Synthese von Zustandsautomaten aus Live Sequence Charts

Héctor Díez-Cubeiro

Betreuer: Carsten Kern

Institut für Softwaremodellierung und Verifikation
RWTH Aachen

24. April 2007

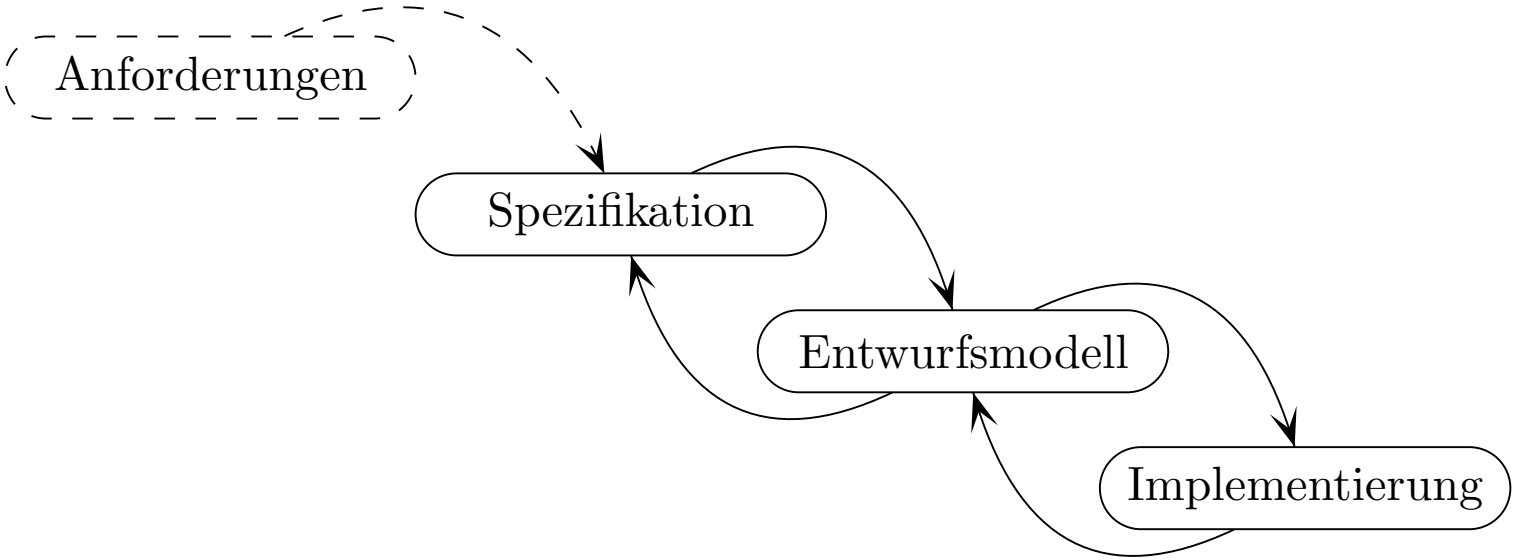
Gliederung

Gliederung

Reaktive Systeme

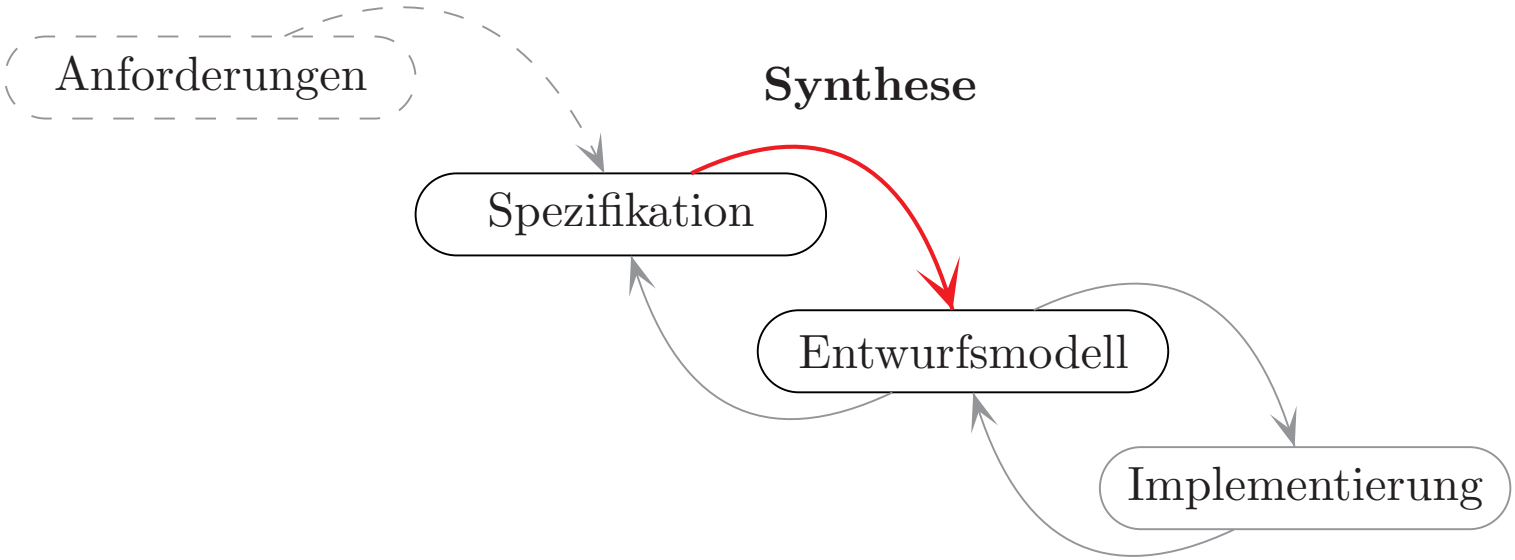
- Verteilte computerbasierte Systeme
- Systemkomponenten in ständiger Interaktion miteinander und mit ihrer Umgebung
- Keine Terminierung der Komponenten
- In sicherheitskritischen Anwendungen eingesetzt

Bisheriger Entwicklungsprozess



- Manuell, daher fehleranfällig und hohe Entwicklungskosten
- Bedarf nach einem automatisierten Prozess
- Heutige Werkzeuge oft zu spezialisiert
  - Bsp.: Automobilindustrie

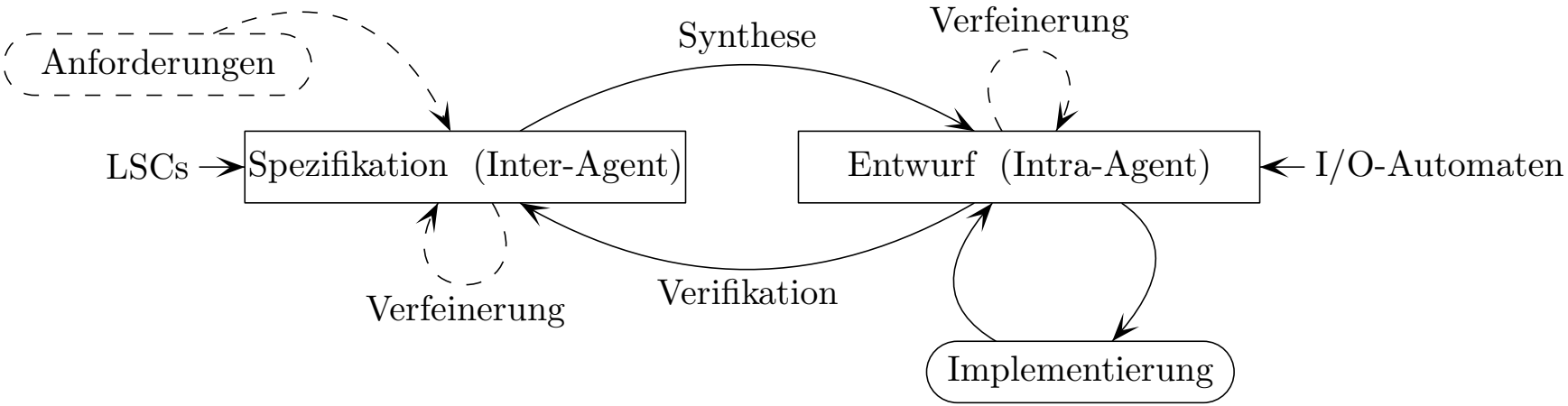
Bisheriger Entwicklungsprozess



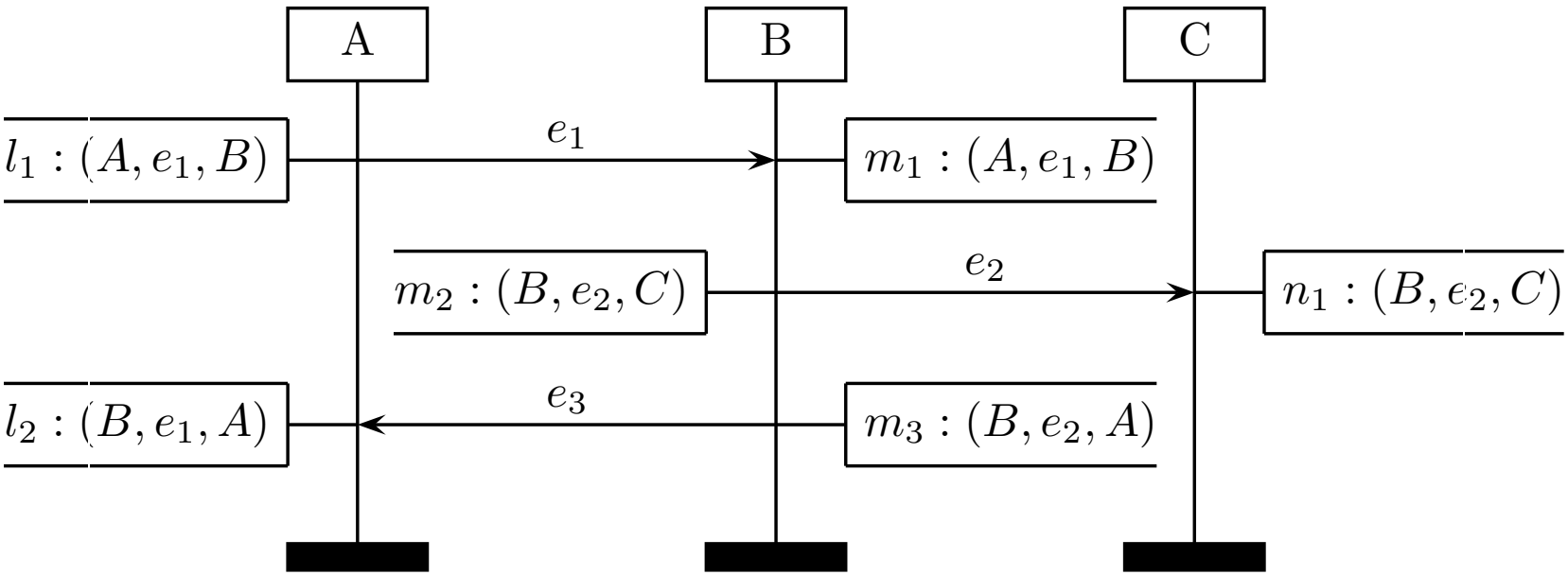
- Das Ziel: Automatisierung der Synthese

Gliederung

Vollständiger Entwicklungsprozess



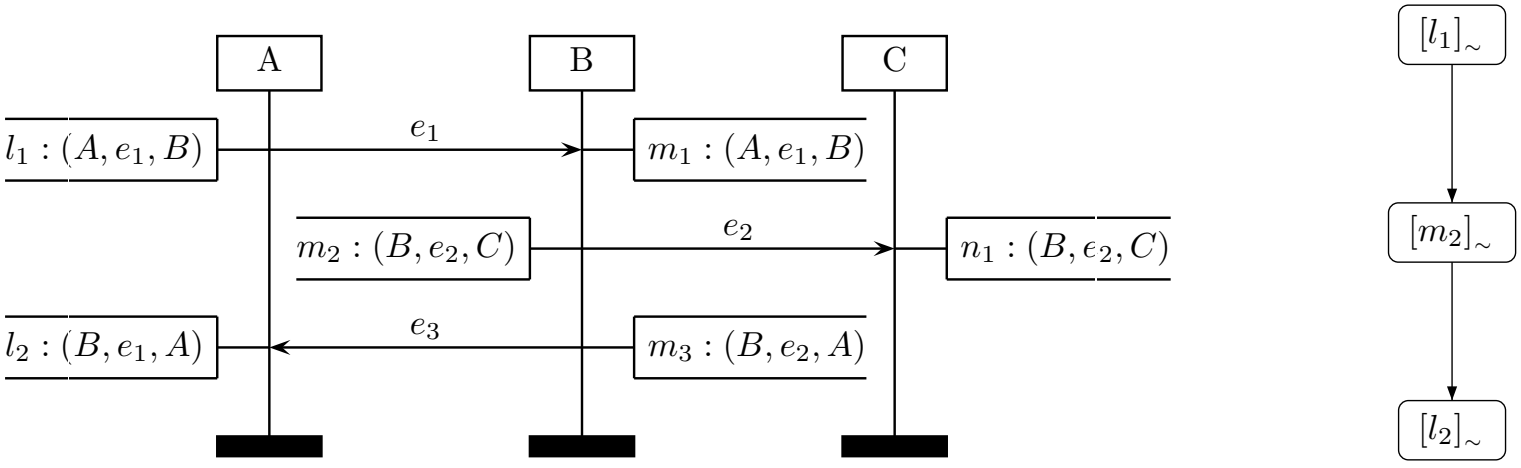
Message Sequence Charts (MSC)



Message Sequence Charts (MSC)

- Ag: endliche Menge von Agenten
- $\mathcal{M}$: endliche Menge von Nachrichtenbezeichnern
- $\Sigma = Ag \times \mathcal{M} \times Ag$: Menge aller Ereignisse
  - (a, m, b): „a schickt Nachricht m an b“
  - $\mathcal{L}(B) \subseteq \Sigma^*$: „Sprache“ eines Diagramms B
- Ereignisse sind *unmittelbar*

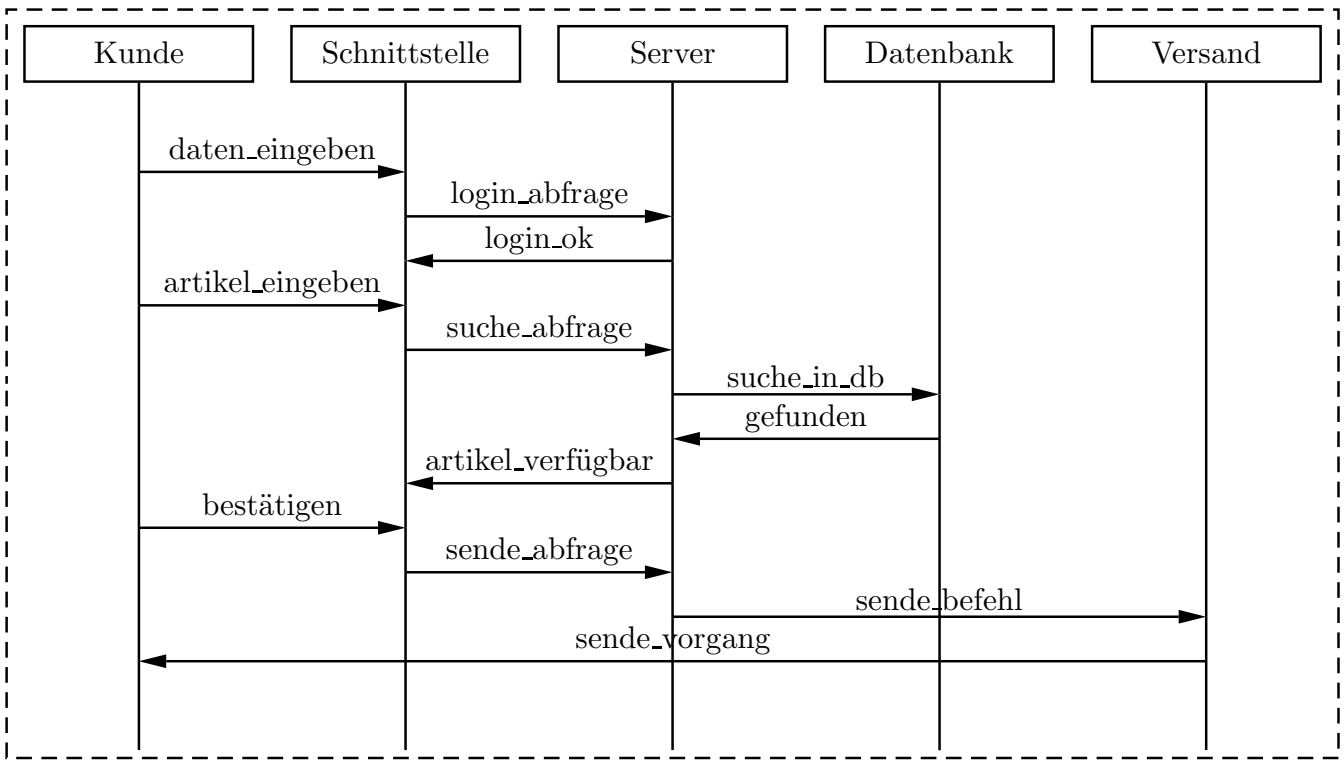
Message Sequence Charts (MSC)



- Zeitgleiche Ereignisse in Äquivalenzklassen zusammengefasst
- Eindeutige Beschriftung von Äquivalenzklassen mit Ereignissen
  $\Rightarrow$ Beschriftete partielle Ordnung $\langle L_C, \prec_C, \phi_C \rangle$
- Grunddiagramme von LSCs

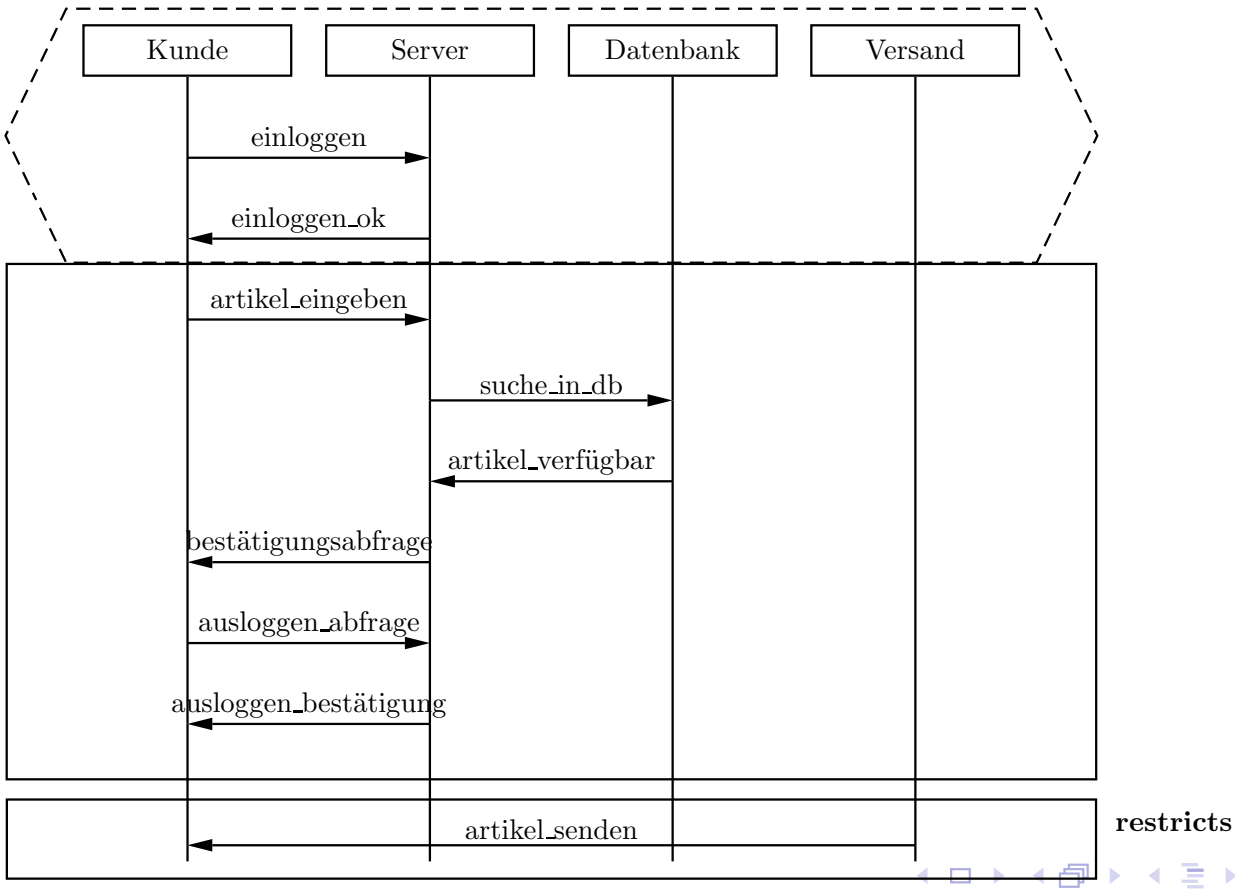
Live Sequence Charts (LSC)
Existentielle LSCs (eLSCs)

- $\Diamond(B, \Sigma_R)$
- Mögliche Ereignissequenzen



Live Sequence Charts (LSC)
Universelle LSCs (uLSCs)

- $\square(P, M, \Sigma_R)$
- Notwendiges Verhalten



Erfüllbarkeit von LSCs

- Linearisierung einer partiellen Ordnung:
  $\forall e, e' \in \Sigma \wedge e \neq e' : (e \prec_C e') \vee (e' \prec_C e)$
  - Intuitiv: Folge von Ereignissen

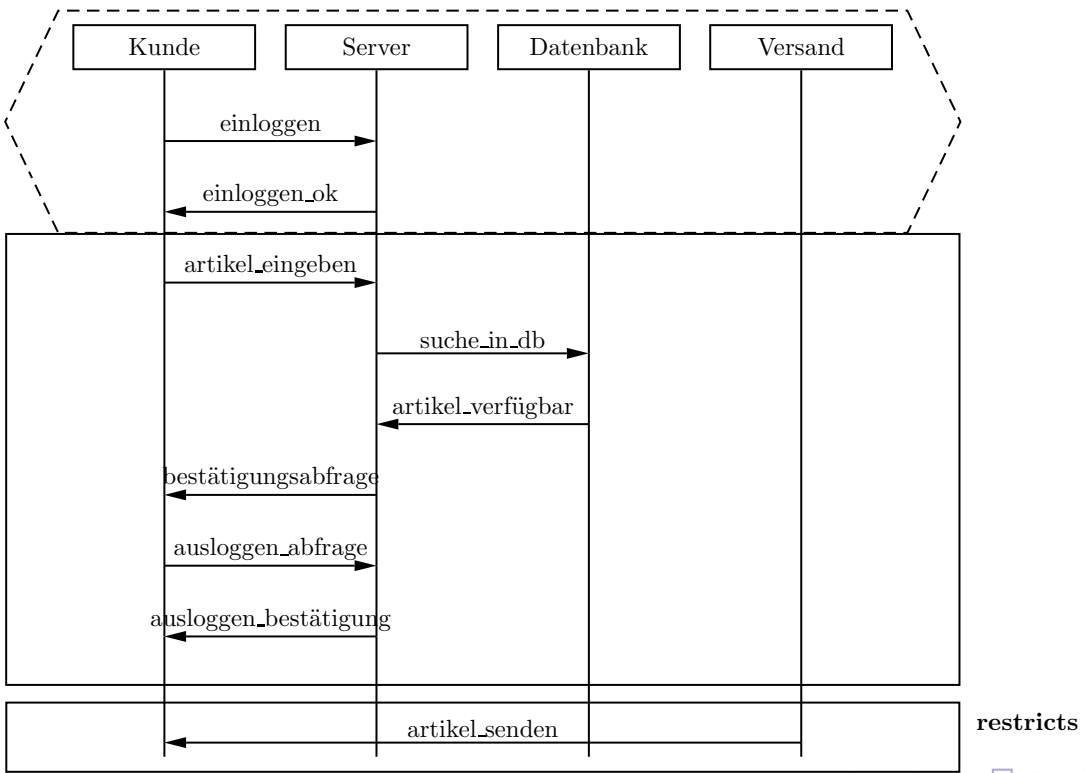
Definition ($\models \subset \Sigma^\omega \times$ LSC)

Sei $\gamma \in \Sigma^\omega$ ein unendlicher Lauf. γ erfüllt ein LSC L ($\gamma \models L$) genau dann, wenn eine der folgenden Bedingungen gilt:

- $L = \Box(P, M, \Sigma_R) \wedge \forall$ Zerlegungen $up\gamma'$ von γ :
  $p|_{\Sigma_R} \in \mathcal{L}(P) \Rightarrow \exists$ Zerlegung $m\gamma''$ von $\gamma' \wedge m|_{\Sigma_R} \in \mathcal{L}(M)$
- $L = \Diamond(B, \Sigma_R) \wedge \exists$ Zerlegung $uv\gamma'$ von $\gamma : v|_{\Sigma_R} \in \mathcal{L}(P)$

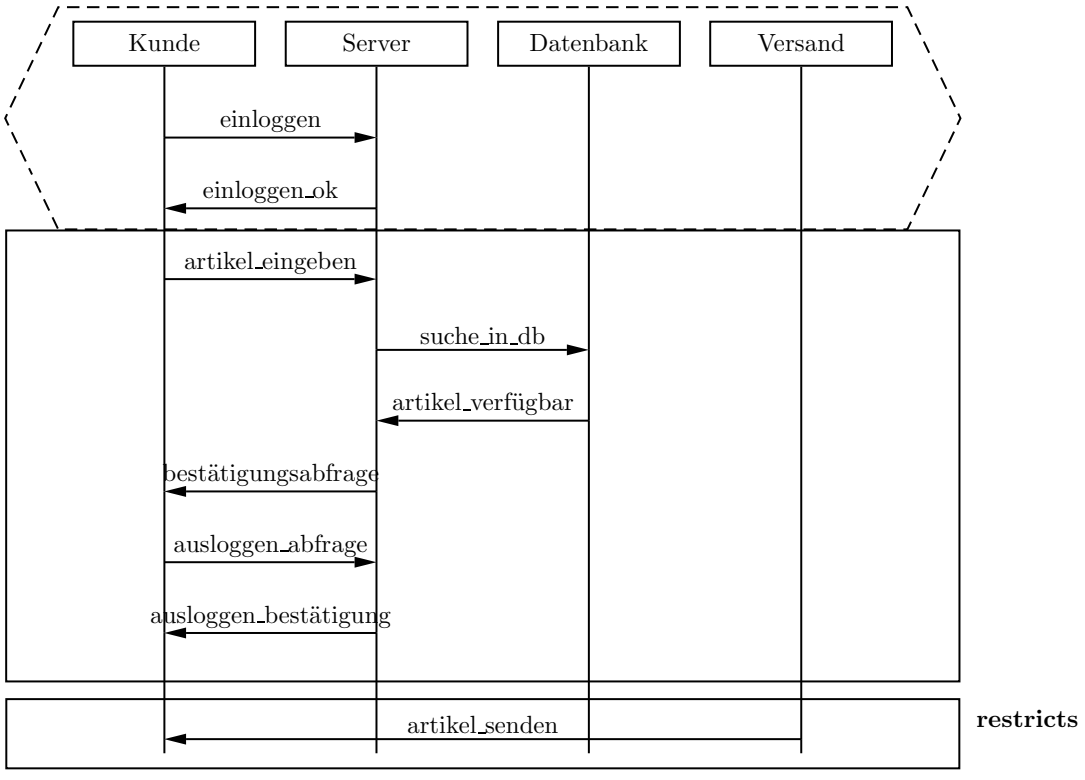
Erfüllbarkeit von LSCs
Beispiel

1. (einloggen·einloggen_ok·artikel_eingeben· suche_in_db·artikel_verfügbar·bestätigungsabfrage· ausloggen_abfrage· ausloggen_bestätigung)$^{\omega}$



Erfüllbarkeit von LSCs
Beispiel

2. $(\text{einloggen} \cdot \text{einloggen\_fehlgeschlagen})^{\omega}$



Erfüllbarkeit von LSCs

Definition ($\mathcal{L}(L)$)

Ein LSC definiert die Sprache $\mathcal{L}(L) = \{\gamma \in \Sigma^\omega | \gamma \models L\}$.

Definition ($\Gamma \models L$)

Eine Menge von Läufen $\Gamma \subseteq \Sigma^\omega$ erfüllt einen LSC L ($\Gamma \models L$) genau dann, wenn

- $L = \Box(P, M, \Sigma_R) \wedge \Gamma \subseteq \mathcal{L}(L)$ oder
- $L = \Diamond(B, \Sigma_R) \wedge \Gamma \cap \mathcal{L}(L) \neq \emptyset$.

Erfüllbarkeit von LSCs

Definition ($\mathcal{L}(L)$)

Ein LSC definiert die Sprache $\mathcal{L}(L) = \{\gamma \in \Sigma^\omega | \gamma \models L\}$.

Definition ($\Gamma \models L$)

Eine Menge von Läufen $\Gamma \subseteq \Sigma^\omega$ erfüllt einen LSC L ($\Gamma \models L$) genau dann, wenn

- $L = \Box(P, M, \Sigma_R) \wedge \Gamma \subseteq \mathcal{L}(L)$ oder
- $L = \Diamond(B, \Sigma_R) \wedge \Gamma \cap \mathcal{L}(L) \neq \emptyset$.

Komposition von LSCs

- Jeder uLSC entspricht einer Anforderung

Definition (uLSC-Spec)

Eine uLSC-Spezifikation ist eine endliche Menge von uLSCs $L_i (1 \leq i \leq n)$

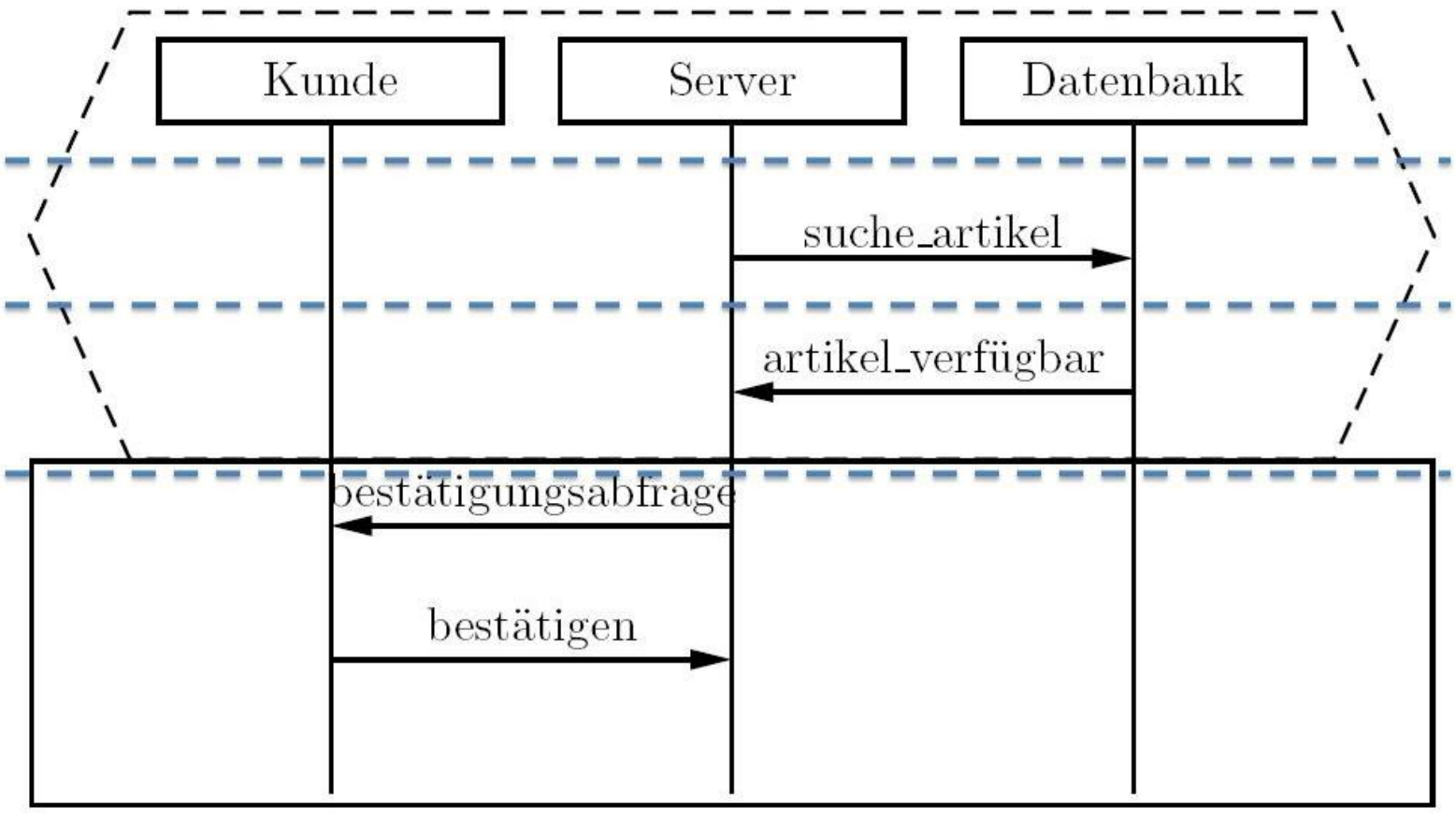
$$\mathcal{L}(\{L_1, \ldots, L_n\}) = \bigcap_{i=1}^{n} \mathcal{L}(L_i).$$

Schnitte

- Zweck: Zustandsbeschreibung von LSCs
- Menge der bereits erreichte Stellen bzw. Äquivalenzklassen
- Nach unten beschränkt
- „Erreichbarkeit“: $c \xrightarrow{l} c' \Leftrightarrow l \notin c \wedge c' = \{l\} \cup c$
- Linearisierung von $\mathfrak{L} = \langle L, <, \phi \rangle$:

$$\emptyset \xrightarrow{l_1} c_1 \xrightarrow{l_2} c_2 \ldots c_{n-1} \xrightarrow{l_n} c_n \Leftrightarrow \phi(l_1) \cdot \ldots \cdot \phi(l_n).$$

Schnitte
Beispiel



- $w =$ suche_artikel $\cdot$ db_leer $\cdot$ artikel_verfügbar

Schnitte

- $gen(w, \mathfrak{L})$ ist die Menge aller von w generierten Schnitte

Definition ($gen(w, \mathfrak{L})$)

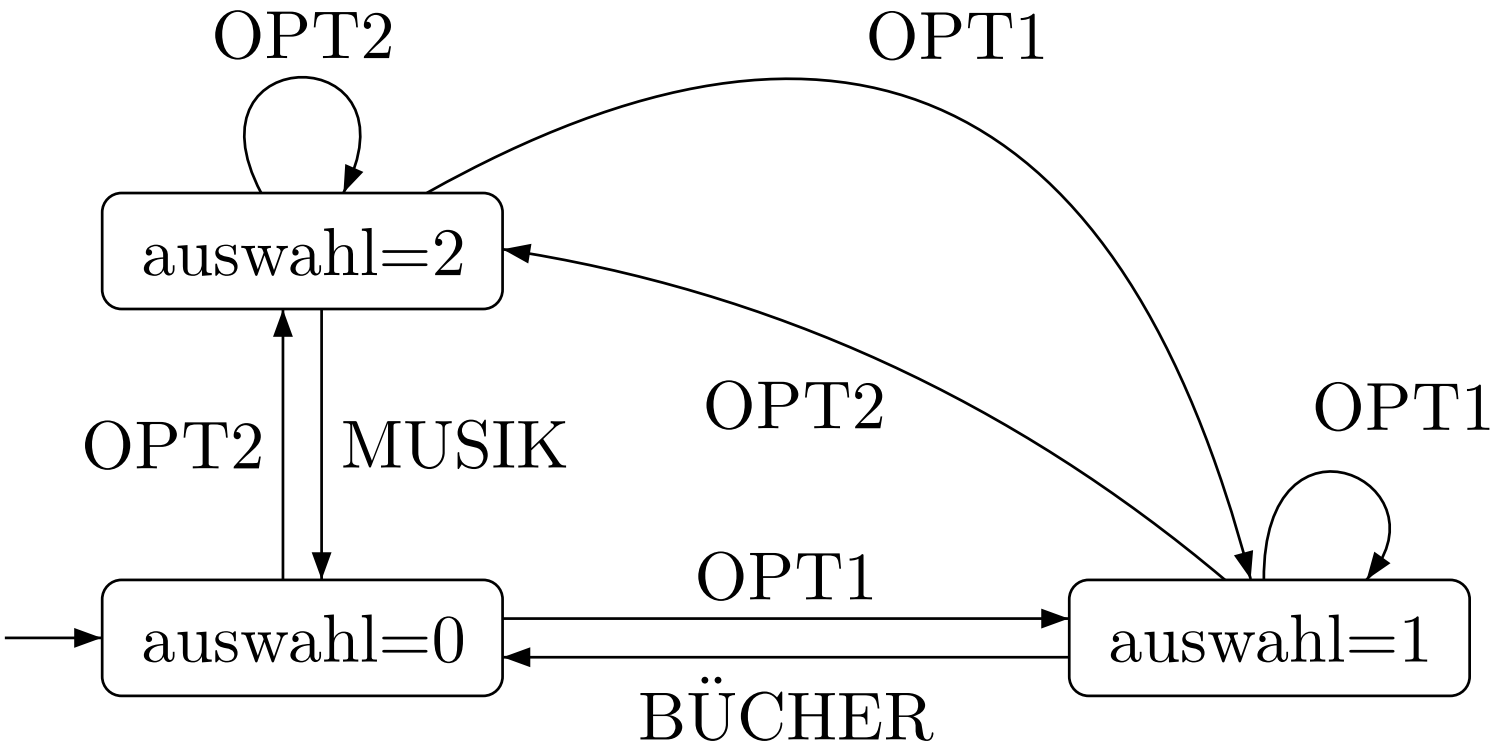
$$gen(\epsilon, \mathfrak{L}) = \{\emptyset\}$$

$$gen(w \cdot a, \mathfrak{L}) = \{\emptyset\} \cup \{c' | \exists c \in gen(w, \mathfrak{L}), l : \phi(l) = a \wedge c \xrightarrow{l} c'\}.$$

- Die Menge aller erzeugbaren Schnitte:
  $Gen(\mathfrak{L}) = \{gen(w, \mathfrak{L}) | w \in \Sigma^*)\}$

I/O-Automaten
Beispiel

- $\Sigma^s_{Env} = \Sigma^r_{Sys} = \{\text{OPT}i \mid 1 \leq i \leq 2\}$
- $\Sigma^r_{Env} = \Sigma^s_{Sys} = \{\text{MUSIK}, \text{BÜCHER}\}$



I/O-Automaten

- Deterministische endliche Automaten
- Unterscheidung zwischen Eingabe- und Ausgabeereignissen
- Keine Blockierung von Eingaben („input-enabledness“)

Modelle

- Statiches Modell: Systemstruktur
  - Statische Information über mögliche Interaktionen zwischen Agenten oder Instanzen
  - Unterscheidung von System- und Umgebungsagenten

Definition (Systemstruktur)

$$\langle Ag, (\Sigma_a^s)_{a \in Ag}, (\Sigma_a^r)_{a \in Ag}, Sys \rangle,$$

- Ag endliche Menge von Agentenbezeichnern.
- Σ_a^s Sendeereignisse.
- Σ_a^r Empfangsereignisse.
- $Sys \subseteq Ag$ Systemagenten.
- $Env \triangleq Ag \setminus Sys$ Umgebungsagenten.

Modelle

- Statiches Modell: Systemstruktur
  - Statische Information über mögliche Interaktionen zwischen Agenten oder Instanzen
  - Unterscheidung von System- und Umgebungsagenten

Definition (Systemstruktur)

$$\langle Ag, (\Sigma_a^s)_{a \in Ag}, (\Sigma_a^r)_{a \in Ag}, Sys \rangle,$$

- Ag endliche Menge von Agentenbezeichnern.
- Σ_a^s Sendeereignisse.
- Σ_a^r Empfangsereignisse.
- $Sys \subseteq Ag$ Systemagenten.
- $Env \triangleq Ag \setminus Sys$ Umgebungsagenten.

Modelle

- Dynamisches Modell: Inter- und Intra-agentes Verhalten
  - Dynamische Information über das Verhalten von Agenten
  - Inter-agent: Spezifikation des Verhaltens *zwischen* Instanzen mithilfe von LSCs
  - Intra-agent: Spezifikation des Verhaltens *einzelner* Instanzen durch I/O-Automaten

Inter-agentes Verhalten

Definition (Inter-agente Spezifikation)

Eine Inter-agente Spezifikation ist ein Paar

$$\langle S, \mathcal{U} \rangle,$$

wobei S eine Systemstruktur und $\mathcal{U}$ eine uLSC-Spezifikation ist.

Intra-agentes Verhalten

- Reaktion von Instanzen abhängig von der Vergangenheit
- Strategie: $f_a : \Sigma^* \to 2^{\Sigma_a^s}$
- Outcome: $Out(f_a) = \{u_0 w_0 u_1 w_1 \ldots | \forall i \geq 0 : u_i \in (\Sigma \setminus \Sigma_a^s)^* \wedge w_i \in f_a(u_0 w_0 \ldots u_i)\}$.

Definition (Intra-agente Spezifikation)

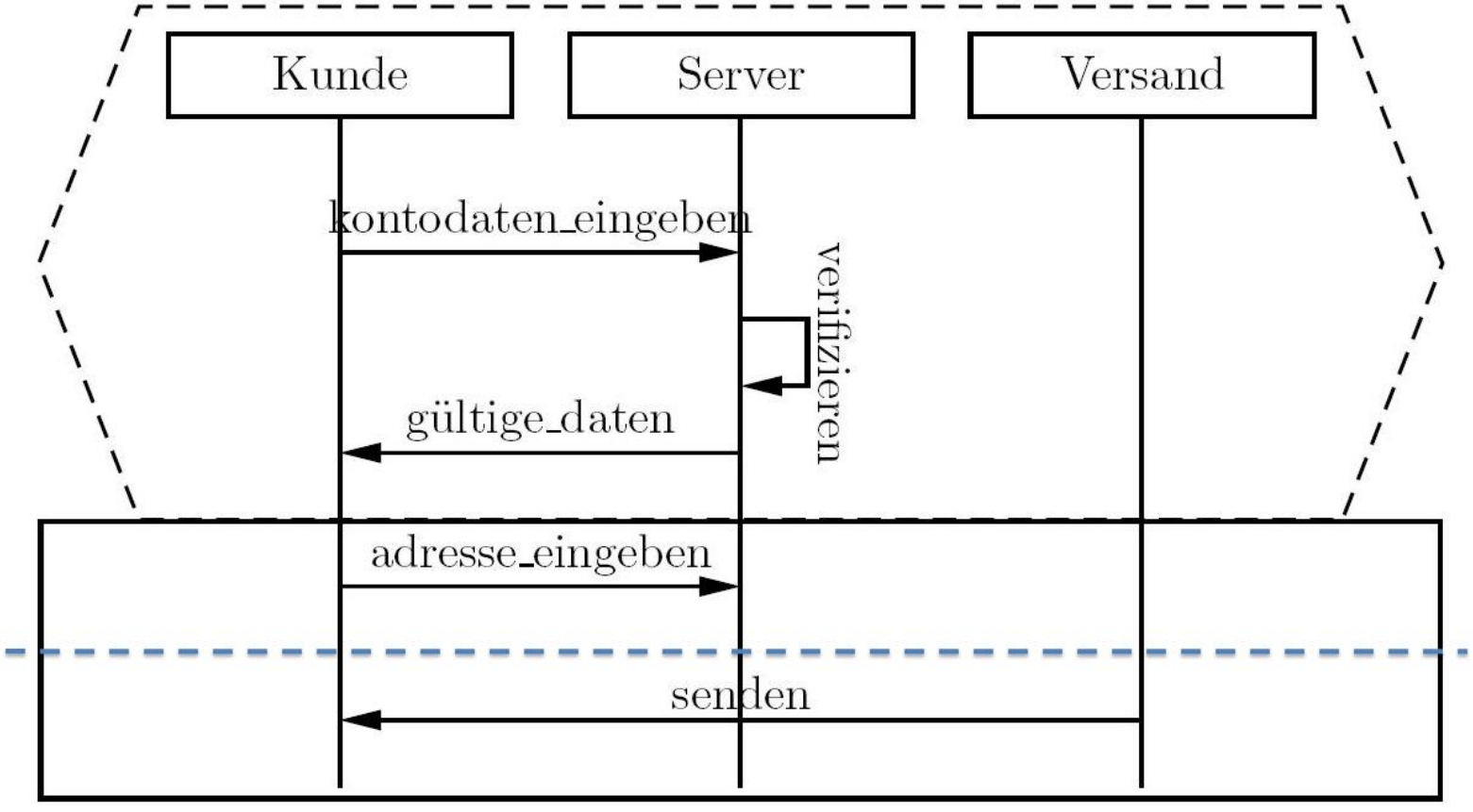
Eine Intra-agente Spezifikation ist ein Paar

$$\langle S, f_{Sys} \rangle,$$

wobei S eine Systemstruktur und f_{Sys} sie Strategie für die Menge Sys ist.

Gliederung

Safety und Liveness
Beispiel



- $w =$ kontodaten_eingeben $\cdot$ verifizieren $\cdot$ gültige_daten $\cdot$ adresse_eingeben

Safety und Liveness

Definition (Verbotenes Ereignis, Safety)

$forbid(w, e) \Leftrightarrow \exists \Box(P, M, \Sigma_R) \in \mathcal{U} \wedge c \in gen(w, P \cdot M)$, wobei:

1. $L_P \subseteq c \subset L_{P \cdot M}$, das Diagramm ist aktiv,
2. e wird von $\Box(P, M, \Sigma_R)$ eingeschränkt,
3. $\forall l \in L : \phi(l) = e \Rightarrow \nexists c' : c \xrightarrow{l} c'$.

Ein Lauf $e_0 e_1 e_2 \ldots \in \Sigma^\omega$ ist *e-sicher*, $\Leftrightarrow$
$\forall i \geq 0 : forbid(e_1 \ldots e_i, e) \Rightarrow e \neq e_{i+1}$.

Safety und Liveness

Definition (Benötigtes Ereignis, Liveness)

$require(w, e) \Leftrightarrow \exists \Box(P, M, \Sigma_R) \in \mathcal{U} \wedge c \in gen(w, P \cdot M)$, wobei:

1. $L_P \subseteq c \subset L_{P \cdot M}$,
2. e wird von $\Box(P, M, \Sigma_R)$ eingeschränkt,
3. $\exists c' : \exists l : (c \xrightarrow{l} c') \wedge e = \phi(l)$.

Ein Lauf $e_0 e_1 e_2 \ldots \in \Sigma^\omega$ ist *e-lebendig*, $\Leftrightarrow$
$\forall i \geq 1 : require(e_1 \ldots e_i, e) \Rightarrow (\exists k : i \leq k : e = e_k)$.

Safety und Liveness

Theorem (uLSC=sicher+lebendig)

Für jedes $\gamma \in \Sigma^\omega$ und jedes uLSC $S = \Box(P, M, \Sigma_R)$ gilt:

$$\gamma \models S \Leftrightarrow \forall e \in \Sigma_R : \gamma \text{ ist e-sicher und e-lebendig.}$$

Defintion (Korrekte Implementierung)

Eine intra-agente Spezifikation $\langle S, f_{Sys} \rangle$ ist eine korrekte Implementierung einer inter-agenten Spezifikation $\langle S, \mathcal{U} \rangle$ genau dann, wenn $\forall \gamma \in Out(f_{Sys})$, folgende Bedingungen gelten:

- γ ist Σ^s_{Env}-lebendig $\Rightarrow \gamma$ ist Σ^s_{Sys}-lebendig.
- γ ist Σ^s_{Env}-sicher $\Rightarrow \gamma$ ist Σ^s_{Sys}-sicher.

Safety und Liveness

Theorem (uLSC=sicher+lebendig)

Für jedes $\gamma \in \Sigma^\omega$ und jedes uLSC $S = \Box(P, M, \Sigma_R)$ gilt:

$$\gamma \models S \Leftrightarrow \forall e \in \Sigma_R : \gamma \text{ ist e-sicher und e-lebendig.}$$

Defintion (Korrekte Implementierung)

Eine intra-agente Spezifikation $\langle S, f_{Sys} \rangle$ ist eine korrekte Implementierung einer inter-agenten Spezifikation $\langle S, \mathcal{U} \rangle$ genau dann, wenn $\forall \gamma \in Out(f_{Sys})$, folgende Bedingungen gelten:

- γ ist Σ^s_{Env}-lebendig $\Rightarrow \gamma$ ist Σ^s_{Sys}-lebendig.
- γ ist Σ^s_{Env}-sicher $\Rightarrow \gamma$ ist Σ^s_{Sys}-sicher.

Das Synthese- oder Konsistenz-Problem

Defintion (Synthese)

- Eingabe:
  - eine inter-agente Spezifikation $\langle S, \mathcal{U} \rangle$,
- Ausgabe:
  - eine intra-agente Spezifikation $\langle S, f_{Sys} \rangle$, die eine korrekte Implementierung von $\langle S, \mathcal{U} \rangle$ ist,
  - `no`, falls es keine korrekte Implementierung für $\langle S, \mathcal{U} \rangle$ gibt.

- Beispiele für inkonsistente uLSC-Spezifikationen
  - Unerfüllbarkeit
  - Deadlock

Das Synthese- oder Konsistenz-Problem

Defintion (Synthese)

- Eingabe:
  - eine inter-agente Spezifikation $\langle S, \mathcal{U} \rangle$,
- Ausgabe:
  - eine intra-agente Spezifikation $\langle S, f_{Sys} \rangle$, die eine korrekte Implementierung von $\langle S, \mathcal{U} \rangle$ ist,
  - `no`, falls es keine korrekte Implementierung für $\langle S, \mathcal{U} \rangle$ gibt.

- Beispiele für inkonsistente uLSC-Spezifikationen
  - Unerfüllbarkeit
  - Deadlock

Lösungsansatz

- Spielbasierter Ansatz
- Ziel: Strategie für das System finden

Lösungsansatz
Transitionssystem

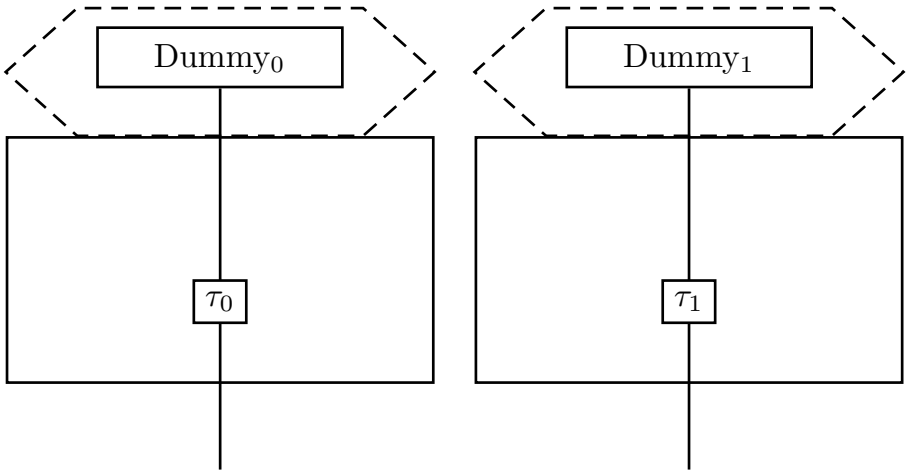
Definition

Sei $\mathcal{U} = \square(P, M, \Sigma_R)$ und $Gen(\mathcal{U}) = Gen(P \cdot M)$. Für jedes $\sigma, \sigma' \in Gen(\mathcal{U})$ und jedes Ereignis $e \in \Sigma$ gilt $\sigma \overset{e}{\Longrightarrow} \sigma'$ genau dann, wenn

- $e \notin \Sigma_R$ und $\sigma = \sigma'$, oder
- $e \in \Sigma_R$ und es gilt
  - $\forall c \in \sigma : L_P \subseteq c \subset L_{P \cdot M} \Rightarrow \exists c' : c \xrightarrow{l} c'$, mit $\phi(l) = e$
  - und $\sigma' = \{\emptyset\} \cup \{c' | \exists c \in \sigma : c \xrightarrow{l} c', \text{mit } \phi(l) = e\}$.

Lösungsansatz
Spielerwechsel

- Einführung von Fairness-Szenarien



- Markierung der Sequenzen mit:
  - τ_0 ($\triangleq$ System, Spieler 0) und
  - τ_1 ($\triangleq$ Umgebung, Spieler 1)
- Nummerierte Ereignisse in Σ_{Sys} und Σ_{Env}
  - τ_0 ist das $|\Sigma_{Sys}| + 1$-te Ereignis
  - τ_1 ist das $|\Sigma_{Env}| + 1$-te Ereignis

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Spielgraph

Definition

Spielgraph $G_{\mathcal{U}} = \langle V, V_0, \Delta, \Omega \rangle$

- $V = (\{0,1\} \times \Sigma \times Gen(U_1) \times \ldots \times Gen(U_n) \times [|\Sigma_{Sys}| + 1] \times [|\Sigma_{Env}| + 1]) \cup \{sink_0, sink_1\}$, Menge der Knoten.
- Knoten haben die Form $(i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)$.
- $V_0 = \{sink_1\} \cup \{(0, e, \sigma_1, \ldots, \sigma_n, c_0, c_1)\}$.
- Transitionsrelation $\Delta \subset V \times V$:
  1. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), (i', e', \sigma'_1, \ldots, \sigma'_n, c'_0, c'_1))$.
  2. $\Delta((i, e, \sigma_1, \ldots, \sigma_n, c_0, c_1), sink_i)$.
  3. $\Delta(sink_i, sink_i)$.
- $\Omega \subseteq V^\omega$, Gewinnläufe für Spieler 0.
  - Streett-Akzeptanzbedingung, $\Omega = \text{Streett}(\{(E, F)\})$ mit
    - $E = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_1 = |\Sigma_{Env}| + 1\} \cup \{sink_0\}$.
    - $F = \{(i, e, \sigma_1 \ldots \sigma_n, c_0, c_1) | c_0 = |\Sigma_{Sys}| + 1\} \cup \{sink_1\}$.

Lösungsansatz
Paritätsspiel

- Eine inter-agente Spezifikation $\langle S, U \rangle$ ist konsistent $\Leftrightarrow$

  es existiert für alle $e \in \Sigma$ eine Gewinnstrategie f vom Startzustand $(1, e, \{\emptyset\}, \ldots, \{\emptyset\}, 1, 1)$ in $G_{\mathcal{U}}$ $\Leftrightarrow$

  eine intra-agente Spezifikation $\langle S, f_{Sys} \rangle$ ist eine korrekte Implementierung von $\langle S, U \rangle$

- Umwandlung in ein Paritätsspiel $\Rightarrow$ Gewinnstrategie

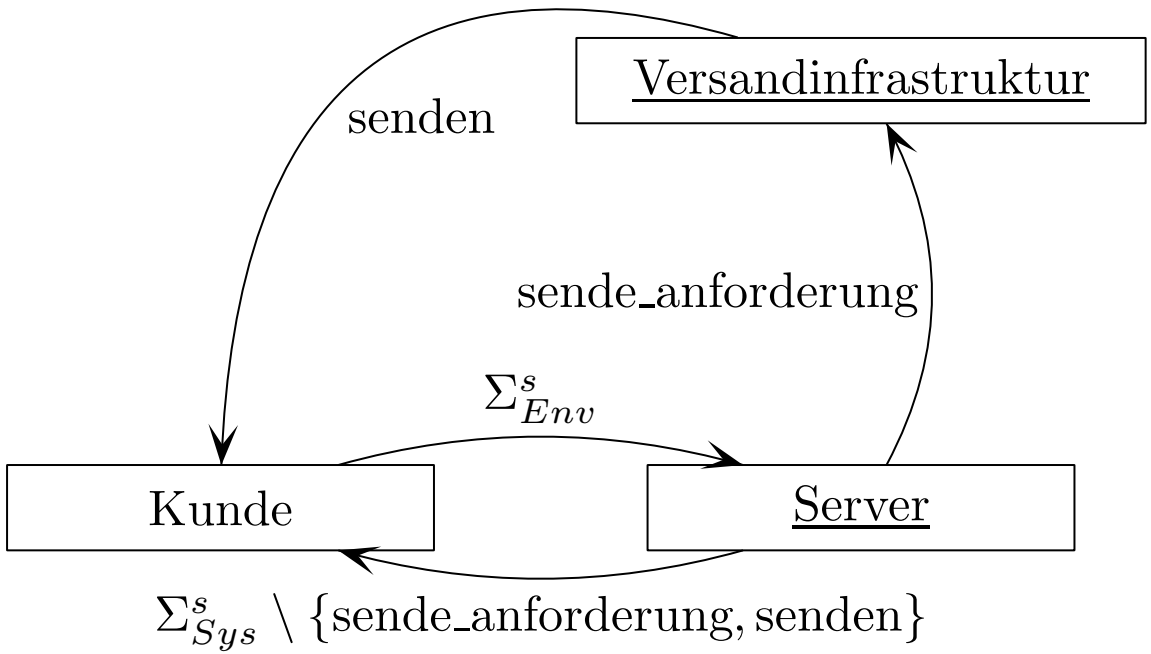
Lösungsansatz
Paritätsspiel

- Graph mit einer Abbildung $\Omega : V \to [k]$ (Färbung)
- Gewählte Färbung:
  - $\Omega(v) = 2$ falls $v \in F$,
  - $\Omega(v) = 1$ falls $v \in E \setminus F$,
  - $\Omega(v) = 0$ sonst.

Lösungsansatz

Beispiel (Inter-agente Spezifikation $\langle \mathcal{S}, \mathcal{U} \rangle$)

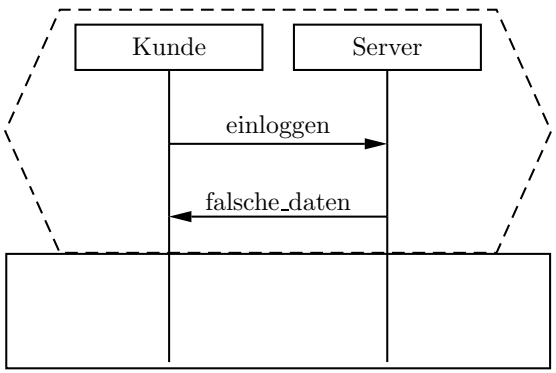
- Systemstruktur $\mathcal{S}$
  - $\Sigma^s_{Env} = \{$einloggen, artikel_suchen, bestellen$\}$
  - $\Sigma^s_{Sys} = \{$einloggen_ok, falsche_daten, artikel_verfügbar sende_anforderung, senden$\}$

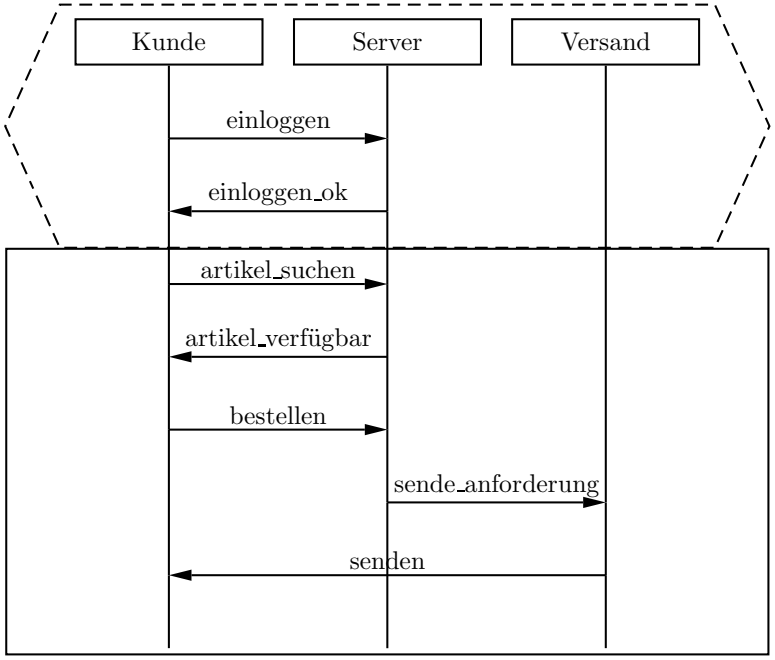


Lösungsansatz

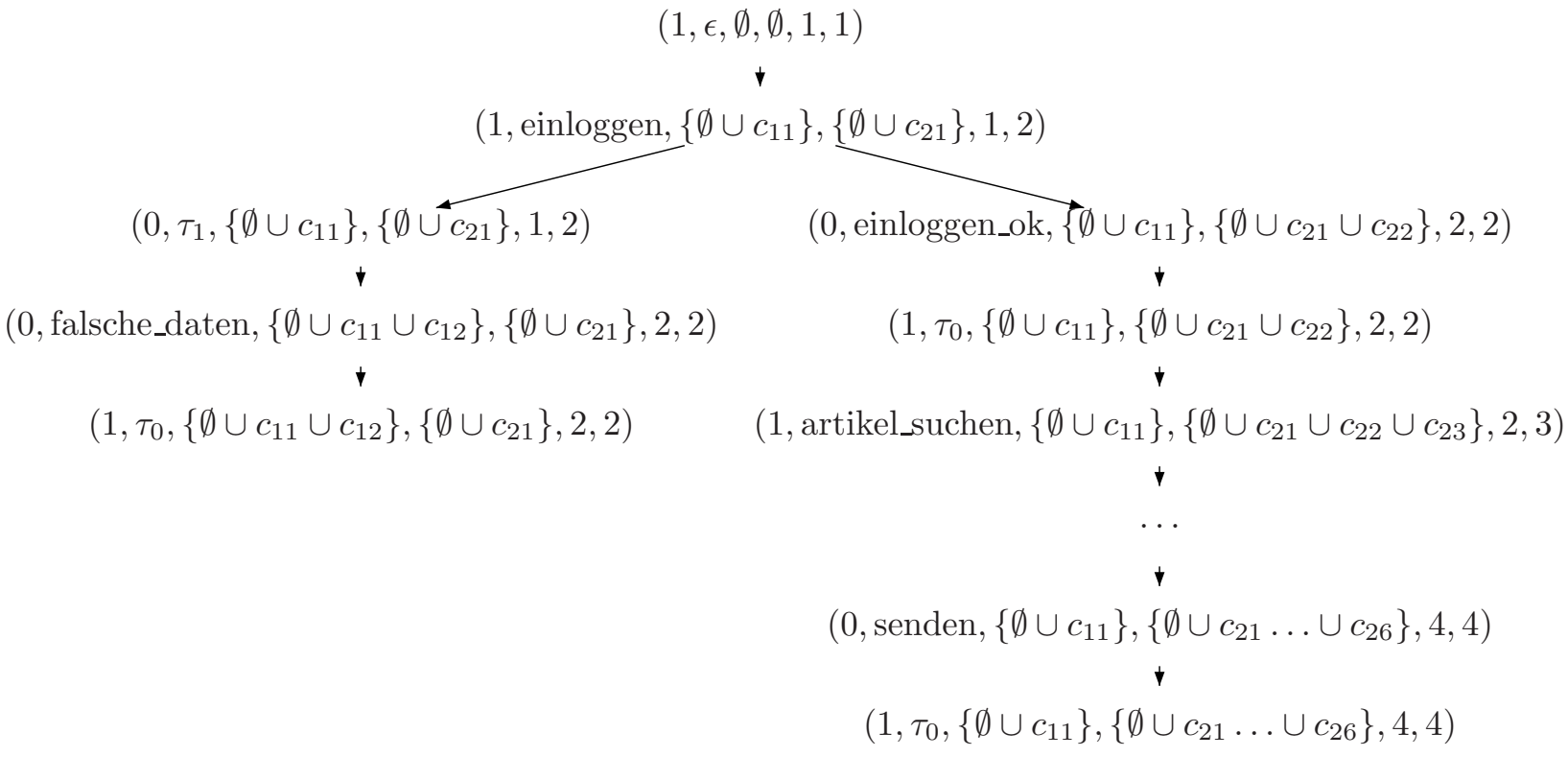
Beispiel (Inter-agente Spezifikation $\langle \mathcal{S}, \mathcal{U} \rangle$)

- uLSC-Spec $\mathcal{U} = \mathcal{L}(U_1) \cap \mathcal{L}(U_2)$

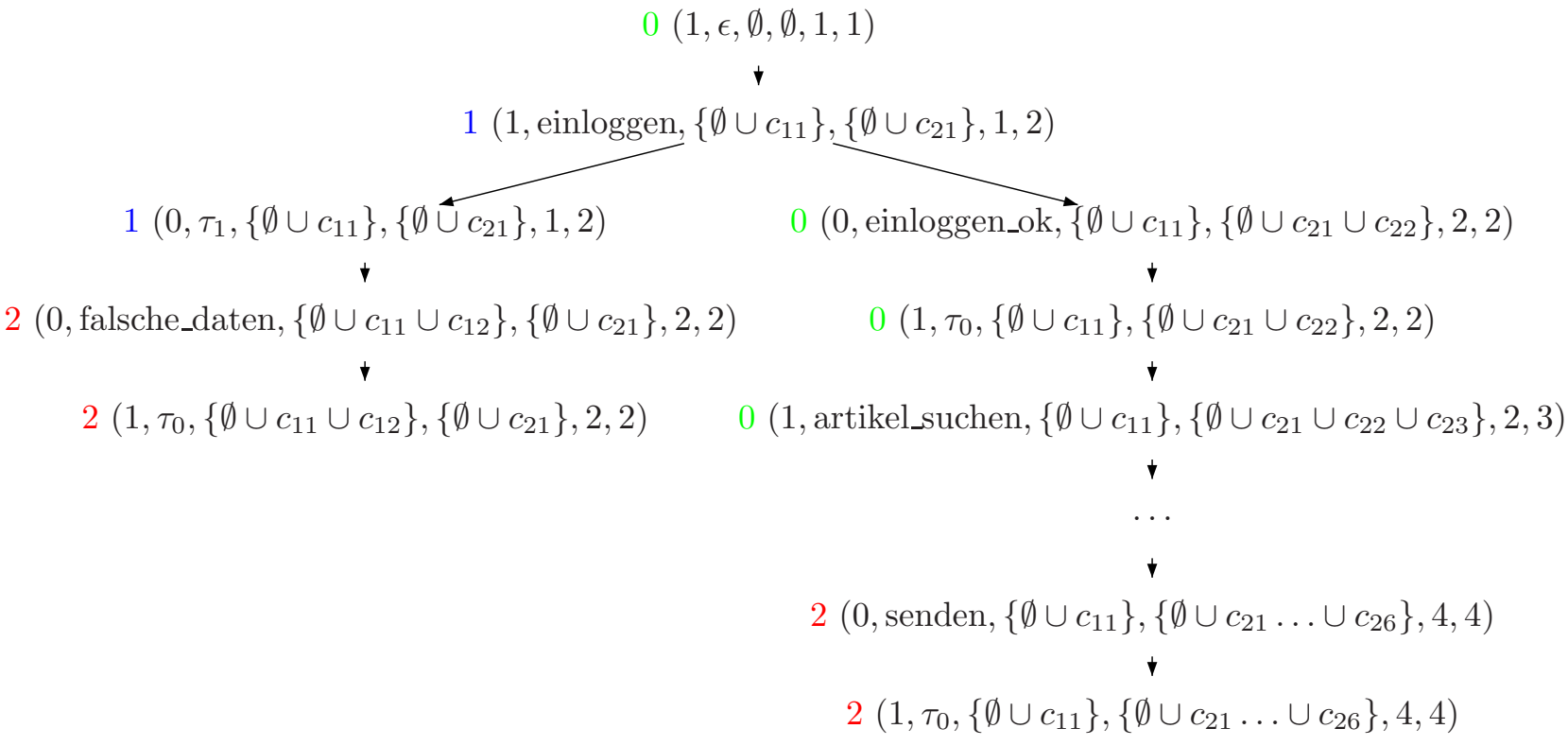




Lösungsansatz
Beispiel (Spielgraph und Paritätsspiel)

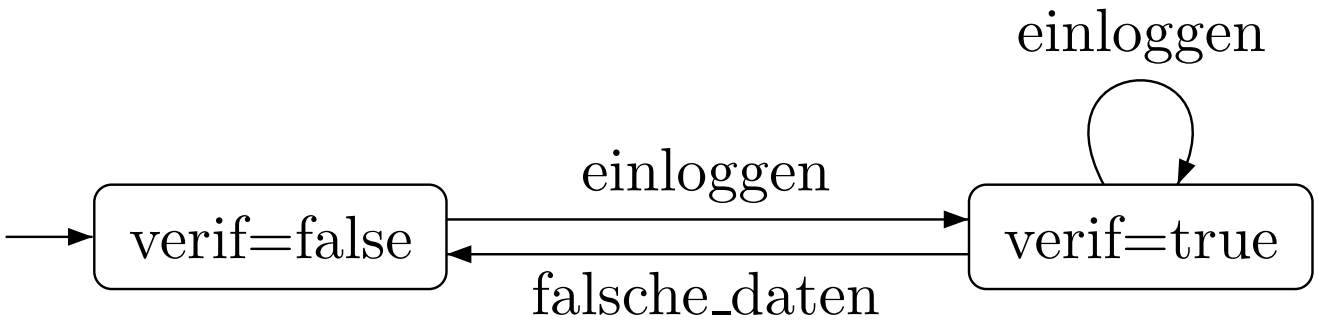


Lösungsansatz
Beispiel (Spielgraph und Paritätsspiel)



Lösungsansatz
Beispiel (I/O-Automat)

- I/O-Automat für die „linke “ Gewinnstrategie



Gliederung

Zusammenfassung

- Ersatz von MSCs durch LSCs zur Anforderungsspezifikation
- Synthese von Zustandsautomaten mithilfe eines spieltheoretischen Ansatzes

Danke für Ihre Aufmerksamkeit!