$

- monokonditional
- monooperational
- nicht monoton
- statisch

insgesamt: *statisch* $\Rightarrow$ entscheidbar (NP-vollständiges Problem)
auch: *endliche Subjektmenge* (3 Subjekte) $\Rightarrow$ entscheidbar

HRU-Safety des Modells

Betrachtung der Zustände



HRU-Safety in Bezug auf Recht...

- *PutHomeworkRight – in jedem Zustand HRU-safe*
  - wird nur ($q_0 \rightarrow q_1$) aus Matrix entfernt
- *GetSampleRight – HRU-safe*, erst sobald alle Kursteilnehmer Hausaufgabe abgegeben
  - q_0: kommt beim Übergang zu q_1 hinzu
  - q_1: wird nur noch entfernt

Anforderung 2

Dynamisches Aufnehmen und Löschen von Kursteilnehmern soll während laufendem Kurs möglich sein.

- *Wer* darf Kursteilnehmer aufnehmen und löschen?
  privilegierter Nutzer s_{Admin}

$\rightarrow$ $S := S \cup \{s_{\text{Admin}}\}$

- *Wen oder was* darf s_{Admin} aufnehmen und löschen? Kursteilnehmer

$\rightarrow$ $O := O \cup \{o_{\text{User}}\}$

- *Welche Rechte* braucht s_{Admin} zum aufnehmen und löschen?

$\rightarrow$ $R := R \cup \{AddUserRight, DelUserRight\}$

- *Was* darf s_{Admin}? Kursteilnehmer aufnehmen und löschen

$\rightarrow$ $A := A \cup \{AddUser, DelUser\}$
AddUser(IN priviligierter Nutzer, IN anzulegender Nutzer, IN Hausaufgabe)
DelUser(IN priviligierter Nutzer, IN zu löschender Nutzer, IN Hausaufgabe)

Anforderung 2

Autorisierungsschema

Regel für *AddUser*

$\delta(q, (AddUser, (s_A, s_B, o_B))) :=$

if $AddUserRight \in M(s_A, o_{\text{User}})$ **then**

create subject s_B
create object o_B
enter $PutHomeworkRight$ **into** $M(s_B, o_B)$

Regel für *DelUser*

$\delta(q, (DelUser, (s_A, s_B, o_B))) :=$

if $DelUserRight \in M(s_A, o_{\text{User}})$ **then**

destroy subject s_B
destroy object o_B

Anforderung 2

Initialisierung

Startzustand q_0 wird

M	o_{User}	o_1	o_2	o_3
s_{Admin}	*AddUserRight* *DelUserRight*			
s_1		*PutHwRight*		
s_2			*PutHwRight*	
s_3				*PutHwRight*

HRU-Safety des Modells

allgemeine Klassifikation

Regel für *AddUser*

$\delta(q, (AddUser, (s_A, s_B, o_B))) :=$

if $AddUserRight \in M(s_A, o_{\text{User}})$ **then**

create subject s_B
create object o_B
enter $PutHomeworkRight$ **into** $M(s_B, o_B)$

- monokonditional
- trioperational
- monoton
- dynamisch

Regel für *DelUser*

$\delta(q, (DelUser, (s_A, s_B, o_B))) :=$

if $DelUserRight \in M(s_A, o_{\text{User}})$ **then**

destroy subject s_B
destroy object o_B

- monokonditional
- bioperational
- nicht monoton
- dynamisch

Keine aus der Vorlesung bekannte entscheidbare Klassifikation anwendbar.

HRU-Safety des Modells

Betrachtung der Zustände

HRU-Safety in Bezug auf Recht...

- *PutHomeworkRight – in jedem Zustand HRU-safe*
  - wird nur in Matrixzellen eingefügt, die im jeweils vorherigen Zustand noch nicht vorhanden
- *GetSampleRight* – wie bisher
- *AddUserRight*, *DelUserRight – HRU-safe*
  - wird in keinem Autorisierungsschema hinzugefügt (kein **enter**)
  - nur s_{Admin} besitzt Recht seit q_0

Das war's!