UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

LEGO meets RV

Kolloquium zur Masterarbeit

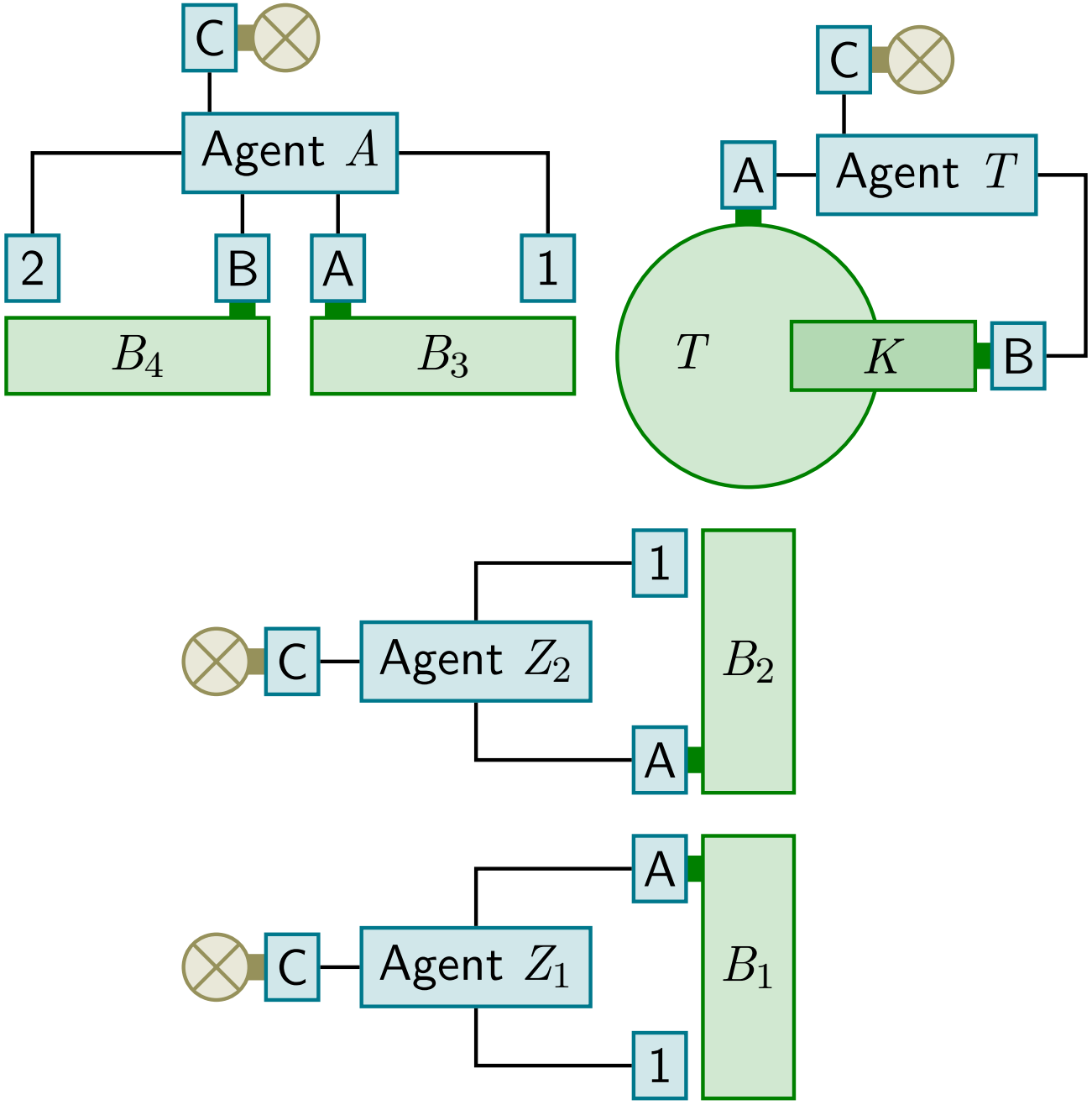
Verteilte Laufzeitverifikation auf eingebetteten Systemen

Malte Schmitz

7. August 2014

ausgegeben und betreut von
Prof. Dr. Martin Leucker


nxt
nxt
nxt


C
Agent A
2
B
A
1
B_4
B_3
C
A
Agent T
T
K
B
1
C
Agent Z_2
B_2
A
A
C
Agent Z_1
B_1
1

LEGO meets RV

Gliederung

Motivation

- Verteiltes System
- Logiken
- Monitorbarkeit

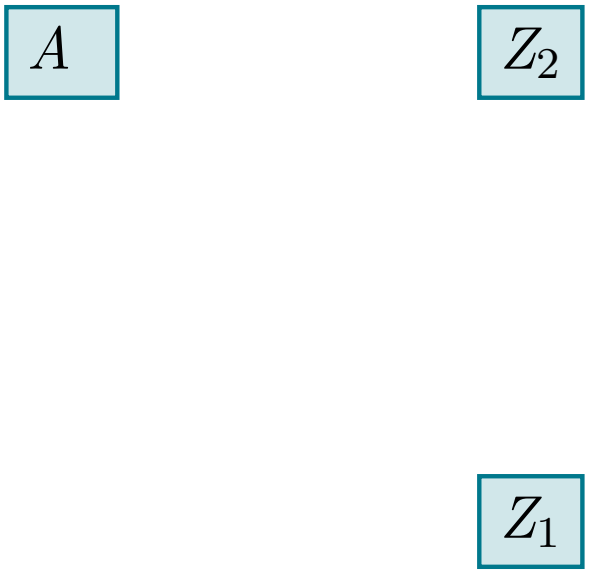
Distributed Temporal Logic

- Semantik
- Monitorgenerierung

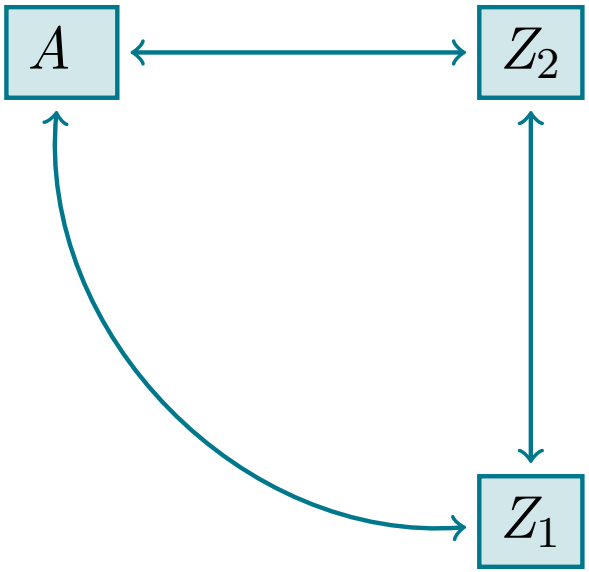
LEGO Mindstorms

- Architektur
- Fallstudie
- Benchmarks

Verteiltes, asynchrones Agentensystem

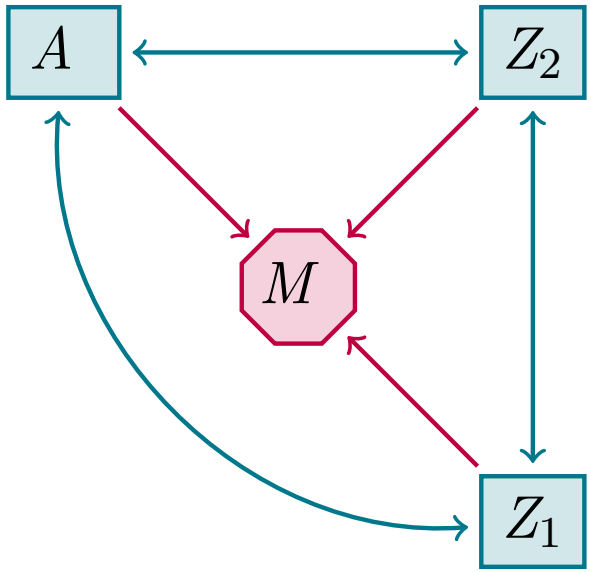


Verteiltes, asynchrones Agentensystem



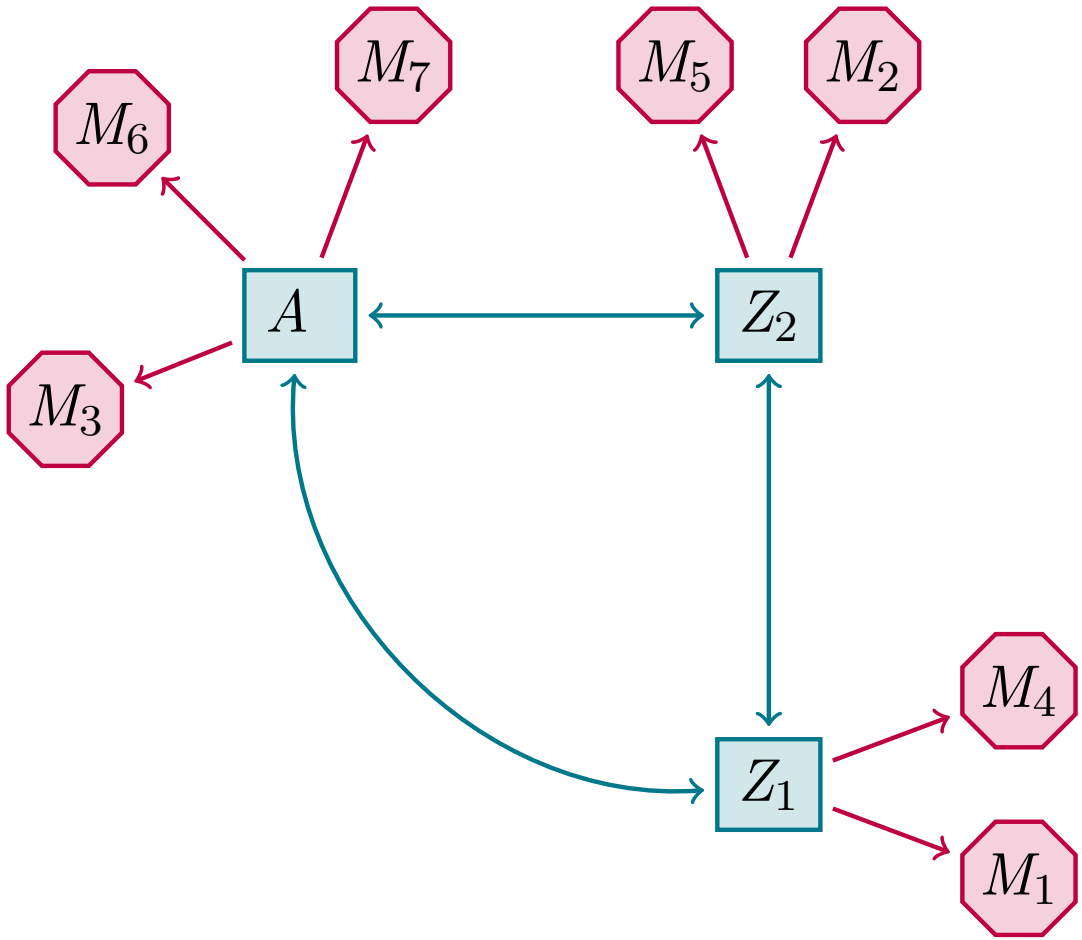
Verteiltes, asynchrones Agentensystem

mit einem zentralen Monitor



Verteiltes, asynchrones Agentensystem

mit ~~einem zentralen~~ vielen dezentralen Monitoren



Logiken

Prinzip

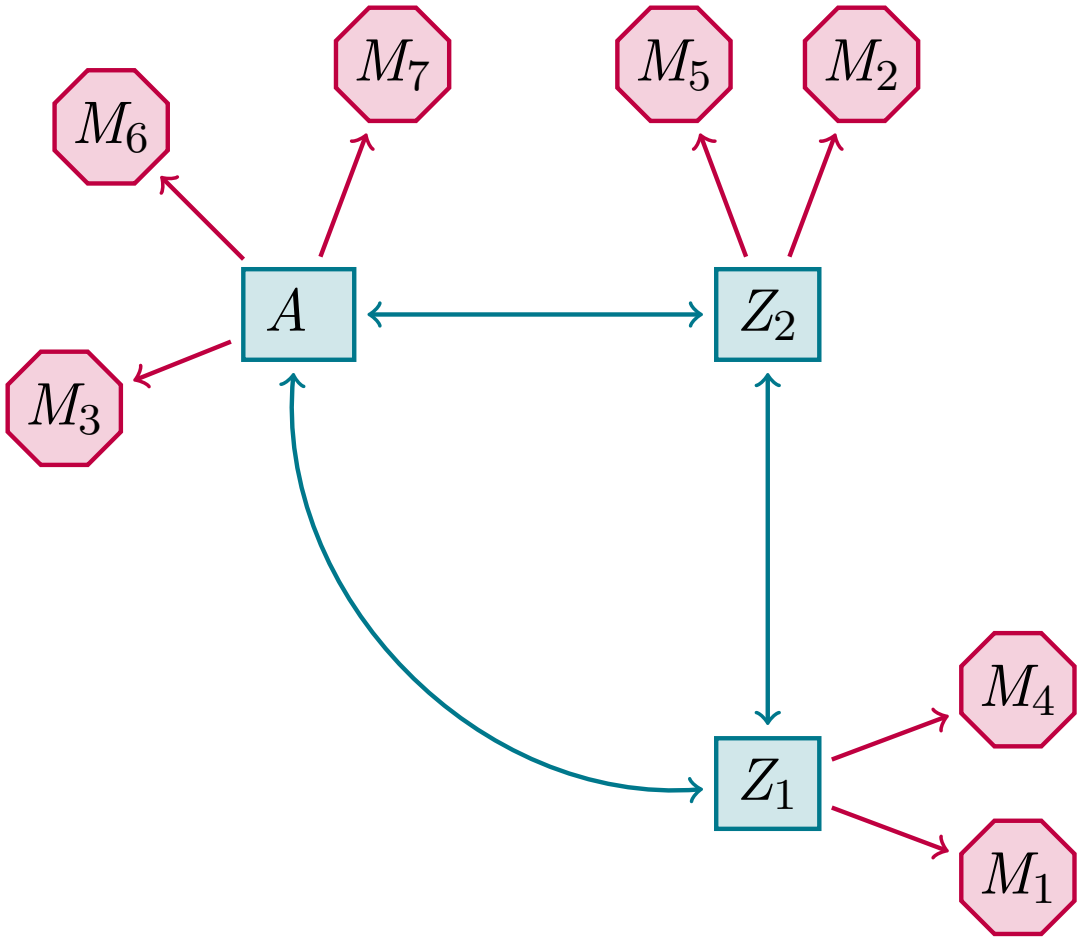
1. Teilformeln auf entfernten Agenten
   $\Longrightarrow$ @-Operator
2. Wissen über entfernte Agenten
   $\Longrightarrow$ letzte bekannte Position

past-time Distributed Temporal Logic (ptDTL)

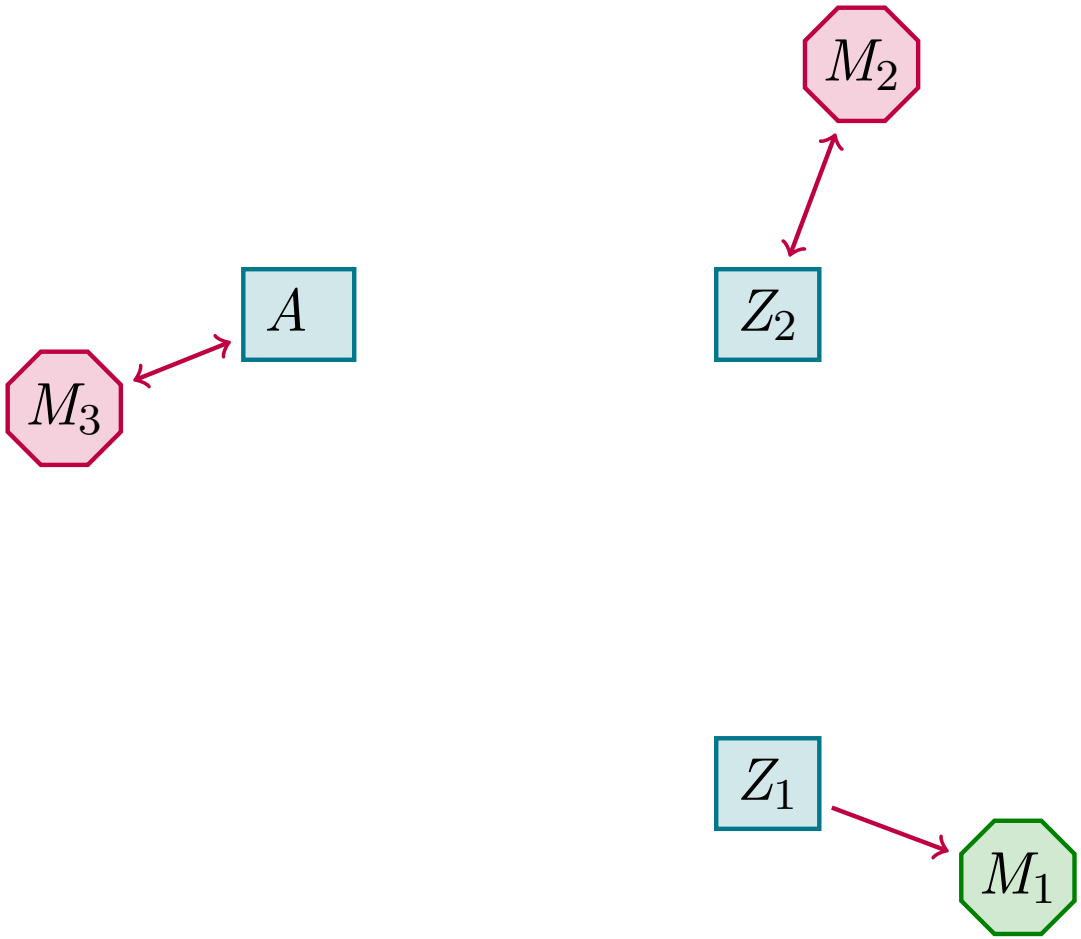
$$@_{Z_2}\left(\neg\, s_1\, \mathcal{S}\, @_{Z_1}\, s_1\right)$$

K. Sen, A. Vardhan, G. Agha, and G. Roşu.
Efficient Decentralized Monitoring of Safety in Distributed Systems.
26th ICSE, 2004.

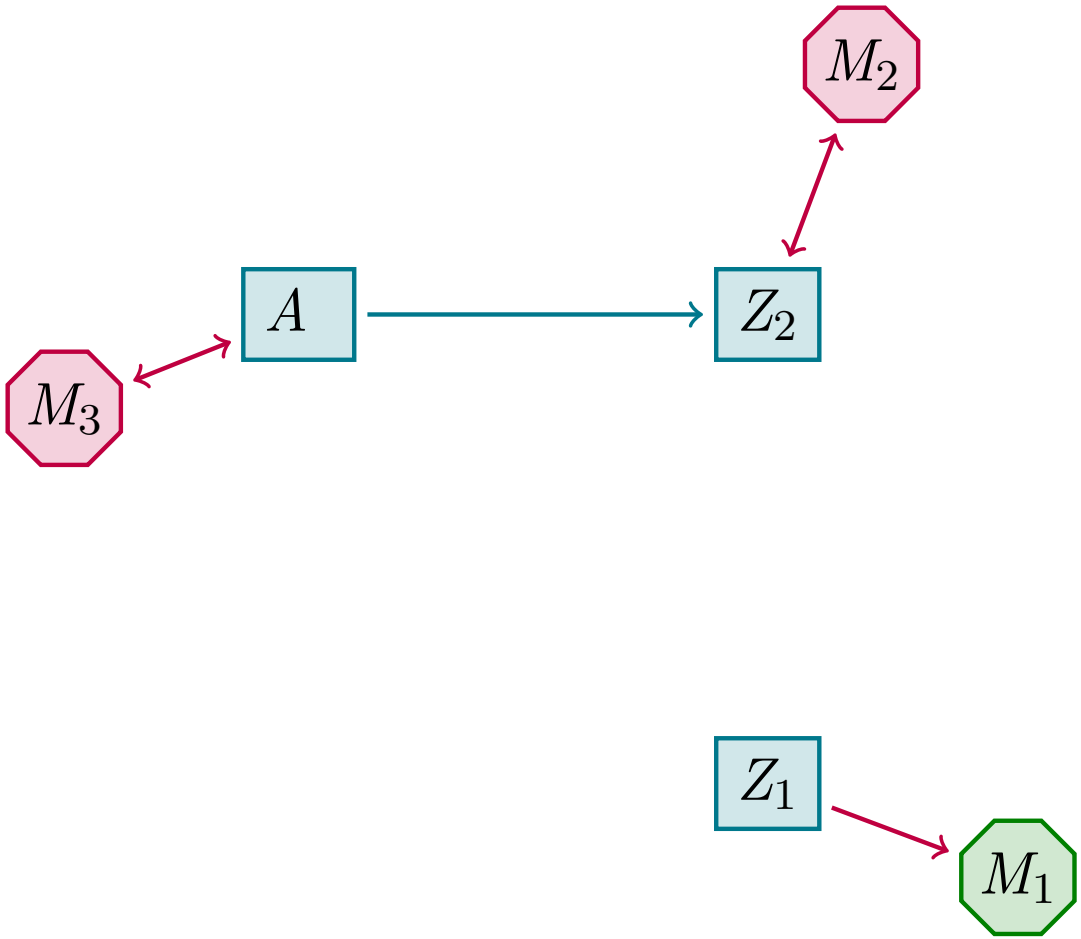
Letzte bekannte Position



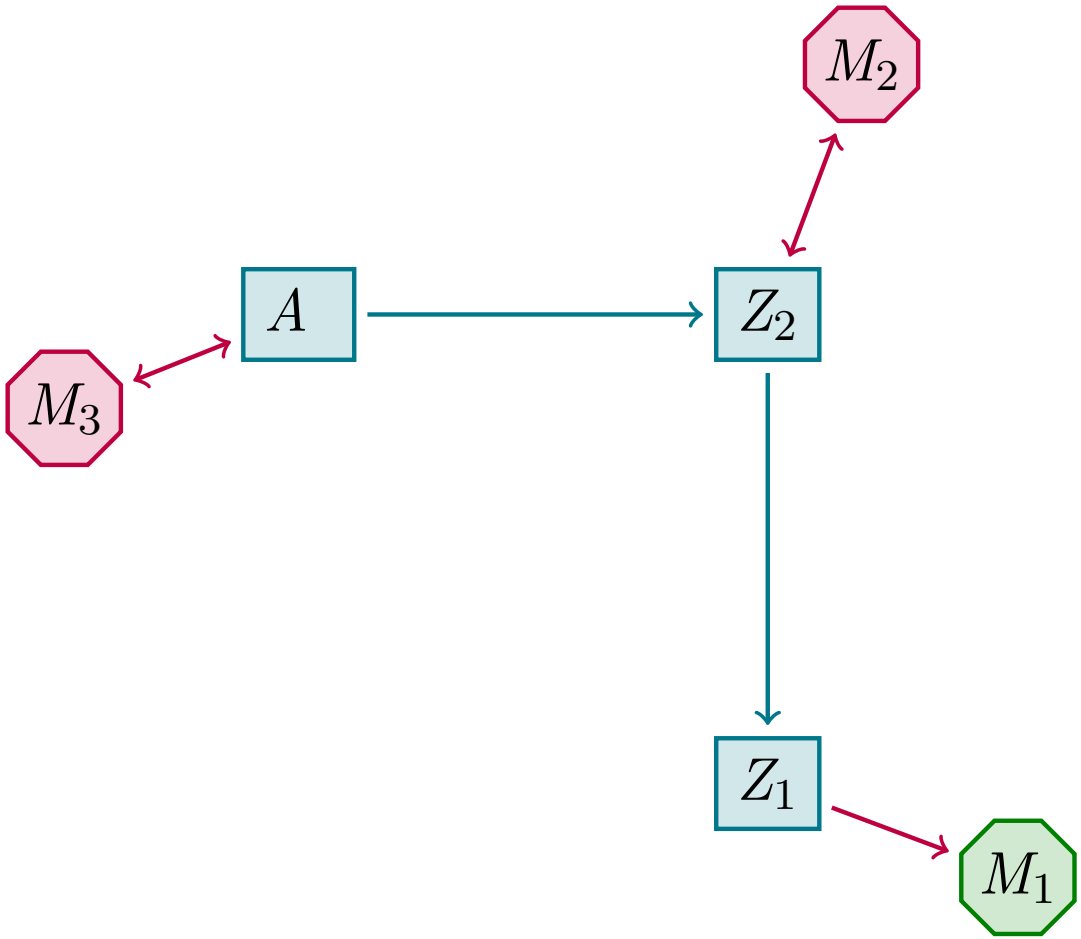
Letzte bekannte Position



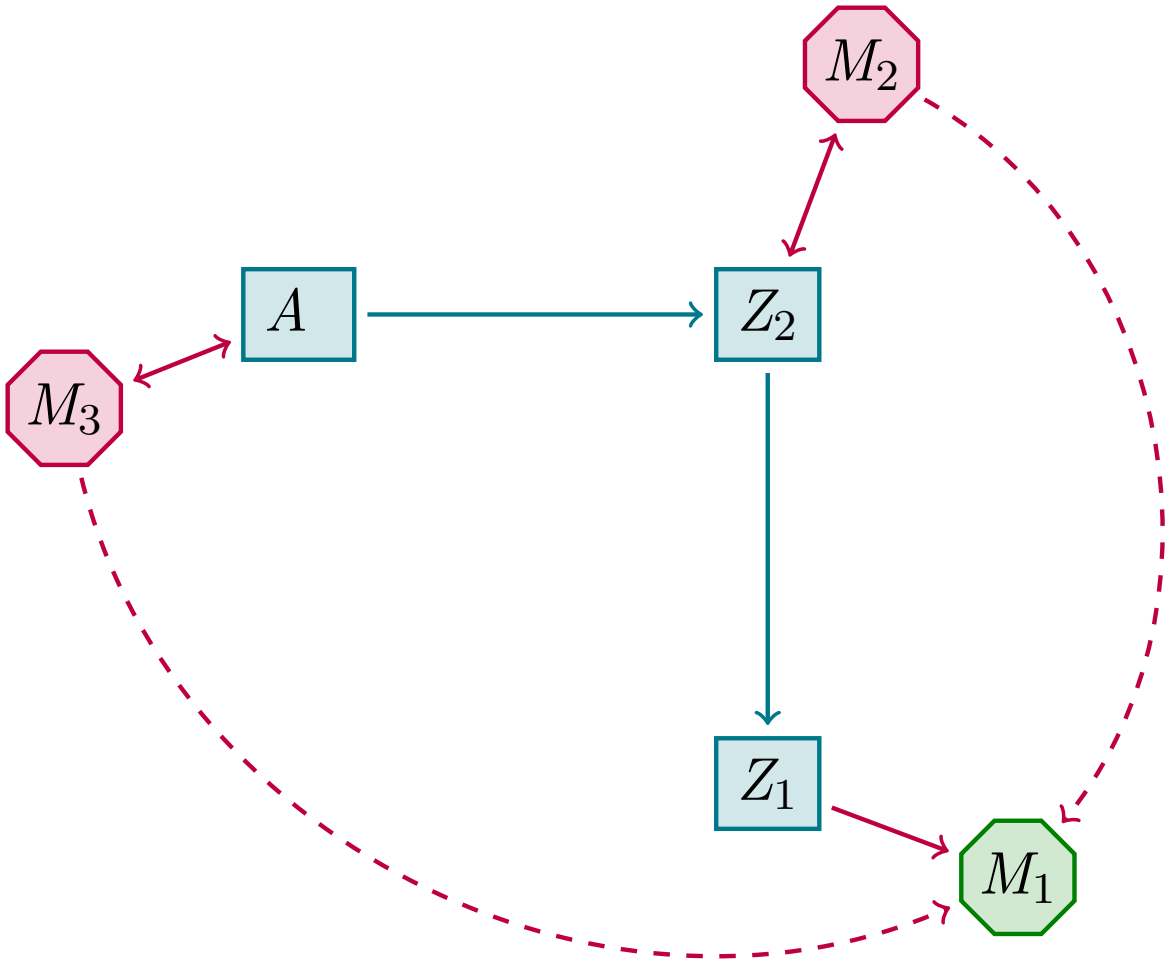
Letzte bekannte Position



Letzte bekannte Position

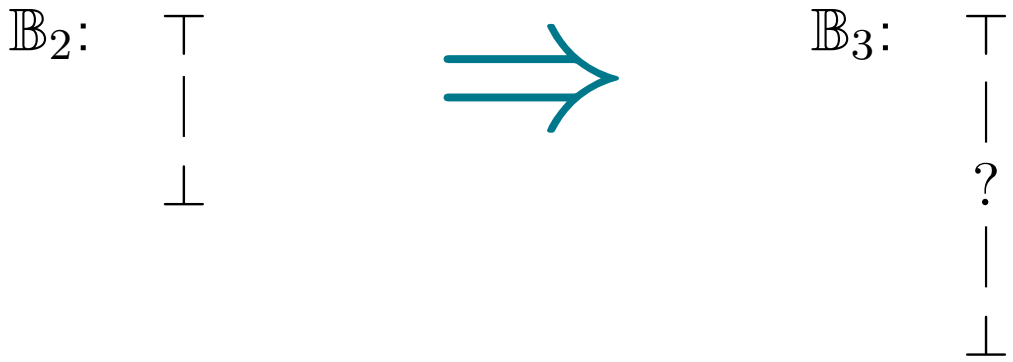


Letzte bekannte Position



LTL_3

Dreiwertige Semantik

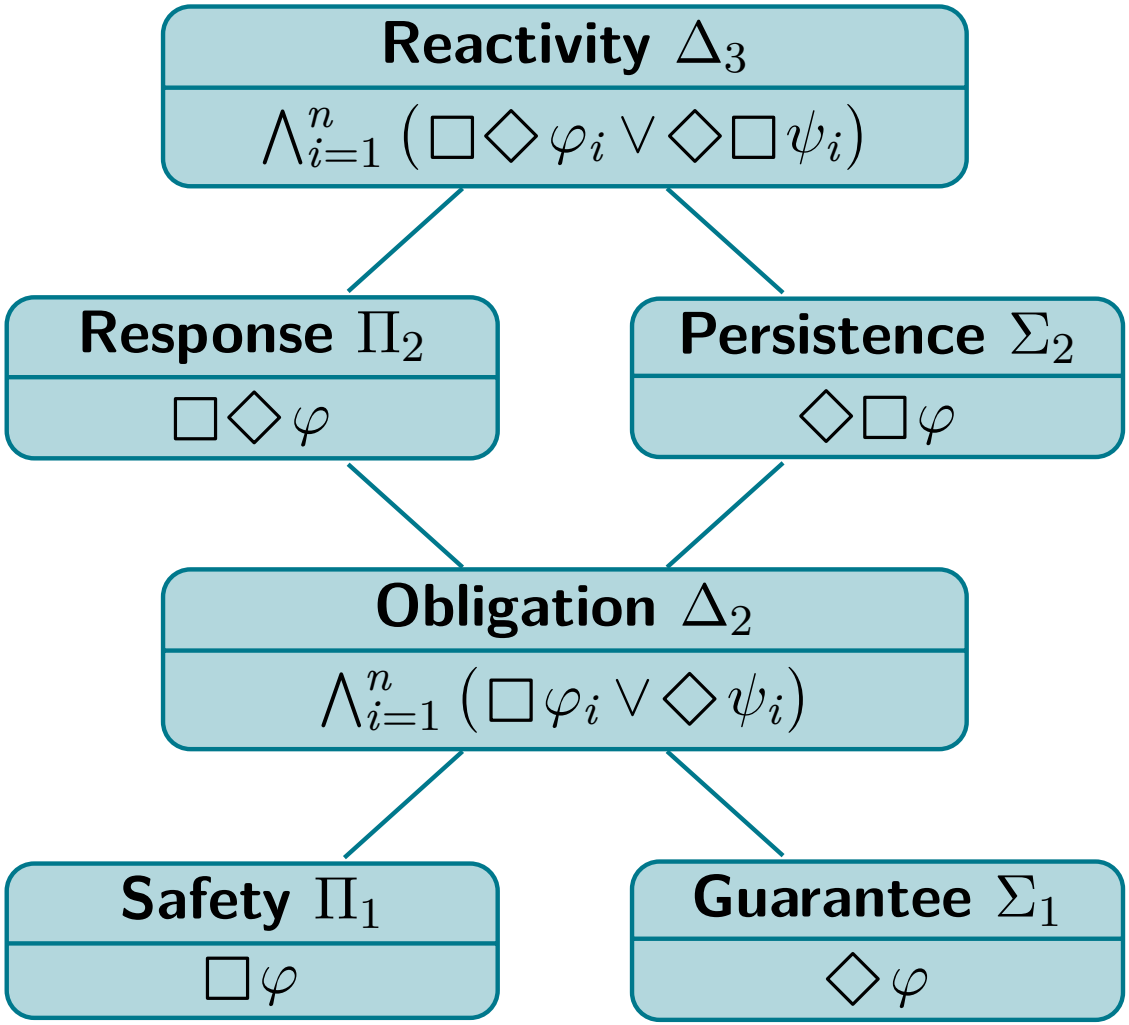


$$[\![u \models \varphi]\!]_{\mathrm{LTL}_3} = \begin{cases} \top & \text{wenn } \forall w \in \Sigma^\omega : uw \models \varphi \\ \bot & \text{wenn } \forall w \in \Sigma^\omega : uw \not\models \varphi \\ ? & \text{sonst.} \end{cases}$$

A. Bauer, M. Leucker, and C. Schallhart.
Runtime Verification for LTL and TLTL.
ACM ToSEM, 2011.

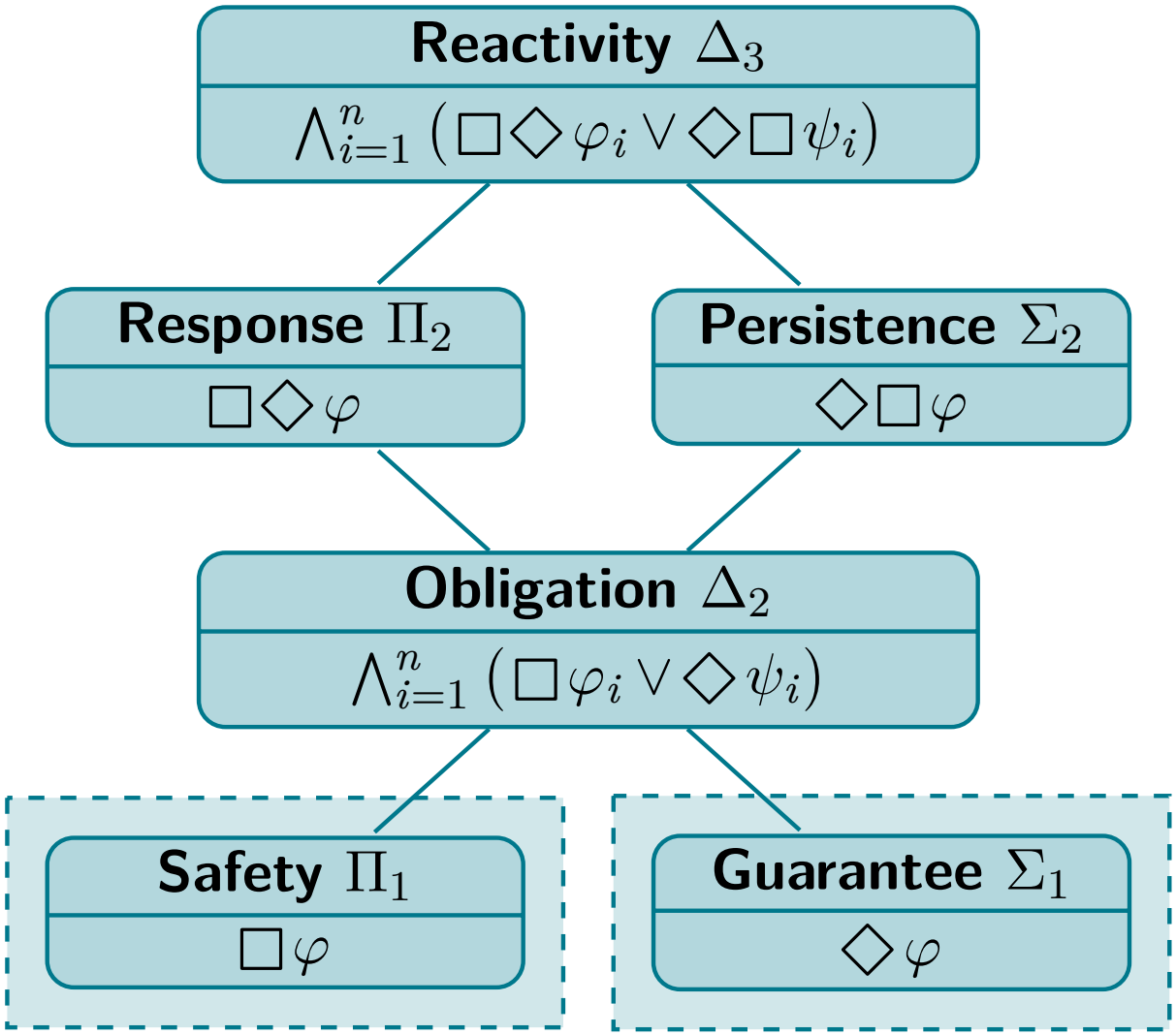
Monitorbarkeit

Temporale Hierarchie (Z. Manna und A. Pnueli, 1990.)



Monitorbarkeit

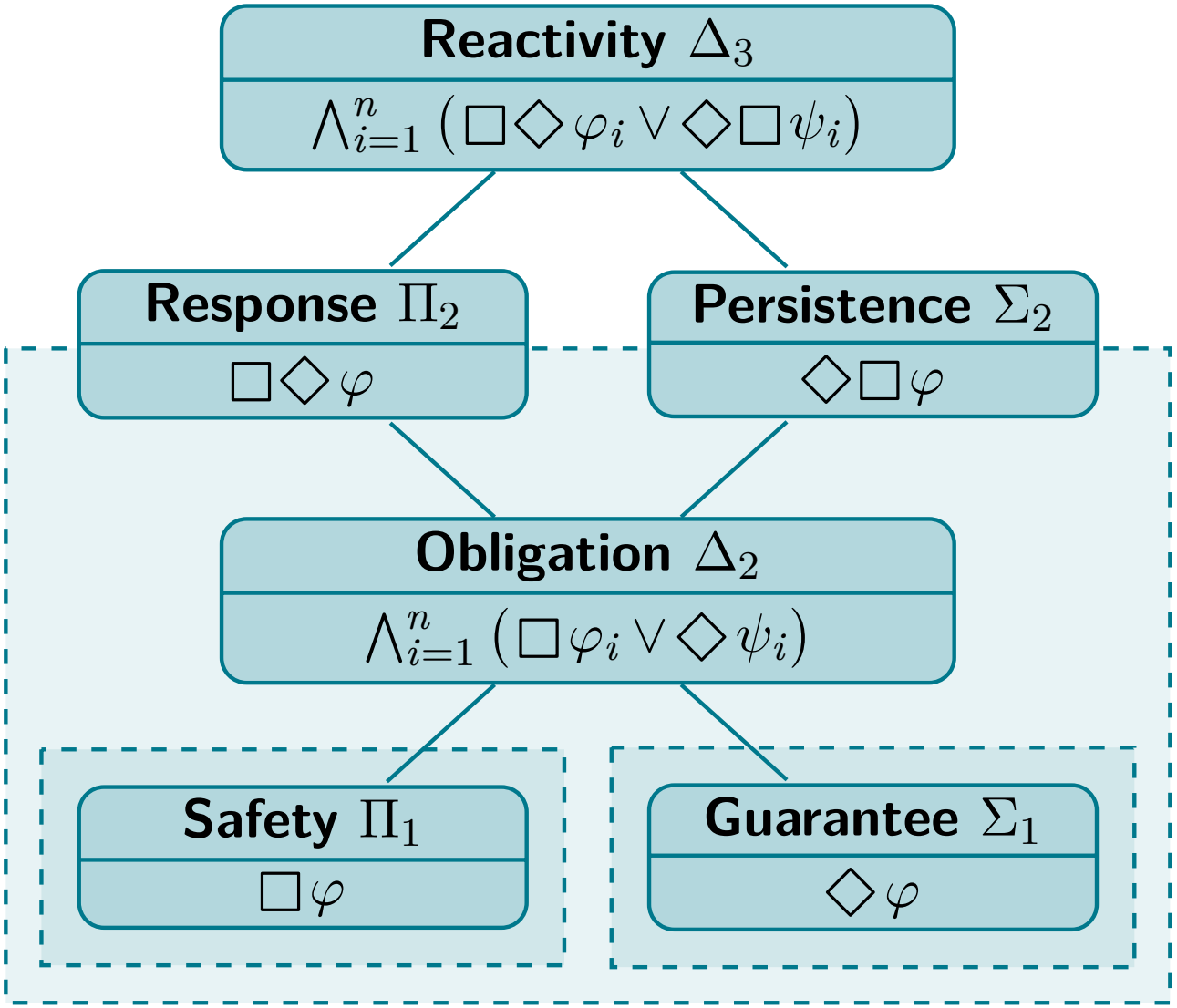
Temporale Hierarchie (Z. Manna und A. Pnueli, 1990.)



$\mathbb{B}_2$

Monitorbarkeit

Temporale Hierarchie (Z. Manna und A. Pnueli, 1990.)

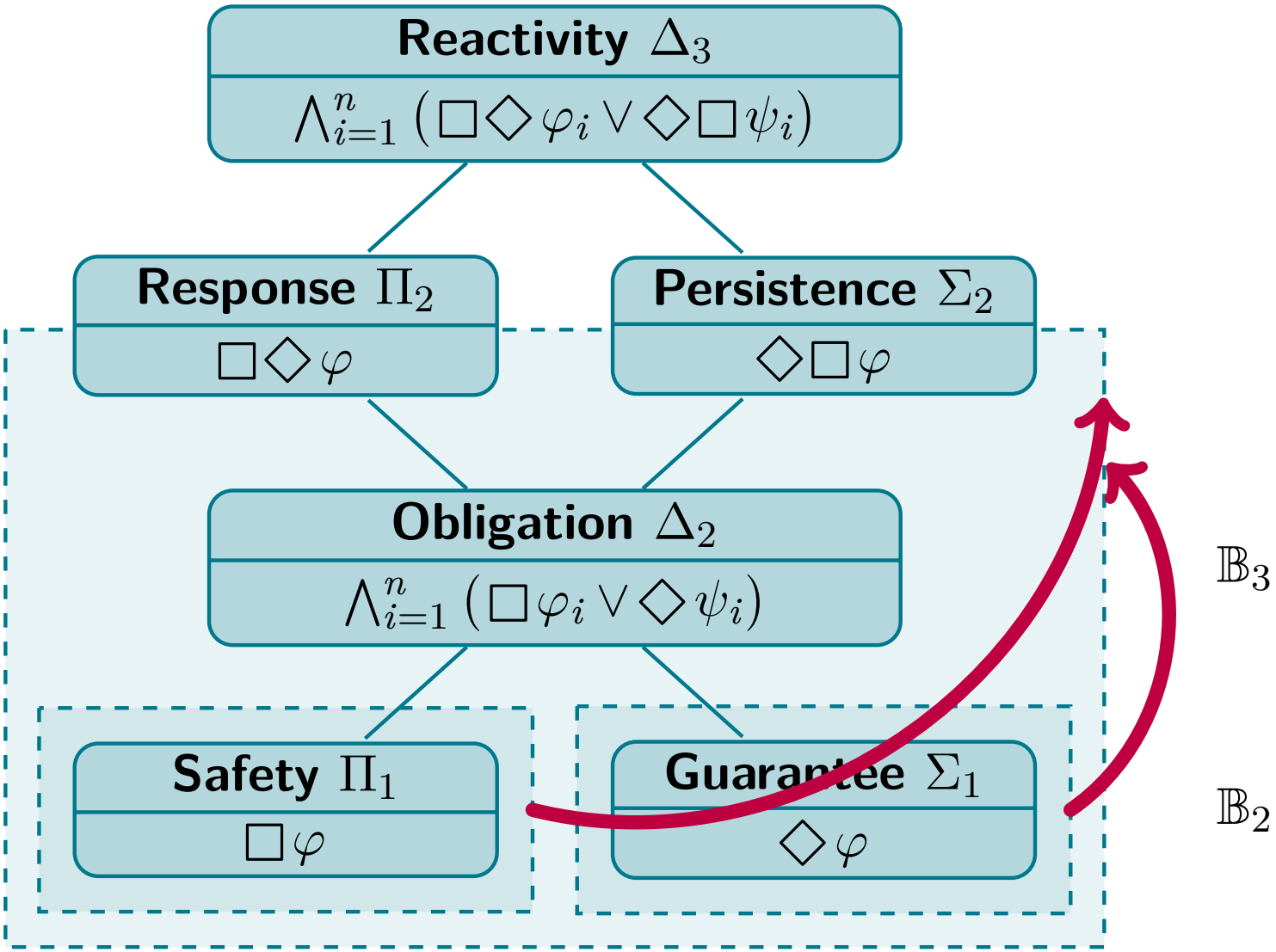


$\mathbb{B}_3$

$\mathbb{B}_2$

Monitorbarkeit

Temporale Hierarchie (Z. Manna und A. Pnueli, 1990.)



Distributed Temporal Logic (DTL)

Idee

- Logik auf endlichen Worten
- Verwendung bestehender Semantiken
- Entfernte Teilformeln = Propositionen

$$@_{Z_2}^{\mathrm{LTL}_3}(\neg\, s_1\, \mathcal{U} @_{Z_1}^{\mathrm{ptLTL}}(m_A \wedge \ominus \diamondsuit\, s_1))$$

DTL-Semantik

Angereicherte Projektion

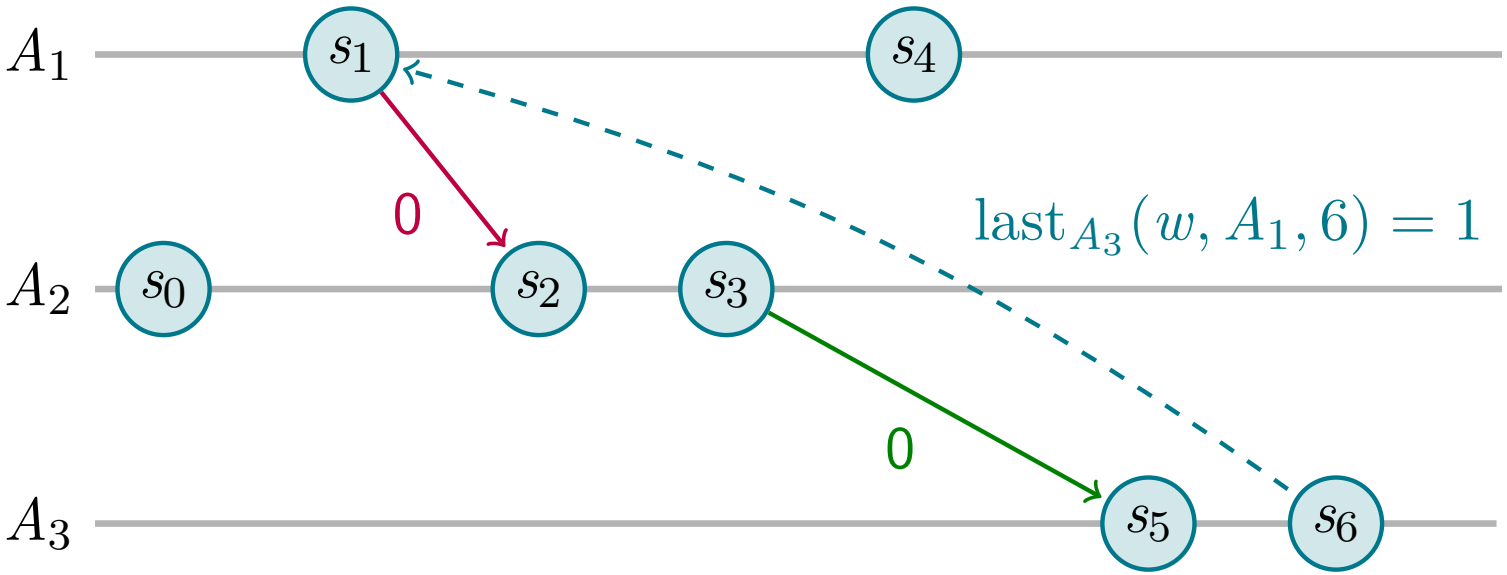
- Projektion von w auf Agent A
- entfernte Proposition r in w_i
  - entfernte Teilformel $@_{A_r}^{\mathrm{TL}_r} \varphi_r$
  - Auswertung φ_r in TL_r
  - auf Projektion von w auf Agent A_r
  - Anfang bis letzte bekannte Position auf A_r

DTL-Semantik

Angereicherte Projektion

- Projektion von w auf Agent A
- entfernte Proposition r in w_i
  - entfernte Teilformel $@_{A_r}^{\mathrm{TL}_r} \varphi_r$
  - Auswertung φ_r in TL_r
  - auf Projektion von w auf Agent A_r
  - Anfang bis letzte bekannte Position auf A_r

Letzte bekannte Position



Verwendung bestehender Semantiken

$$@_{Z_2}^{\mathrm{LTL}_3}(\neg\, s_1\, \mathcal{U} @_{Z_1}^{\mathrm{ptLTL}}(m_A \wedge \ominus \diamondsuit\, s_1))$$

	ptLTL	LTL_3
Hauptformel		✔
Teilformel	✔	

Verwendung bestehender Semantiken

$$@^{\mathrm{LTL}_3}_{Z_2}(\neg\, s_1\, \mathcal{U} @^{\mathrm{ptLTL}}_{Z_1}(m_A \wedge \ominus \diamondsuit\, s_1))$$

	ptLTL	LTL_3
Hauptformel	✔	✔
Teilformel	✔	

Verwendung bestehender Semantiken

$$@_{Z_2}^{\mathrm{LTL}_3}(\neg\, s_1\, \mathcal{U} @_{Z_1}^{\mathrm{ptLTL}}(m_A \wedge \ominus \diamondsuit\, s_1))$$

	ptLTL	LTL_3
Hauptformel	✔	✔
Teilformel	✔	✘

Verwendung bestehender Semantiken

$$@_{Z_2}^{\mathrm{LTL}_3}(\neg\, s_1\, \mathcal{U} @_{Z_1}^{\mathrm{ptLTL}}(m_A \wedge \ominus \diamondsuit\, s_1))$$

	ptLTL	LTL_3
Hauptformel	✔	✔
Teilformel	✔	✘

future Distributed Temporal Logic (fDTL)

fDTL_ω-Semantik auf unendlichen Worten (vgl. LTL)

- $p \in \mathrm{AP}$ atomare Proposition
- $r \in \mathrm{CP}$ dreiwertige Proposition

$$[\![(w,i) \models p]\!]_{\mathrm{fDTL}_\omega} = w_i(p)$$

$$[\![(w,i) \models r]\!]_{\mathrm{fDTL}_\omega} = \begin{cases} w_k(r) & \text{wenn } \exists k : w_k(r) \in \{\top, \bot\} \\ ? & \text{sonst} \end{cases}$$

future Distributed Temporal Logic (fDTL)

fDTL_ω-Semantik auf unendlichen Worten (vgl. LTL)

- $p \in \mathrm{AP}$ atomare Proposition
- $r \in \mathrm{CP}$ dreiwertige Proposition

$$[\![(w,i) \models p]\!]_{\mathrm{fDTL}_\omega} = w_i(p)$$

$$[\![(w,i) \models r]\!]_{\mathrm{fDTL}_\omega} = \begin{cases} w_k(r) & \text{wenn } \exists k : w_k(r) \in \{\top, \bot\} \\ ? & \text{sonst} \end{cases}$$

fDTL-Semantik auf endlichen Worten (vgl. LTL_3)

- $\top$ oder $\bot$ wenn alle Verlängerungen gleich
- sonst $?$

ptLTL-Monitorgenerierung

Beispiel ptLTL-Formel

$$\varphi = a \wedge \ominus\ominus b$$

ptLTL-Monitorgenerierung

Beispiel ptLTL-Formel

$$\varphi = a \wedge \ominus \ominus b$$

temporale Teilformeln

$$\varphi_1 = \ominus c$$
$$\varphi_2 = \ominus \varphi_1$$

ptLTL-Monitorgenerierung

Beispiel ptLTL-Formel

$$\varphi = a \wedge \ominus \ominus b$$

temporale Teilformeln

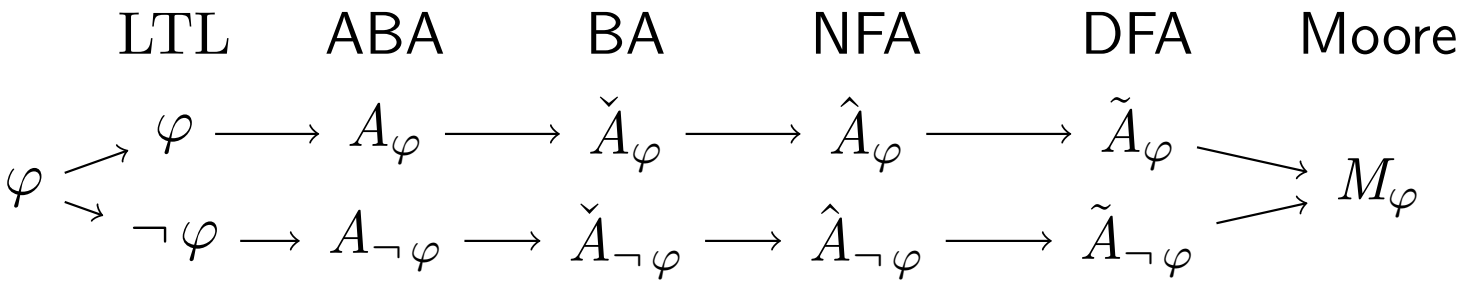
$$\varphi_1 = \ominus c$$
$$\varphi_2 = \ominus \varphi_1$$

Monitor

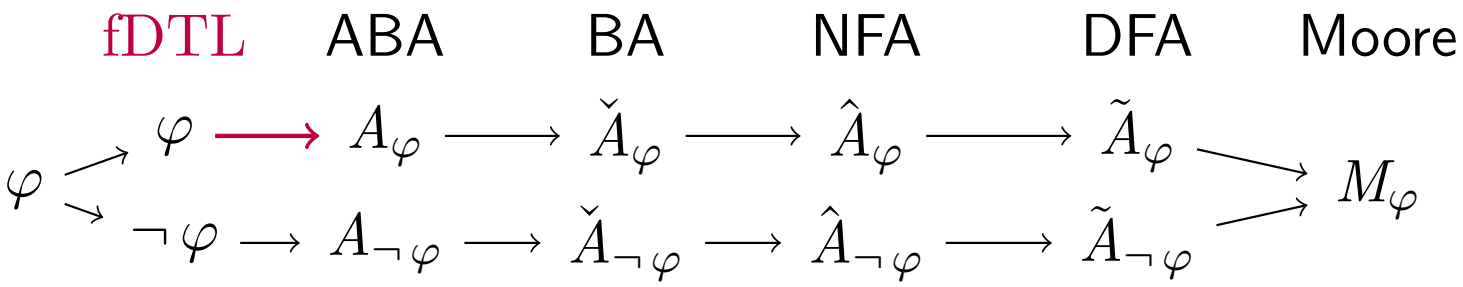
Zustandsübergang s nach s' / Ausgabe b

$$s'(1) = \text{eval}(c)$$
$$s'(2) = s(1)$$
$$b = \text{eval}(a) \sqcap s(2)$$

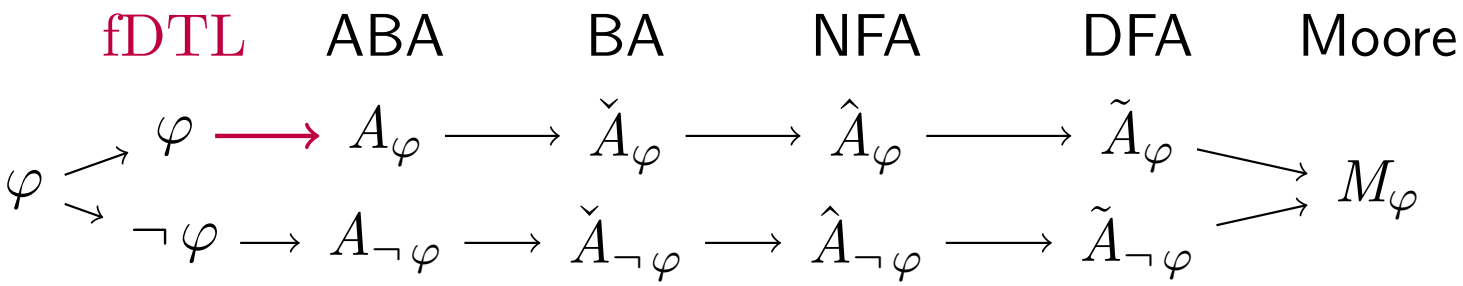
fDTL-Monitorgenerierung



fDTL-Monitorgenerierung



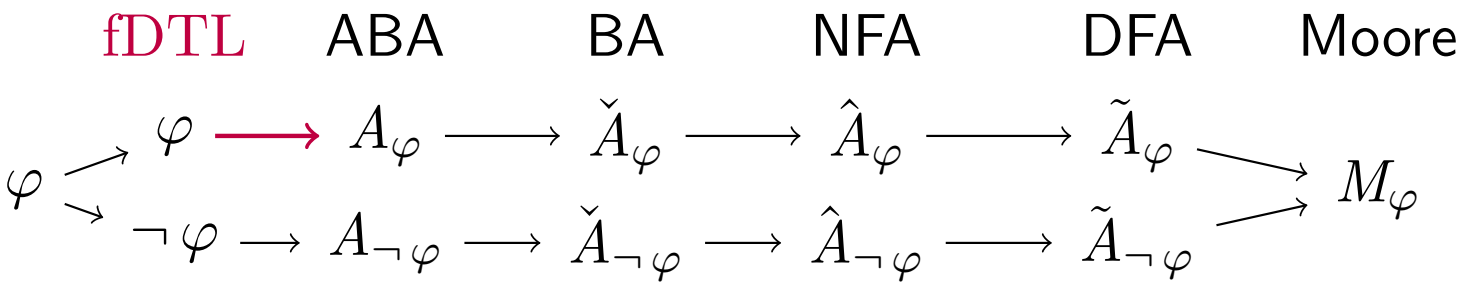
fDTL-Monitorgenerierung



Beispiel fDTL-Formel

$$(\Diamond\, p) \vee r \qquad p \in \mathrm{AP}, r \in \mathrm{CP}$$

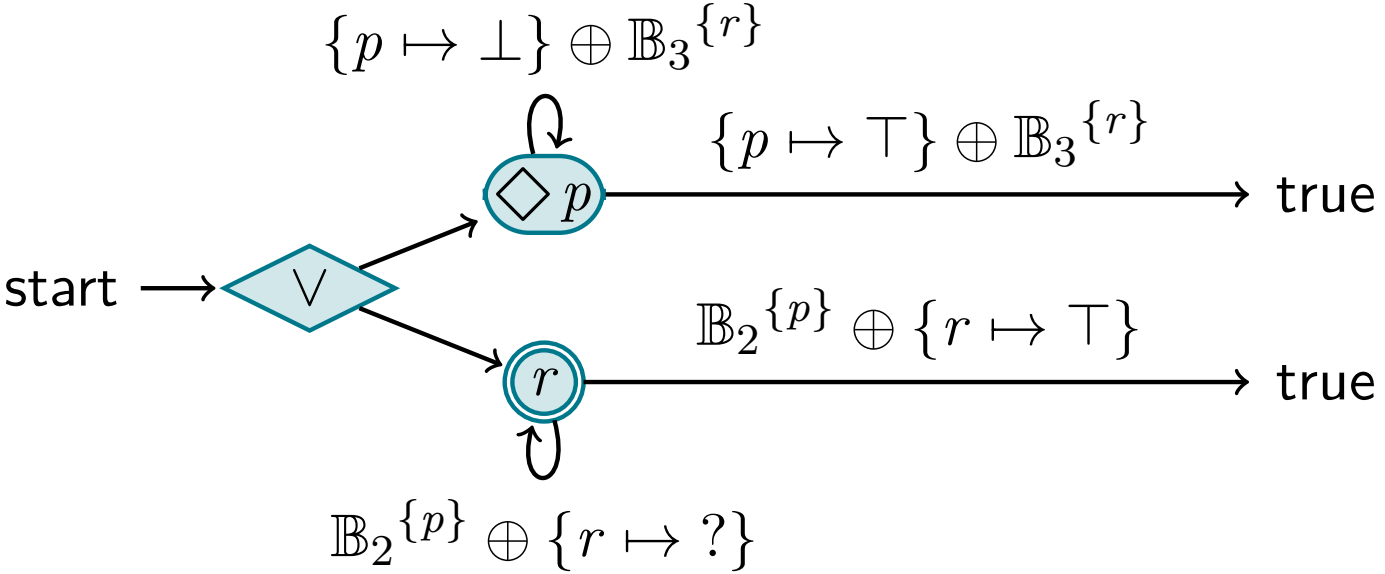
fDTL-Monitorgenerierung



Beispiel fDTL-Formel

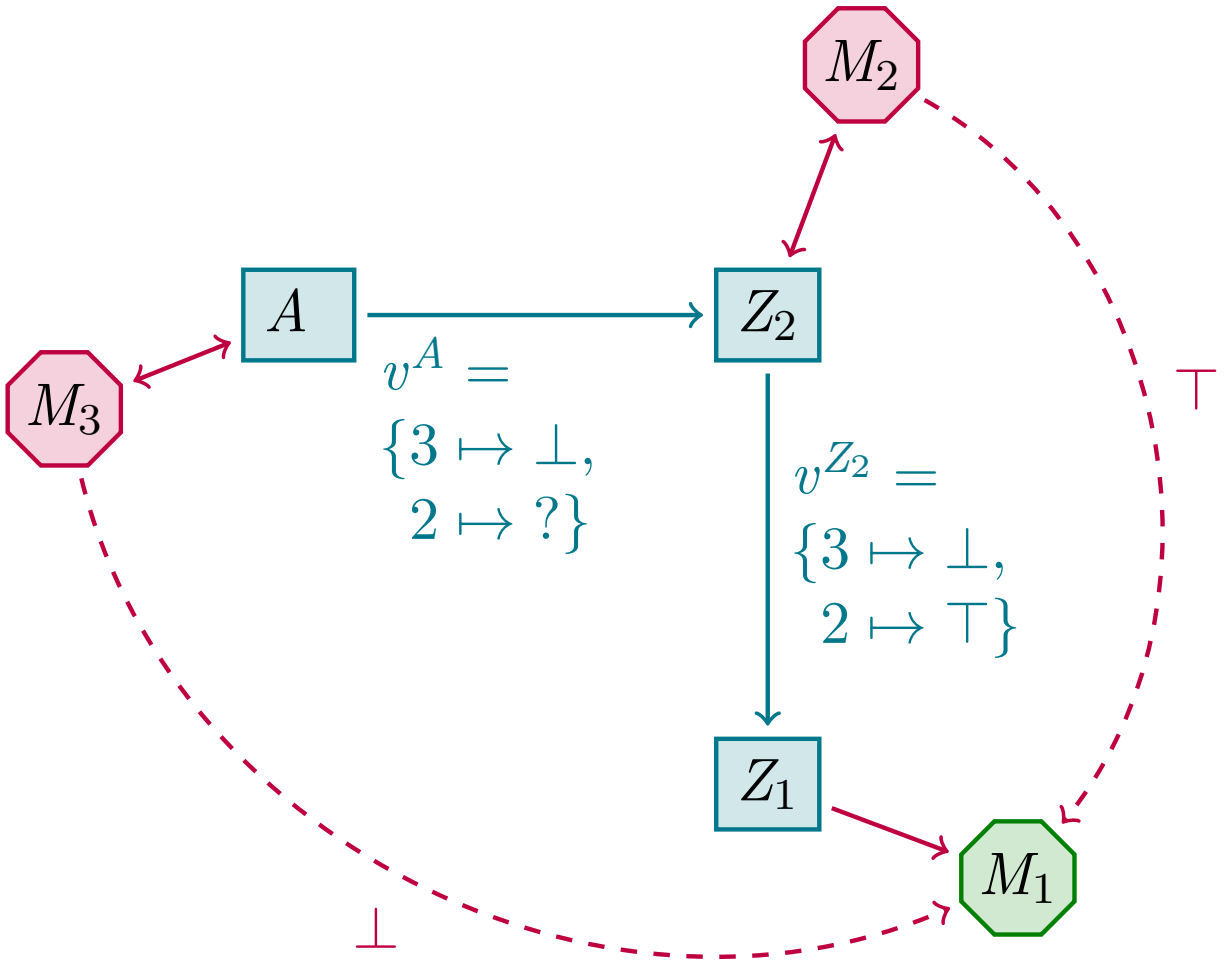
$$(\Diamond\, p) \vee r \qquad p \in \mathrm{AP}, r \in \mathrm{CP}$$

Monitor



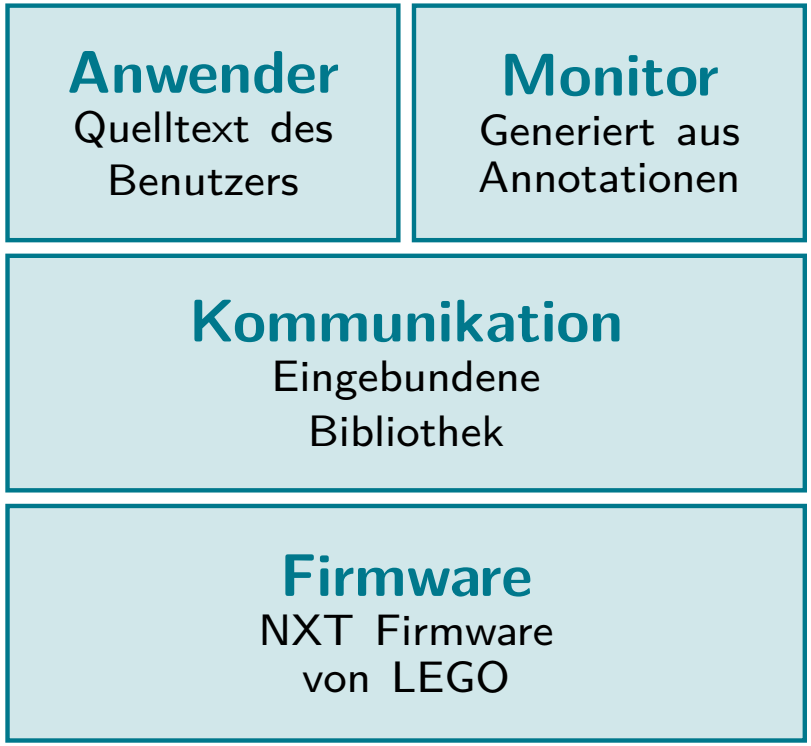
DTL-Monitorgenerierung

Knowledge Vector, Schnappschuss-Algorithmus (Lamport, 1985)



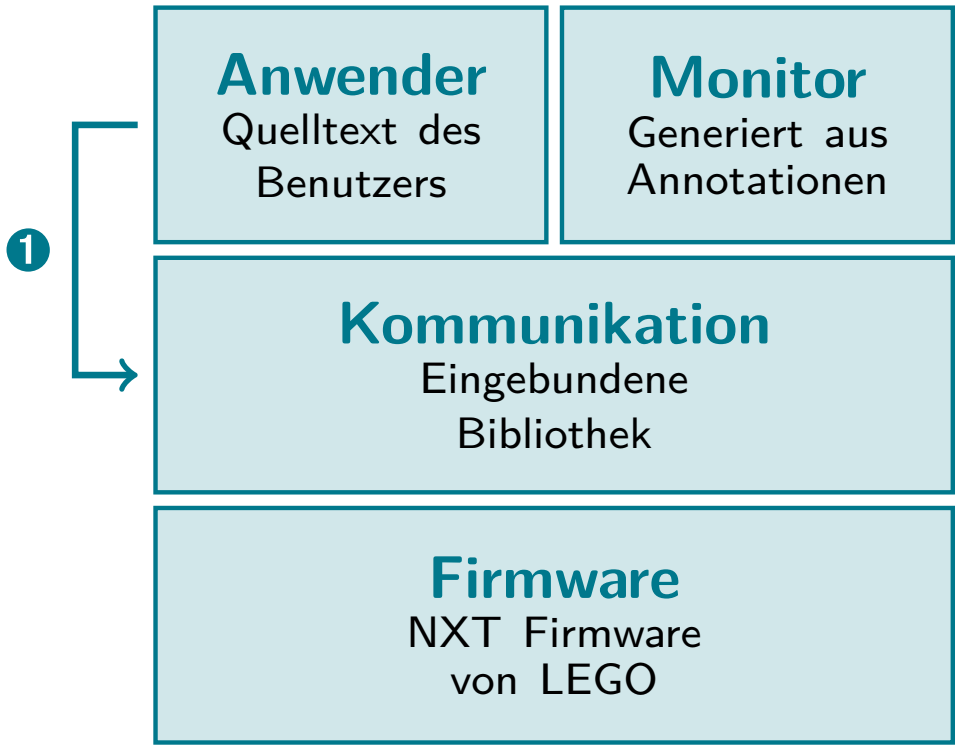
Versenden von Nachrichten

Datenfluss zwischen den Architekturschichten auf dem NXT



Versenden von Nachrichten

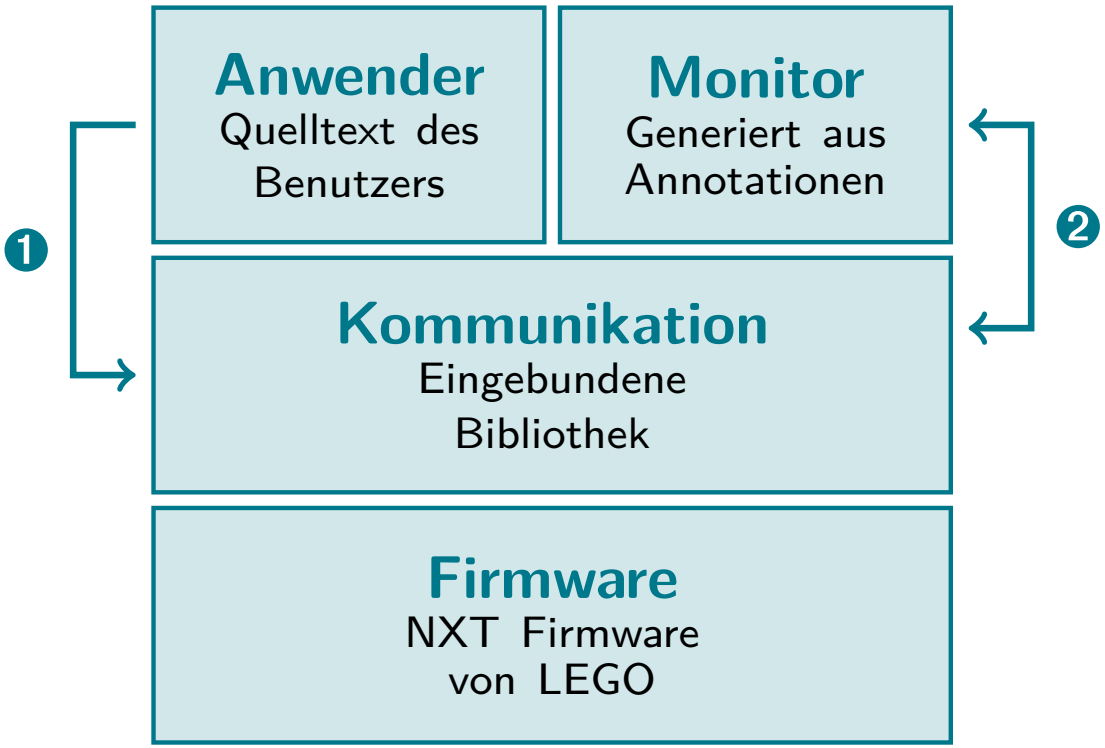
Datenfluss zwischen den Architekturschichten auf dem NXT



1. Anwender schickt Nachricht

Versenden von Nachrichten

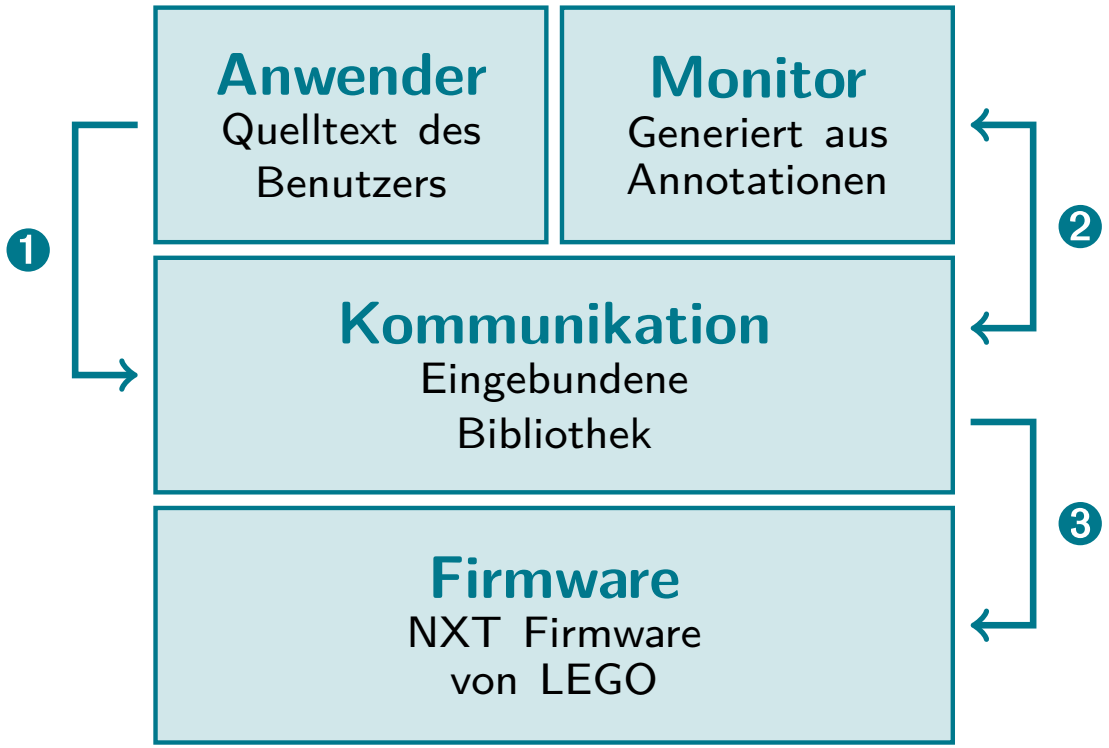
Datenfluss zwischen den Architekturschichten auf dem NXT



1. Anwender schickt Nachricht
2. Monitor hängt KV an

Versenden von Nachrichten

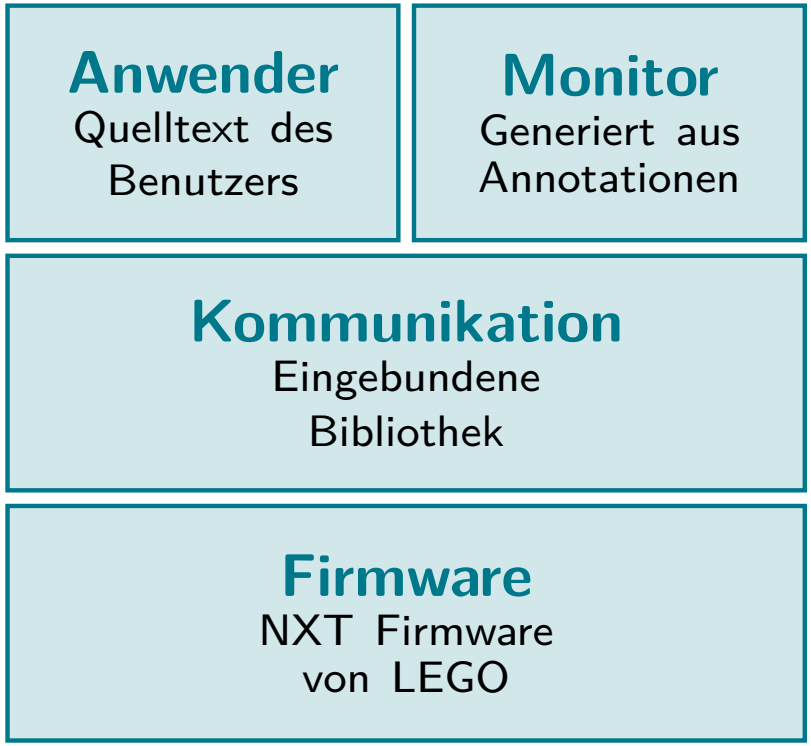
Datenfluss zwischen den Architekturschichten auf dem NXT



1. Anwender schickt Nachricht
2. Monitor hängt KV an
3. Firmware versendet Daten

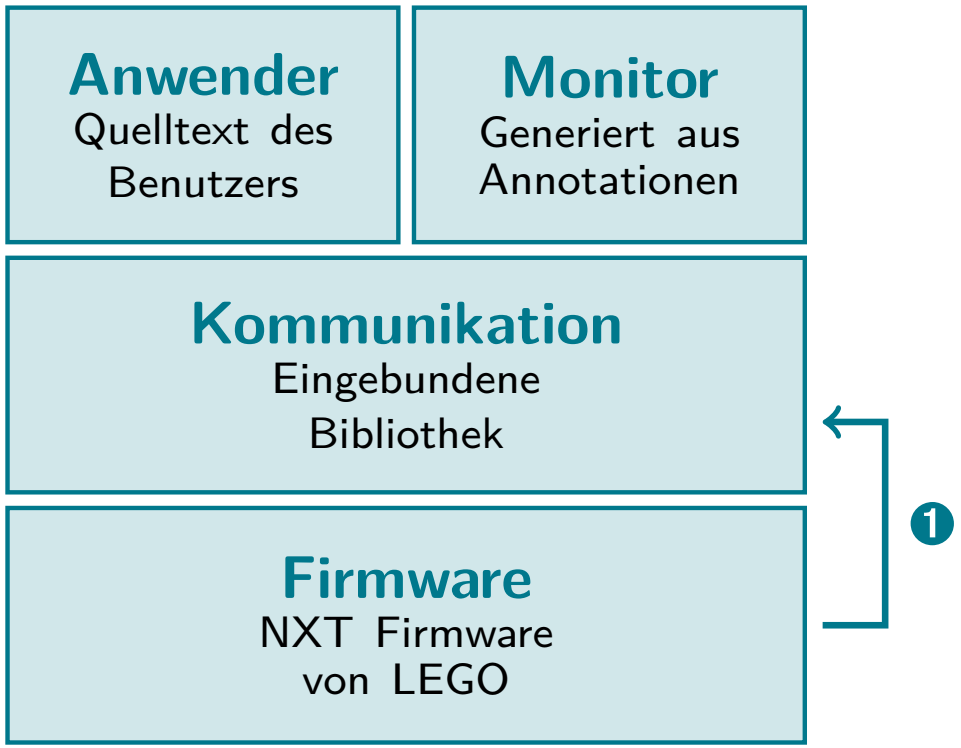
Empfangen von Nachrichten

Datenfluss zwischen den Architekturschichten auf dem NXT



Empfangen von Nachrichten

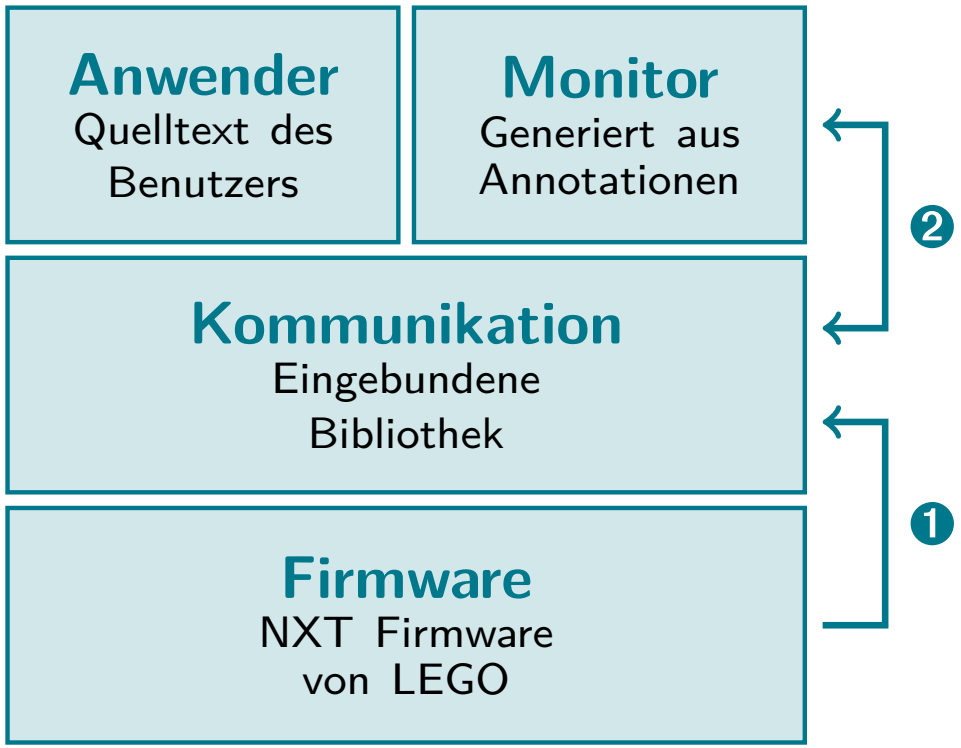
Datenfluss zwischen den Architekturschichten auf dem NXT



❶ Firmware empfängt Daten

Empfangen von Nachrichten

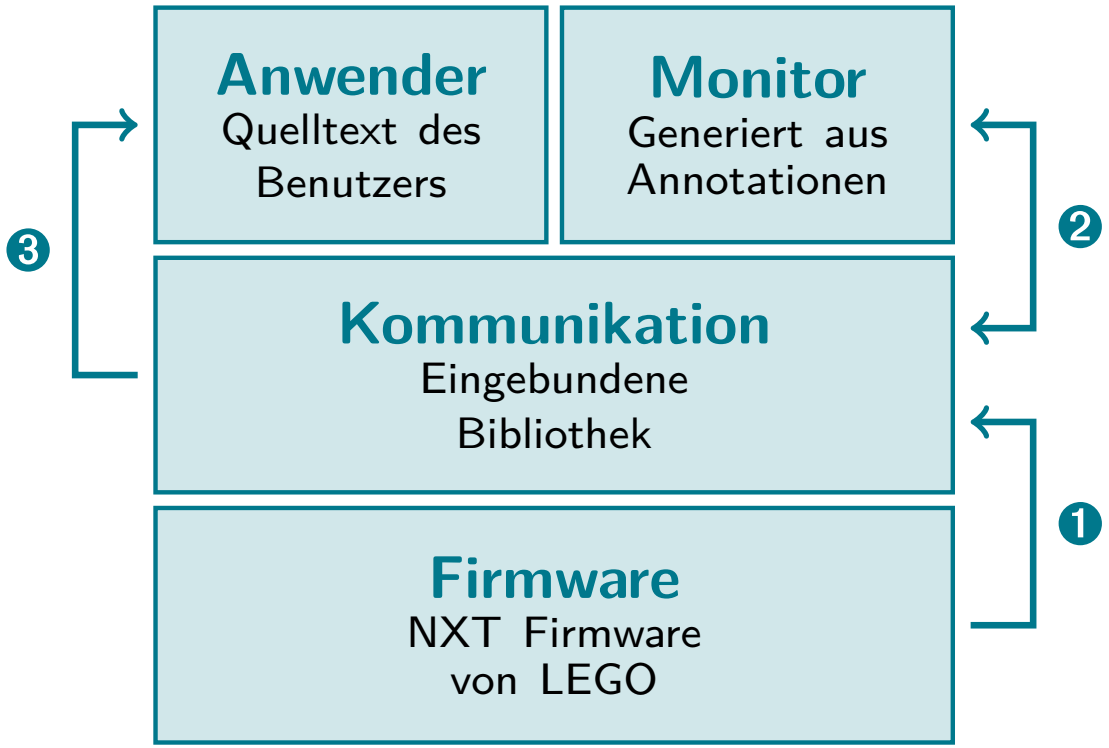
Datenfluss zwischen den Architekturschichten auf dem NXT



❶ Firmware empfängt Daten
❷ Monitor liest KV aus

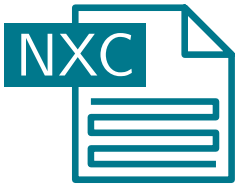
Empfangen von Nachrichten

Datenfluss zwischen den Architekturschichten auf dem NXT



❶ Firmware empfängt Daten
❷ Monitor liest KV aus
❸ Anwender erhält Nachricht

Präprozessierung des Quelltexts

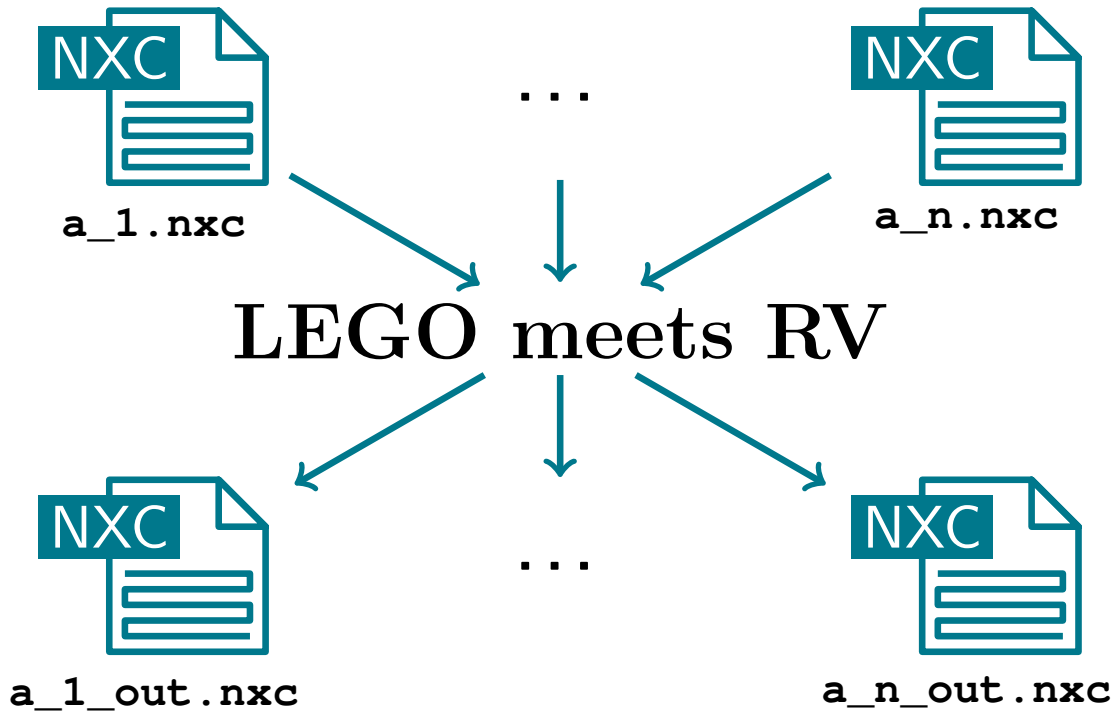


`a_1.nxc`

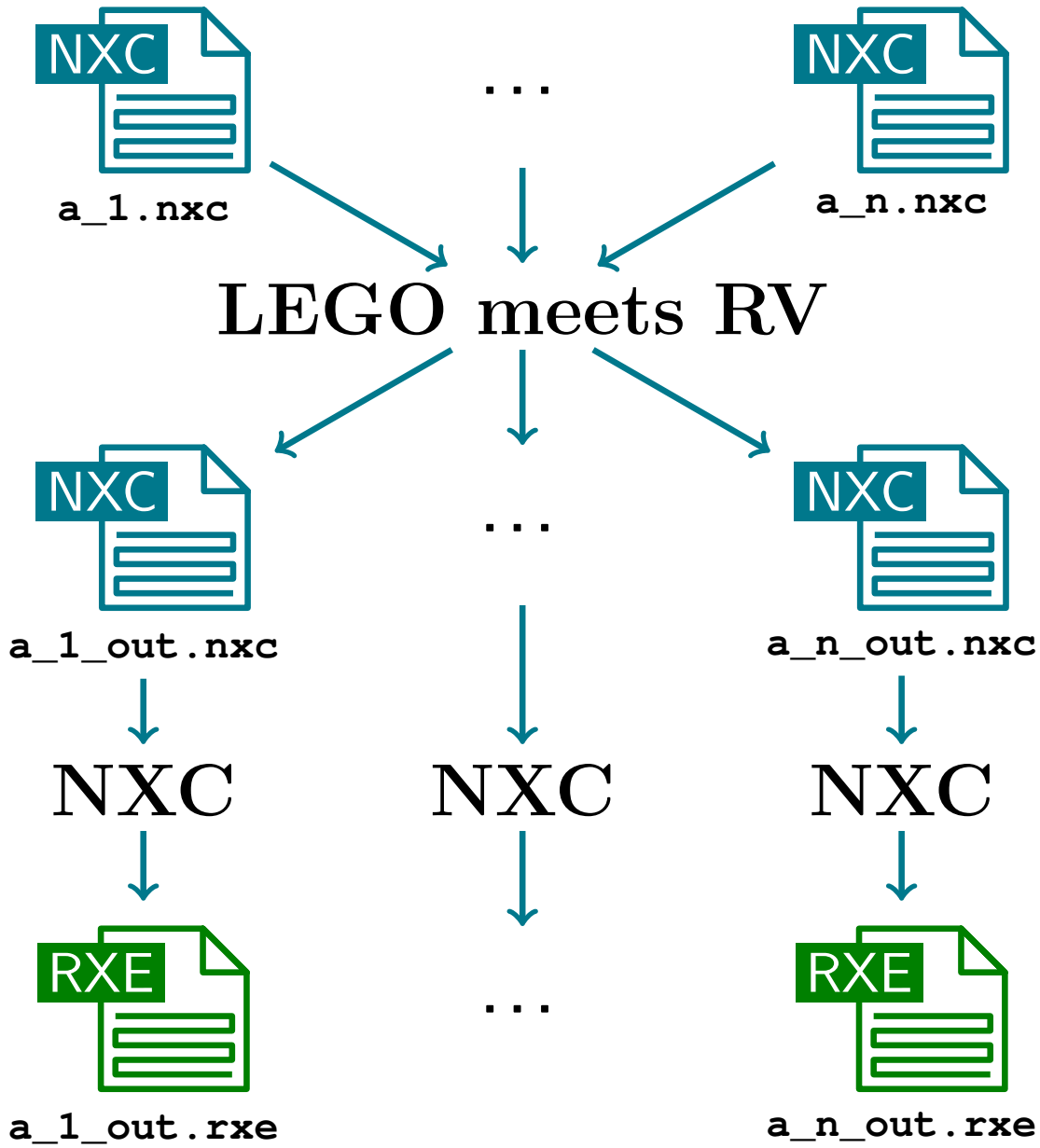
...

`a_n.nxc`

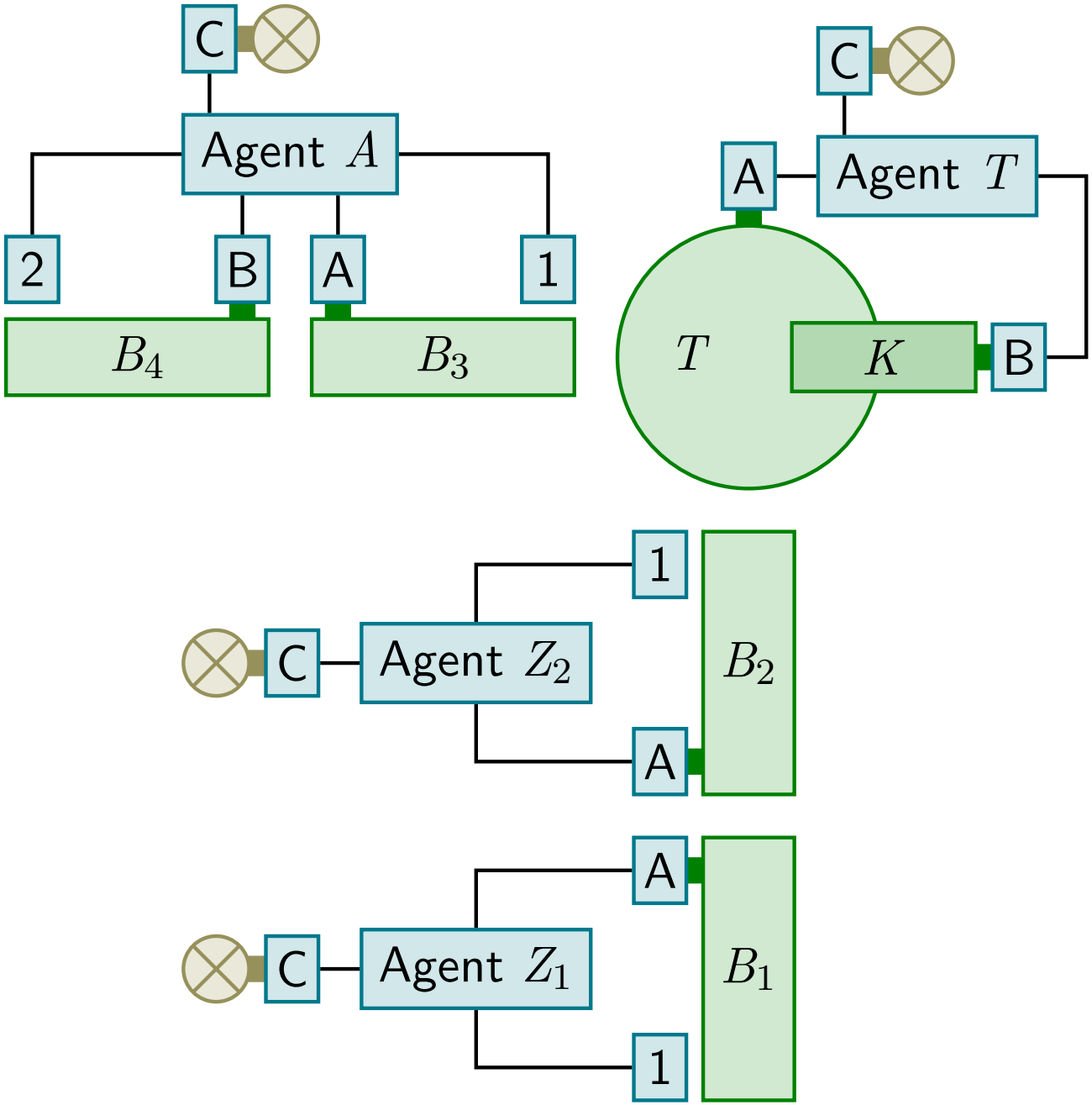
Präprozessierung des Quelltexts



Präprozessierung des Quelltexts



Fallstudie


C
Agent A
2
B
A
1
B_4
B_3
C
A
Agent T
T
K
B
1
C
Agent Z_2
B_2
A
A
C
Agent Z_1
B_1
1

Fallstudie

Beispiel DTL-Formel

$$@^{\mathrm{fDTL}}_{Z_1} \Box \Big((s_1 \wedge \Diamond\, @^{\mathrm{ptLTL}}_{Z_2} s_1) \rightarrow (((@^{\mathrm{ptLTL}}_{Z_2} s_1) \rightarrow (\neg\, s_1\, \mathcal{U}\, @^{\mathrm{ptLTL}}_{A} s_2))\, \mathcal{U}\, @^{\mathrm{ptLTL}}_{A} s_2) \Big)$$

Fallstudie

Beispiel DTL-Formel

$$@_{Z_1}^{\mathrm{fDTL}} \Box \Big((s_1 \wedge \Diamond @_{Z_2}^{\mathrm{ptLTL}} s_1) \rightarrow (((@_{Z_2}^{\mathrm{ptLTL}} s_1) \rightarrow (\neg\, s_1\, \mathcal{U}\, @_{A}^{\mathrm{ptLTL}} s_2))\, \mathcal{U}\, @_{A}^{\mathrm{ptLTL}} s_2) \Big)$$

Fallstudie

Beispiel DTL-Formel

$$@_{Z_1}^{\text{fDTL}} \Box \Big((s_1 \wedge \Diamond @_{Z_2}^{\text{ptLTL}} s_1) \rightarrow (((@_{Z_2}^{\text{ptLTL}} s_1) \rightarrow (\neg\, s_1\, \mathcal{U}\, @_{A}^{\text{ptLTL}} s_2))\, \mathcal{U}\, @_{A}^{\text{ptLTL}} s_2) \Big)$$

Fallstudie

Beispiel DTL-Formel

$$@_{Z_1}^{\text{fDTL}} \Box \Big((s_1 \wedge \Diamond @_{Z_2}^{\text{ptLTL}} s_1) \rightarrow (((@_{Z_2}^{\text{ptLTL}} s_1) \rightarrow (\neg s_1 \,\mathcal{U}\, @_{A}^{\text{ptLTL}} s_2)) \,\mathcal{U}\, @_{A}^{\text{ptLTL}} s_2) \Big)$$

Fallstudie

Beispiel DTL-Formel

$$@^{\mathrm{fDTL}}_{Z_1} \Box \Big((s_1 \wedge \Diamond\, @^{\mathrm{ptLTL}}_{Z_2} s_1) \rightarrow (((@^{\mathrm{ptLTL}}_{Z_2} s_1) \rightarrow (\neg\, s_1\, \mathcal{U}\, @^{\mathrm{ptLTL}}_{A} s_2))\, \mathcal{U}\, @^{\mathrm{ptLTL}}_{A} s_2) \Big)$$

Lokale Propositionen

```
//= PROPOSITION s1 DEFINE ↵
      (Sensor(S1) >= STONE)
```

Entfernte Propositionen (= entfernte Teilformeln)

```
//= PROPOSITION a2 EXTERNAL ausgabe
//= PROPOSITION z2 EXTERNAL zufuhr2
```

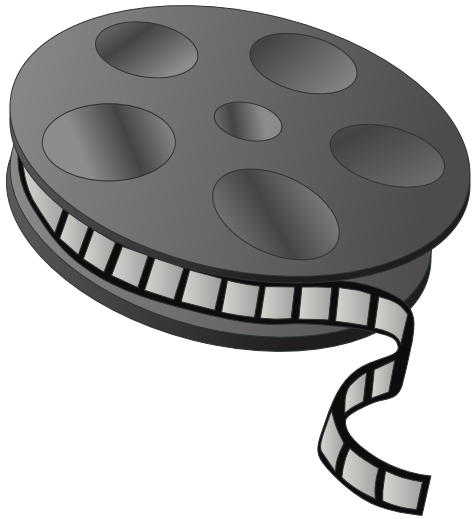
Formel bzw. Monitor

```
//= MONITOR aa FDTL = G((s1 && F z2) -> ↵
      ((z2 -> (!s1 U a2)) U a2))
```

Ereignisse (= Weiterschalten des Monitors)

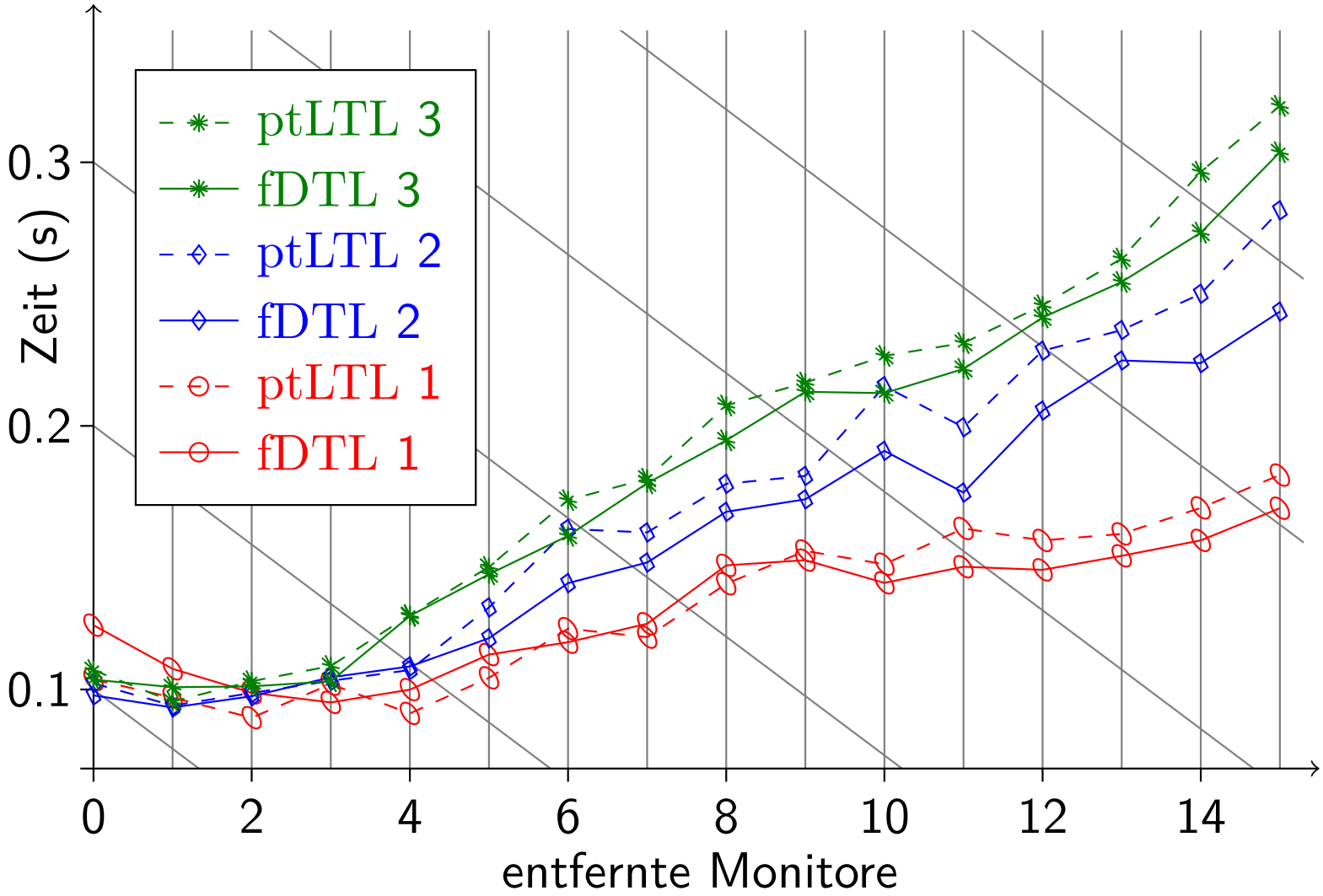
```
//= EVENT aa CHANGE s1, e2, z2
```

Fallstudie



Benchmarks: Umlaufzeit einer Nachricht

Hauptformel: ptLTL, fDTL / **entfernte Formel:** ptLTL
neue Hauptformel: ab 7 / **atomare Propositionen: 1–3**



Zusammenfassung

1. DTL erweitert vorhandene Logiken um den @-Operator. ptLTL und LTL_3 als Hauptformel direkt verwenden.
2. fDTL erlaubt LTL_3 als entfernte Teilformel.
3. DTL vergrößert die Monitorbarkeit.
4. DTL-Monitoring mit Knowledge-Vektoren, ptLTL-Monitoring mit Zustandsspeicher, fDTL-Monitoring mit ABAs
5. ptLTL-, ptDTL-, DTL-, LTL_3- und fDTL-Monitoring wurde auf LEGO Mindstorms NXT implementiert. Benchmarks zeigen gute Performance.

Ausblick

1. Monitor Oriented Programming (MOP)
   $\Longrightarrow$ Zurücksetzen von verteilten Monitoren
2. Weitere Logiken: RLTL oder RLTL_4
3. Implementierung / weitere Untersuchung von fSDTL
4. Automaten mit Propositionen an den Kanten
5. ...

Ziel

- Logiken und Monitorgenerierungen zur Verifikation verteilter Systeme
- Monitorbarkeit erweitern
- Verteilte Laufzeitverifikation auf LEGO Mindstorms implementierten
- Zustandsgenerierung und Propositionsbelegung im C-Code definieren

Verteiltes System

Verteiltes, asynchrones System

- keine globale Sicht
- Nachrichten
- Verzögerung

Verteiltes System

Verteiltes, asynchrones System

- keine globale Sicht
- Nachrichten
- Verzögerung

Verteiltes, synchrones System

Andreas Klaus Bauer and Yliès Falcone.
Decentralised LTL Monitoring.
18th Formal Methods, 2012.

Verteiltes System

Verteiltes, asynchrones System

- keine globale Sicht
- Nachrichten
- Verzögerung

Verteiltes, synchrones System

Andreas Klaus Bauer and Yliès Falcone.
Decentralised LTL Monitoring.
18th Formal Methods, 2012.

LEGO Mindstorms NXT

- Beispiel eines verteilten, eingebetteten Systems
- Not Exactly C (NXC) (J. Hansen, 2007)
- Präprozessierung des Quelltexts Annotationen $\rightarrow$ Monitore

Semantik

- $w \in \Sigma^+$ Wort, φ ptDTL-Formel, $p \in \mathrm{AP}$ atomare Proposition
- Semantikfunktion
  $[\![(\cdot,\cdot) \models \cdot]\!]_{\mathrm{ptLTL}} : \Sigma^* \times \mathbb{N} \times \mathrm{ptLTL} \to \mathbb{B}_2$

$$[\![(w,i) \models \ominus \varphi]\!]_{\mathrm{ptLTL}} =$$

$$\begin{cases} [\![(w,i+1) \models \varphi]\!]_{\mathrm{ptLTL}} & \text{wenn } i+1 < |w| \\ [\![(w,i) \models \varphi]\!]_{\mathrm{ptLTL}} & \text{sonst} \end{cases}$$

$$[\![(w,i) \models \varphi \,\mathcal{S}\, \psi]\!]_{\mathrm{ptLTL}} =$$

$$\begin{cases} \top & \text{wenn } \exists i \leq k < |w| : [\![(w,k) \models \psi]\!]_{\mathrm{ptLTL}} = \top \\ & \quad \text{und } \forall i \leq \ell < k : [\![(w,\ell) \models \varphi]\!]_{\mathrm{ptLTL}} = \top \\ \bot & \text{sonst} \end{cases}$$

Monitorkonstruktion

- $\mathrm{sub}_T(\varphi)$ Menge der temporalen Teilformeln
- Speicher $S = \{1, 2, \ldots, n\} \to \mathbb{B}_2$ für temporale Teilformeln

Schritt: Übergang von s nach s'

$$s'(i) = \begin{cases} \mathrm{eval}(\psi, a) & \text{wenn } \varphi_i = \ominus \psi \\ \mathrm{eval}(\psi' \vee (\psi \wedge s(i)), a) & \text{wenn } \varphi_i = \psi \,\mathcal{S}\, \psi' \end{cases}$$

Ausgabe: $b = \mathrm{eval}(\varphi)$

$$\mathrm{eval}(\varphi_i, a) = \begin{cases} s(i) & \text{wenn } \varphi_i = \ominus \psi \\ s'(i) & \text{wenn } \varphi_i = \psi \,\mathcal{S}\, \psi' \end{cases}$$

DTL-Semantik

Angereicherte Projektion

$\text{enrich}_{\varphi,A}(w,k) = u$

- Projektion auf Agent A
- Belegung entfernter Propositionen r in w_i zu entfernte Teilformel $@_{A_r}^{\text{TL}_r}\varphi_r$

$$r \mapsto [\![\text{enrich}_{\varphi_r,A_r}(w,\ell), 0 \models \varphi_r]\!]_{\text{TL}_r},$$
$$\ell = \text{last}_A(w, A_r, i)$$

$$[\![w, i \models @_A^{\text{TL}}\varphi]\!]_{\text{DTL}} = [\![\text{enrich}_{\varphi,A}(w, |w|-1), i \models \psi]\!]_{\text{TL}}$$

Semantik

Propositionen

Menge aller möglichen Belegungen

- $B^M = \{a \mid a : M \to B\}$
- $a \in \Sigma = \mathcal{B}_{\in}{}^{\mathrm{AP}}$ statt $a' \in \Sigma' = 2^{\mathrm{AP}}$
- $\forall p \in \mathrm{AP} : a(p) = \top \Leftrightarrow p \in a'$

- AP atomare Propositionen
- CP dreiwertige Propositionen (impartial!)
- $\Sigma = 3^{\mathrm{CP}} \oplus 2^{\mathrm{AP}}$ Alphabet

Kombination von Alphabeten

$$\Sigma = \mathcal{B}_2{}^{\mathrm{AP}} \oplus \mathcal{B}_3{}^{\mathrm{CP}}$$
$$= \{u \mid u \in \mathrm{AP} \cup \mathrm{CP} \to \mathcal{B}_3 \wedge \forall p \in \mathrm{AP} : u(p) \in \mathcal{B}_2\}$$

Semantik auf unendlichen Worten

- $w \in \mathrm{imp}(\Sigma^\omega)$ Wort, φ, ψ fDTL-Formeln, $p \in \mathrm{AP}$ atomare Proposition, $r \in \mathrm{CP}$ dreiwertige Proposition
- Semantikfunktion
  $[\![(\cdot,\cdot) \models \cdot]\!]_{\mathrm{fDTL}_\omega} : \Sigma^\omega \times \mathbb{N} \times \mathrm{LTL} \to \mathbb{B}_3$

$$[\![(w,i) \models p]\!]_{\mathrm{fDTL}_\omega} = w_i(p)$$

$$[\![(w,i) \models r]\!]_{\mathrm{fDTL}_\omega} = \begin{cases} w_k(r) & \text{wenn } \exists k : w_k(r) \in \{\top, \bot\} \\ ? & \text{sonst} \end{cases}$$

Semantik auf endlichen Worten

- $w \in \mathrm{imp}(\Sigma^*)$, φ fDTL-Formeln
- Semantikfunktion $[\![(\cdot,\cdot) \models \cdot]\!]_{\mathrm{fDTL}} : \Sigma^* \times \mathbb{N} \times \mathrm{LTL} \to \mathbb{B}_3$

$$[\![(w,i) \models \varphi]\!]_{\mathrm{fDTL}} = \begin{cases} \top & \text{wenn } \forall ww' \in \mathrm{imp}(\Sigma^\omega) : [\![(ww',i) \models \varphi]\!]_{\mathrm{fDTL}_\omega} = \top \\ \bot & \text{wenn } \forall ww' \in \mathrm{imp}(\Sigma^\omega) : [\![(ww',i) \models \varphi]\!]_{\mathrm{fDTL}_\omega} = \bot \\ ? & \text{sonst} \end{cases}$$

Monitorkonstruktion

- Automatenmodell für fDTL_ω
  - Warten auf dreiwertige Propositionen
  - Einsetzungsfunktion in Next-Operator
- LTL_3-Konstruktion

- $p \in \mathrm{AP}$ atomare Proposition, $r \in \mathrm{CP}$ dreiwertige Proposition, $a \in \Sigma$ Zeichen
- Transitionsfunktion $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$

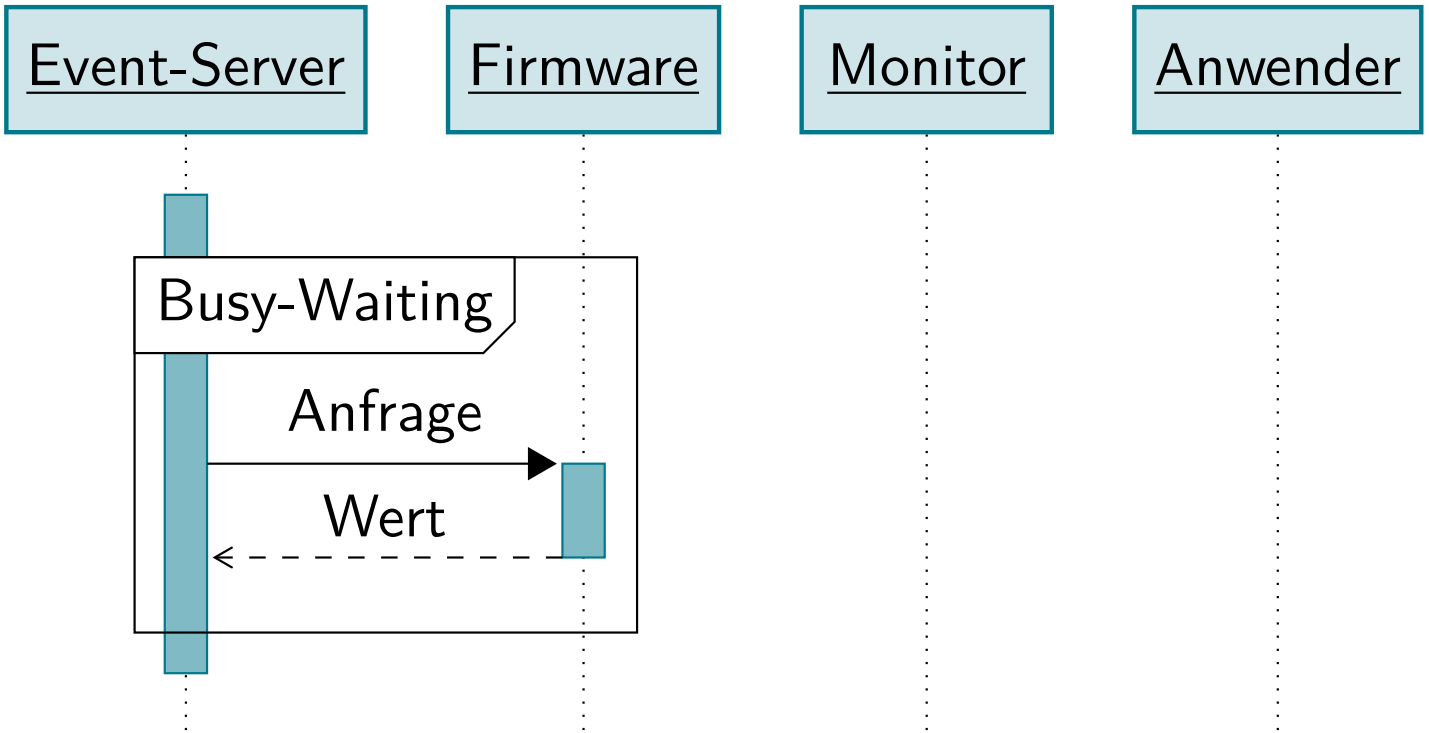
$$\delta(p, a) = \begin{cases} \mathrm{true} & \text{wenn } a(p) = \top \\ \mathrm{false} & \text{sonst} \end{cases}$$

$$\delta(q, a) = \begin{cases} \mathrm{true} & \text{wenn } a(r) = \top \\ r & \text{wenn } a(r) = ? \\ \mathrm{false} & \text{sonst} \end{cases}$$

$$\delta(\bigcirc \varphi, a) = \mathrm{repl}(\varphi, a).$$

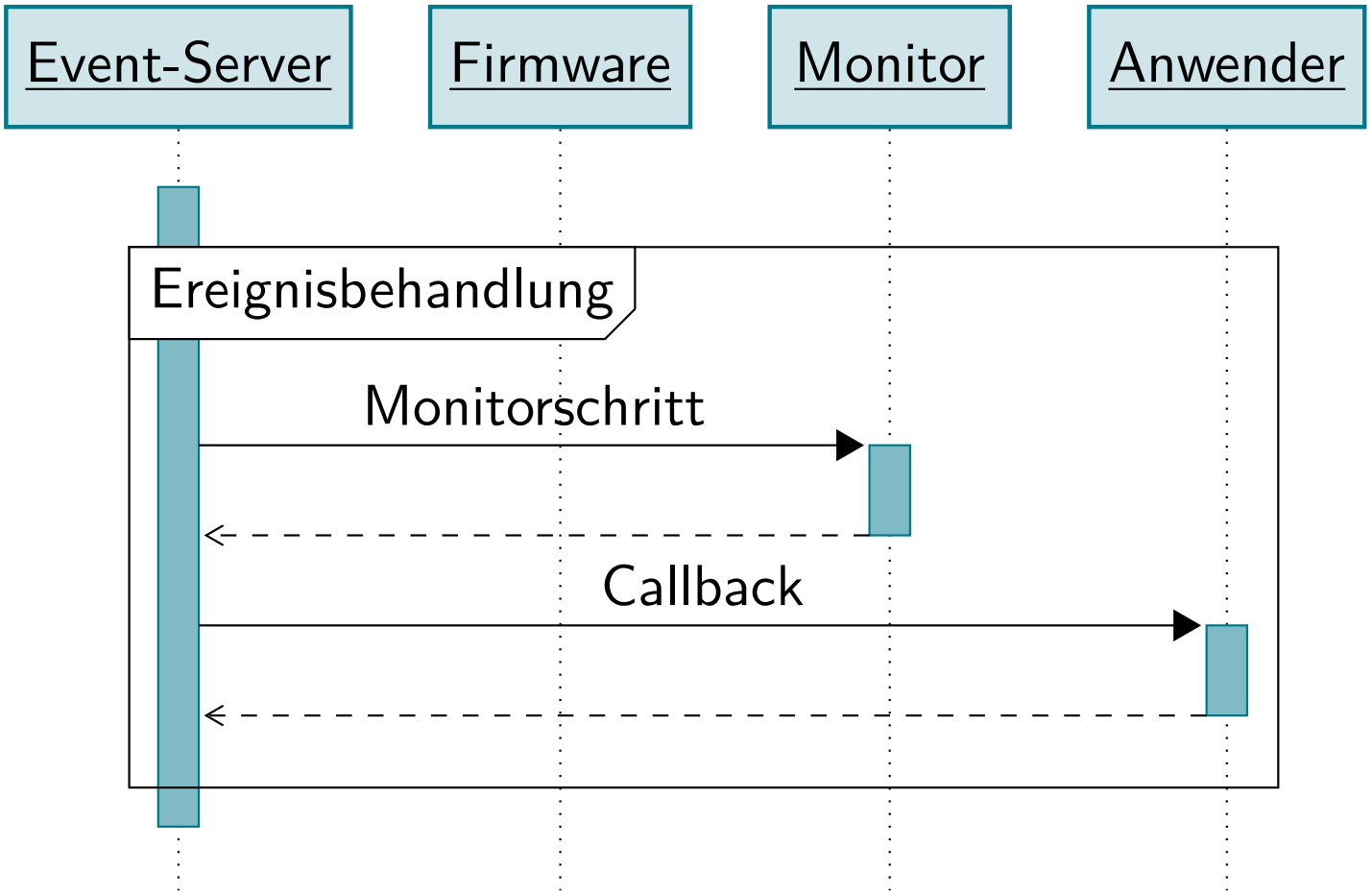
Event-Server zur Ereignisbehandlung

UML-Sequenzdiagramm des Ablaufs auf dem NXT



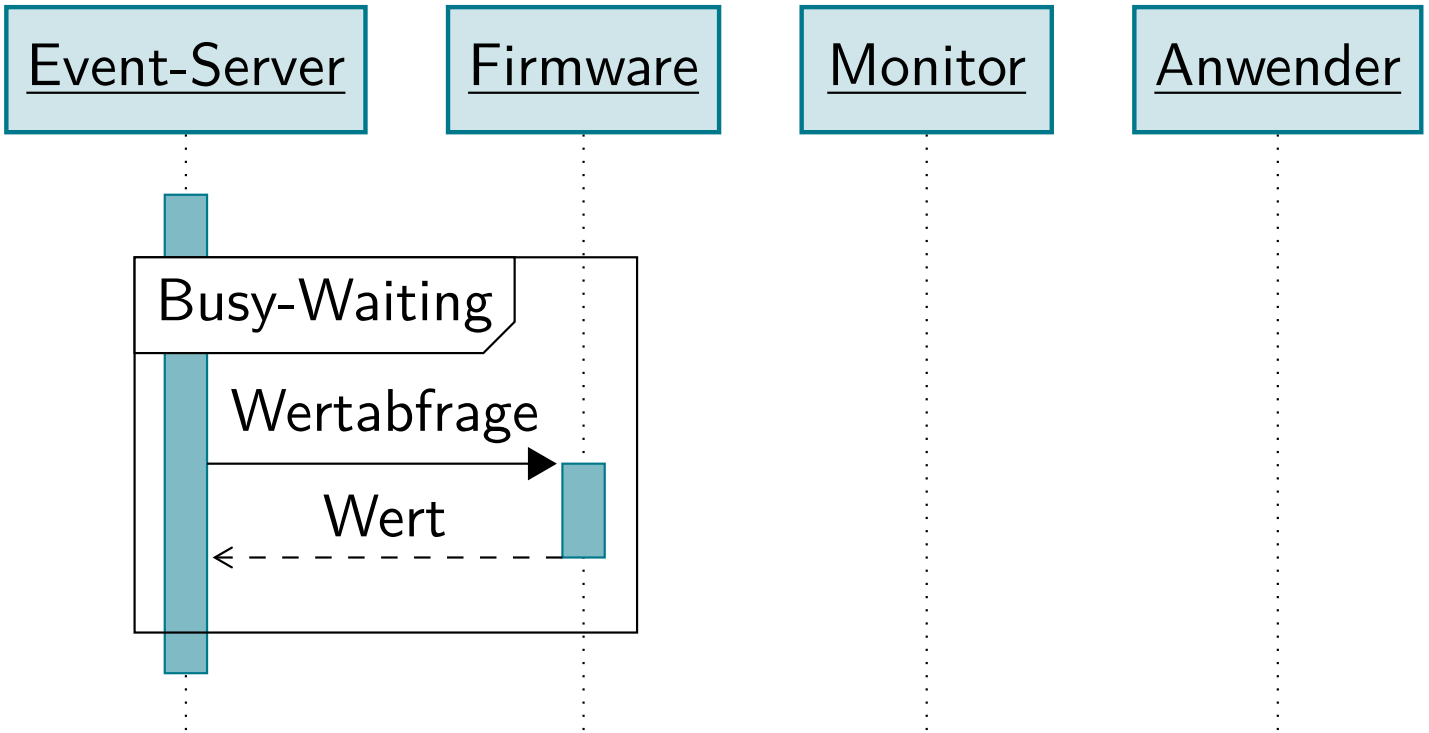
Event-Server zur Ereignisbehandlung

UML-Sequenzdiagramm des Ablaufs auf dem NXT



Event-Server zur Ereignisbehandlung

UML-Sequenzdiagramm des Ablaufs auf dem NXT

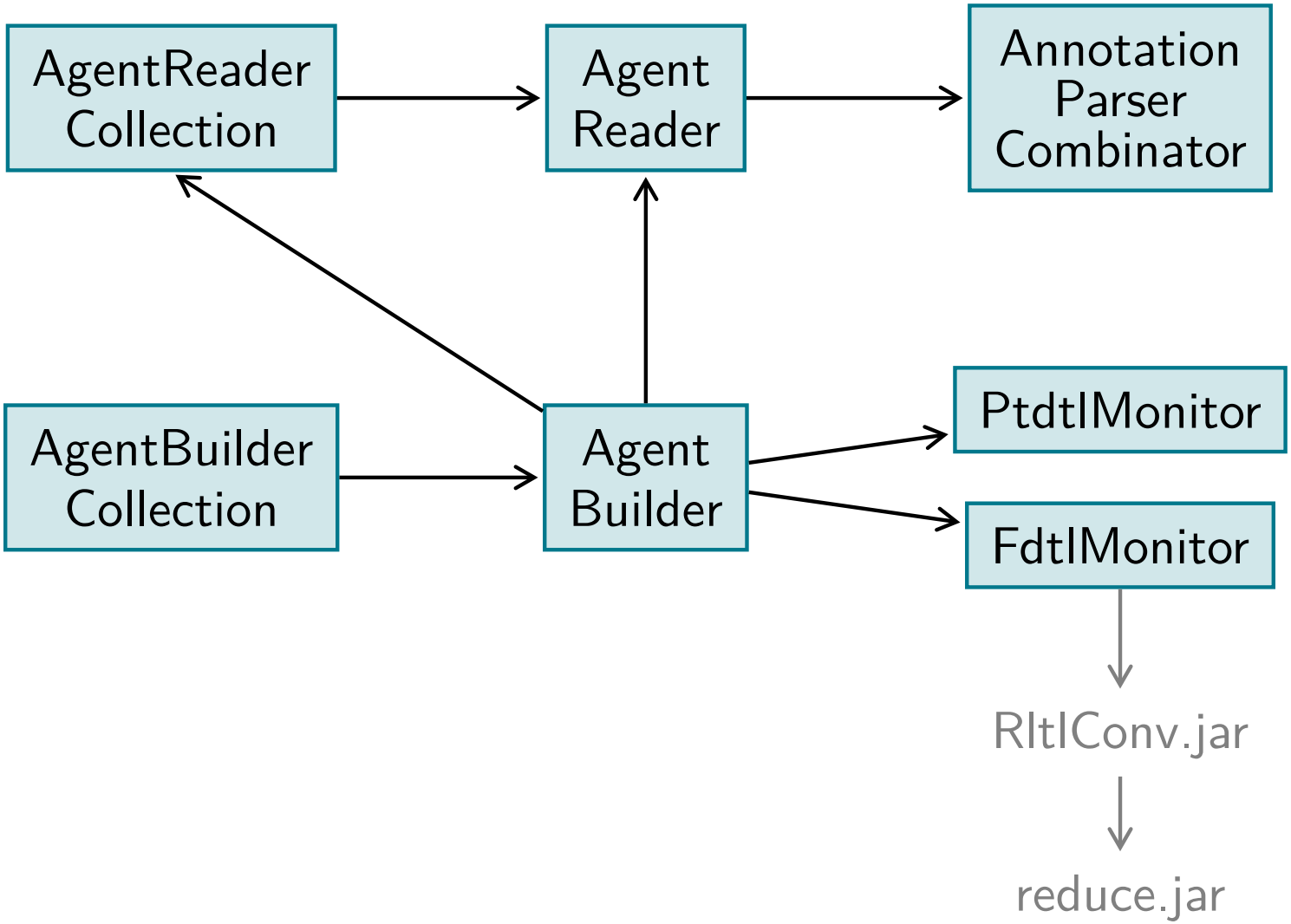


Architektur der Monitorgenerierung

UML-Klassendiagramm

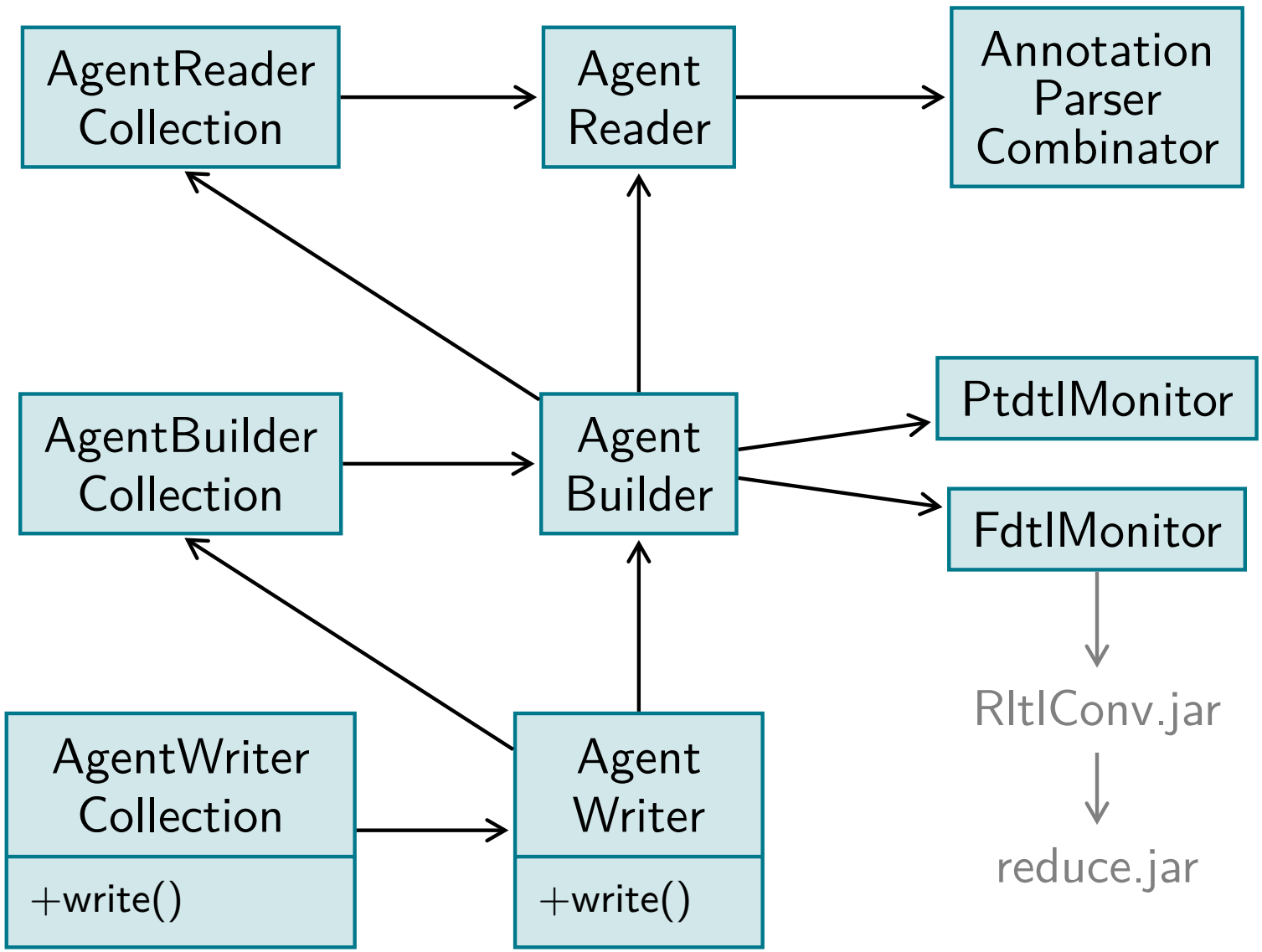
Architektur der Monitorgenerierung

UML-Klassendiagramm



Architektur der Monitorgenerierung

UML-Klassendiagramm



Monitore

```
//= AGENT receiver
//= MONITOR m FDTL=G(light -> active)
//= PROPOSITION active EXTERNAL sender
```

receiver.nxc → receiver_out.nxc

```
//= AGENT sender
//= PUBLIC MONITOR active PTDTL=go S ready
```

sender.nxc → sender_out.nxc

Explizite Umschaltung

```
//=PROPOSITION button
// ...
while(true) {
  if (Sensor(IN_1) == 1) {
    //=ON button
    // ...
    until (Sensor(IN_1) == 0);
    //=OFF button
    // ...
  }
}
```

Explizite C-Funktion

```
//=PROPOSITION button
//=    DEFINE (Sensor(IN_1) == 1)
// ...
while(true) {
  if (Sensor(IN_1) == 1) {
    // ...
    until (Sensor(IN_1) == 0);
    // ...
  }
}
```

Programmtransformation

```
//=PROPOSITION running
//=     ON /OnFwd\(OUT_A[^)]+\);/
//=     OFF /Off\(OUT_A\);/
//...
OnFwd(OUT_A);
Wait(5000);
Off(OUT_A);
```

Expliziter Schritt

```
while(true) {
  if (Sensor(IN_1) == 1) {
    //=STEP mon
    // ...
    until (Sensor(IN_1) == 0);
    //=STEP mon
    // ...
  }
}
```

Programmtransformation

```
//=EVENT mon ON /value = [0-9]+;/
```

Bei Änderung einer Proposition

```
//=EVENT mon CHANGE light
```

Bei Änderung einer Proposition

```
//=EVENT mon CHANGE light
```

Achtung

- Doppeltes Busy-Waiting bei expliziter C-Funktion
- Reihenfolge nicht deterministisch
- Lösung: Event-Server

```
//=EVENT mon CHANGE light
//=     CALL light_handler
```

Nach fester Zeit

```
//=EVENT mon TIME 100ms
```

Benchmarks: Umlaufzeit einer Nachricht

Hauptformel: ptLTL, fDTL / **entfernte Formel:** ptLTL, fDTL
neue Hauptformel: ab 4 / atomare Propositionen: 2