

# Process Algebra CSP – Practice

---

Hui Shi

27. Juni 2002



---

# Contents

- Summary
  - CSP Syntax
  - Operational Semantics
  - Denotational Semantics
- A Case Study
  - NFS – Network File System
  - Specification with CSP
  - Verification with FDR

---

# Summary

# CSP Syntax

## (some important processes and operators)

- Stop, Skip
- Fundamental operators
  - Prefixing  $e \rightarrow P$
  - Recursion  $P = F(P)$  oder  $\mu p.F(p)$
  - External choice  $P \square Q$
  - Internal choice  $P \sqcap Q$

- Parallel operators
  - Interleaving  $P \parallel\parallel Q$
  - Generalized parallel  $P \underset{A}{\parallel} Q$
- Hiding  $P \setminus A$
- Sequential composition  $P; Q$

# Operational Semantics

- Labelled Transition System (LTD)
  - a set of nodes as process states
  - a starting node  $n_0$
  - for each  $a \in \Sigma^{\checkmark, \tau}$ , a relation  $\xrightarrow{a}$  between nodes, where
$$\Sigma^{\checkmark, \tau} = \Sigma \cup \{\checkmark, \tau\}$$
 $\Sigma$  is the alphabet of all communications of a process

- Firing rules for CSP processes (examples)

$\Omega$  denotes any terminated process

$$\frac{}{Skip \xrightarrow{\checkmark} \Omega}$$

$$\frac{}{e \rightarrow P \xrightarrow{a} subs(a, e, P)} \quad (a \in comms(e))$$

$$\frac{}{P \sqcap Q \xrightarrow{\tau} P}$$

$$\frac{}{P \sqcap Q \xrightarrow{\tau} Q}$$

$$\frac{P \xrightarrow{\tau} P'}{P \sqcap Q \xrightarrow{\tau} P' \sqcap Q}$$

$$\frac{Q \xrightarrow{\tau} Q'}{P \sqcap Q \xrightarrow{\tau} P \sqcap Q'}$$

$$\frac{P \xrightarrow{a} P'}{P \sqcap Q \xrightarrow{a} P'} \quad (a \neq \tau)$$

$$\frac{Q \xrightarrow{a} Q'}{P \sqcap Q \xrightarrow{a} Q'} \quad (a \neq \tau)$$

$$\frac{P \xrightarrow{a} P'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q} \quad (a \in \Sigma \setminus X)$$

$$\frac{Q \xrightarrow{a} Q'}{P \parallel_X Q \xrightarrow{a} P \parallel_X Q'} \quad (a \in \Sigma \setminus X)$$

$$\frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q'} \quad (a \in X)$$

$$\frac{P \xrightarrow{x} P'}{P \setminus B \xrightarrow{x} P' \setminus B} \quad (x \notin B \cup \{\checkmark\})$$

$$\frac{P \xrightarrow{\checkmark} P'}{P \setminus B \xrightarrow{\checkmark} \Omega}$$

$$\frac{P \xrightarrow{a} P'}{P \setminus B \xrightarrow{\tau} P' \setminus B} \quad (a \in B)$$

$$\frac{P \xrightarrow{x} P'}{P; Q \xrightarrow{x} P'; Q} \quad (x \neq \checkmark)$$

$$\frac{\exists P'. P \xrightarrow{\checkmark} P'}{P; Q \xrightarrow{\tau} Q}$$

# Denotational Semantics

- The traces model ( $\mathcal{T}$ )

- Definition

A **trace** of process  $P$  is a sequence of communications between the environment and  $P$ .

For any process  $P$ , **traces**( $P$ ) is defined to be the set of all its finite traces, i.e., members of  $\Sigma^*$ .

- Rules for working out  $traces(P)$  (examples)

$$traces(Stop) = \{\langle \rangle\}$$

$$traces(Skip) = \{\langle \rangle, \langle \checkmark \rangle\}$$

$$traces(e \rightarrow P) = \{\langle \rangle\}$$

$$\cup \{\langle a \rangle^s \mid a \in comms(e) \wedge s \in traces(subs(a, e, P))\}$$

$$traces(P \square Q) = traces(P) \cup traces(Q)$$

$$traces(P \sqcap Q) = traces(P) \cup traces(Q)$$

$$traces(\mu .p.F(p)) = \bigcup_{i=0}^{\infty} traces(F^i(Stop))$$

$$\text{traces}(P \parallel_X Q) = \bigcup \{s \parallel_X t \mid s \in \text{traces}(P) \wedge t \in \text{traces}(Q)\}$$

$$\text{traces}(P \setminus A) = \{s \upharpoonright \Sigma \setminus A \mid s \in \text{traces}(P)\}$$

$$\begin{aligned} \text{traces}(P; Q) &= (\text{traces}(P) \cap \Sigma^*) \\ &\quad \cup \{s \hat{\ } t \mid s \hat{\ } \langle \checkmark \rangle \in \text{traces}(P) \wedge t \in \text{traces}(Q)\} \end{aligned}$$

- Traces refinement

Process  $Q$  **refines** process  $P$  ( $P \sqsubseteq_T Q$ ) if  $traces(Q) \subseteq traces(P)$ .

▷ Mechanical verification with tools like FDR.

▷ Refinement has many properties that can be exploited, for example

$$P \sqsubseteq Q \wedge Q \sqsubseteq R \Leftrightarrow P \sqsubseteq R$$

$$P \sqsubseteq Q \wedge Q \sqsubseteq P \Leftrightarrow P = Q$$

▷ **Compositionality**, that is

$$P \sqsubseteq Q \Rightarrow C[P] \sqsubseteq C[Q].$$

▷ **Stepwise refinement**

$$Spec \sqsubseteq P_1 \sqsubseteq \dots \sqsubseteq P_n \sqsubseteq Impl$$

- The failures model ( $\mathcal{F}$ )

- Definitions

$$\text{refusals}(P) = \{a \mid \nexists s \in \Sigma^*. a \hat{ } s \in \text{traces}(P)\}$$

$\text{failures}(P)$  consists of all pairs of  $(s, X)$  where  $s$  is a trace of  $P$  and  $X$  a set of actions  $P$  can refuse in some **stable state** (unable to perform  $\tau$  or  $\checkmark$ ) after  $s$ , or results from a state after  $s$  which can perform  $\checkmark$  and  $X \subseteq \Sigma$ .

○ Calculating  $failures(P)$  (examples)

$$failures(Stop) = \{(\langle \rangle, X) \mid X \subseteq \Sigma^\vee\}$$

$$failures(Skip) = \{(\langle \rangle, X) \mid X \subseteq \Sigma\} \cup \{(\langle \checkmark \rangle, X) \mid X \subseteq \Sigma^\vee\}$$

$$\begin{aligned} failures(e \rightarrow P) = & \{(\langle \rangle, X) \mid comms(e) \cap X = \{\}\} \\ & \cup \{(\langle a \rangle^s, X) \mid a \in comms(e) \\ & \quad \wedge (s, X) \in failures(subs(a, e, P))\} \end{aligned}$$

$$failures(P \sqcap Q) = failures(P) \cup failures(Q)$$

$$\begin{aligned} failures(P \sqcup Q) = & \{(\langle \rangle, X) \mid (\langle \rangle, X) \in failures(P) \cap failures(Q)\} \\ & \cup \{(s, X) \mid (s, X) \in failures(P) \cup failures(Q) \wedge s \neq \langle \rangle\} \\ & \cup \{(\langle \rangle, X) \mid X \subseteq \Sigma \wedge \langle \checkmark \rangle \in traces(P) \cup traces(Q)\} \end{aligned}$$

$$\begin{aligned}
failures(P \parallel_X Q) = & \\
& \{(u, Y \cup Z) \mid Y \setminus (X \cup \checkmark) = Z \setminus (X \cup \checkmark) \\
& \wedge \exists s, t. (s, Y) \in failures(P) \\
& \wedge (t, Z) \in failures(Q) \wedge u \in s \parallel_X t\}
\end{aligned}$$

$$failures(P \setminus X) = \{(s \setminus X, Y) \mid (s, Y \cup X) \in failures(P)\}$$

$$\begin{aligned}
failures(P; Q) = & \\
& \{(s, X) \mid (s, X \cup \{\checkmark\}) \in failures(P)\} \\
& \cup \{(s \hat{\ } t, X) \mid s \hat{\ } \langle \checkmark \rangle \in traces(P) \wedge (t, X) \in failures(Q)\}
\end{aligned}$$

- The failures/divergences model ( $\mathcal{FD}$ )

- Definition

**failures/divergences model:**  $(failures_{\perp}(P), divergences(P))$

$divergences(P)$  is a set of traces  $s$  of  $P$  on which  $P$  can diverge, in the sense that an infinite unbroken sequence of  $\tau$  actions can occur after some  $s' \leq s$ .

$$failures_{\perp}(P) = failures(P) \cup \{(s, X) \mid s \in divergences(P)\}$$

$$traces_{\perp}(P) = traces(P) \cup \{s \mid s \in divergences(P)\}$$

- Calculating  $\text{divergences}(P)$  (examples)

$$\text{divergences}(\text{Stop}) = \{\}$$

$$\text{divergences}(\text{Skip}) = \{\}$$

$$\begin{aligned} \text{divergences}(e \rightarrow P) = \{ \langle a \rangle^s \mid & a \in \text{comms}(e) \\ & \wedge s \in \text{divergences}(\text{subs}(a, e, P)) \} \end{aligned}$$

$$\text{divergences}(P \square Q) = \text{divergences}(P) \cup \text{divergences}(Q)$$

$$\text{divergences}(P \sqcap Q) = \text{divergences}(P) \cup \text{divergences}(Q)$$

$$\begin{aligned}
 \text{divergences}(P \parallel_X Q) &= \{u \hat{v} \mid \\
 &\exists s \in \text{traces}_\perp(P), t \in \text{traces}_\perp(Q). u \in (s \parallel_X t) \cap \Sigma^* \\
 &\wedge (s \in \text{divergences}(P) \vee t \in \text{divergences}(Q))\}
 \end{aligned}$$

$$\text{divergences}(P \setminus X) = \{(s \setminus X) \hat{t} \mid s \in \text{divergences}(P)\}$$

$$\begin{aligned}
 \text{divergences}(P; Q) &= \text{divergences}(P) \cup \\
 &\{s \hat{t} \mid s \hat{\langle \checkmark \rangle} \in \text{traces}_\perp(P) \wedge t \in \text{divergences}(Q)\}
 \end{aligned}$$

# Process relations

- Equivalences

$$P =_{\mathcal{T}} Q \quad \text{traces}(Q) = \text{traces}(P)$$

$$P =_{\mathcal{F}} Q \quad \text{failures}(Q) = \text{failures}(P)$$

$$P =_{\mathcal{FD}} Q \quad \text{failures}_{\perp}(Q) = \text{failures}_{\perp}(P) \text{ and} \\ \text{divergences}(Q) = \text{divergences}(P)$$

- Refinements

$$P \sqsubseteq_{\mathcal{T}} Q \quad \text{traces}(Q) \subseteq \text{traces}(P)$$

$$P \sqsubseteq_{\mathcal{F}} Q \quad \text{failures}(Q) \subseteq \text{failures}(P)$$

$$P \sqsubseteq_{\mathcal{FD}} Q \quad \text{failures}_{\perp}(Q) \subseteq \text{failures}_{\perp}(P) \text{ and} \\ \text{divergences}(Q) \subseteq \text{divergences}(P)$$

---

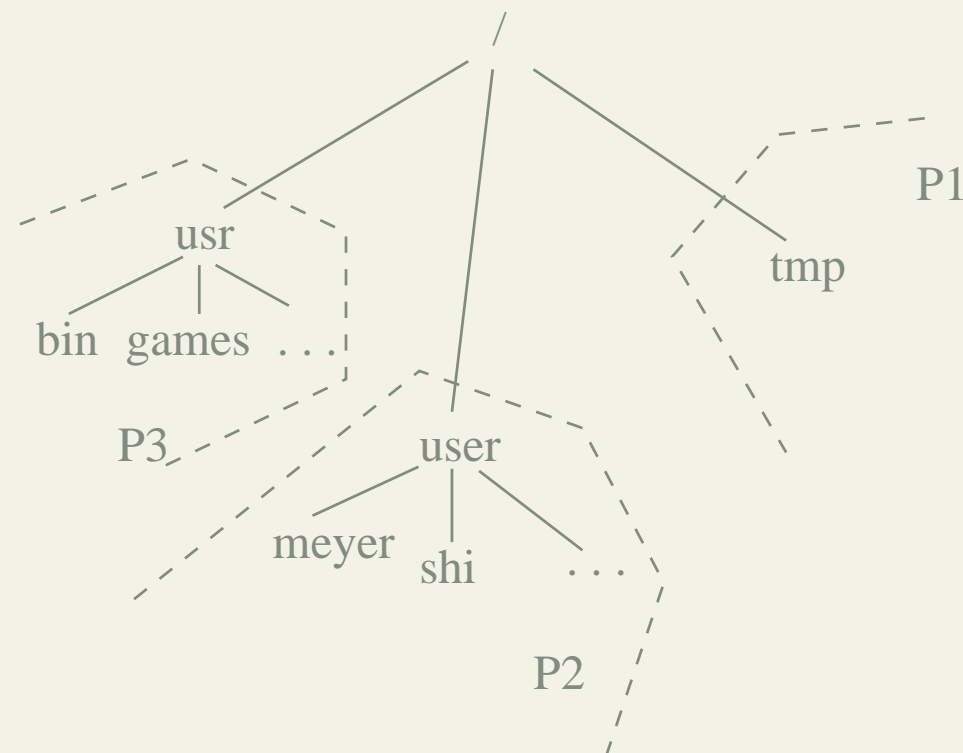
# A Case Study

# NFS – Network File System

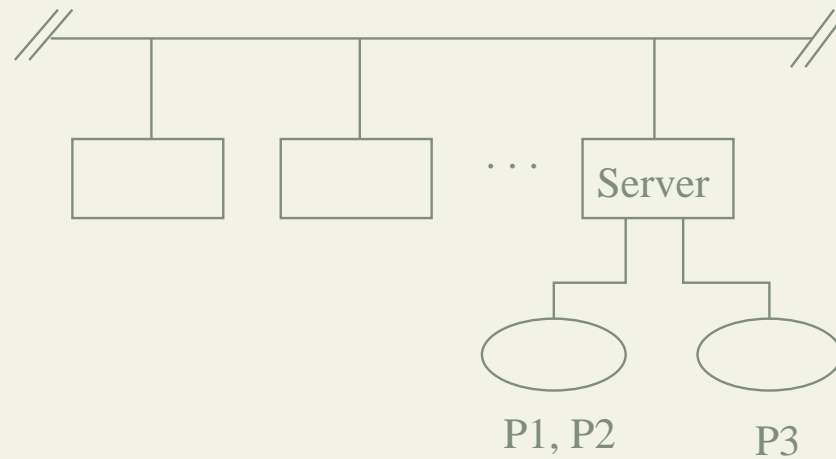
- Konzepte

- Verteilte Dateisysteme

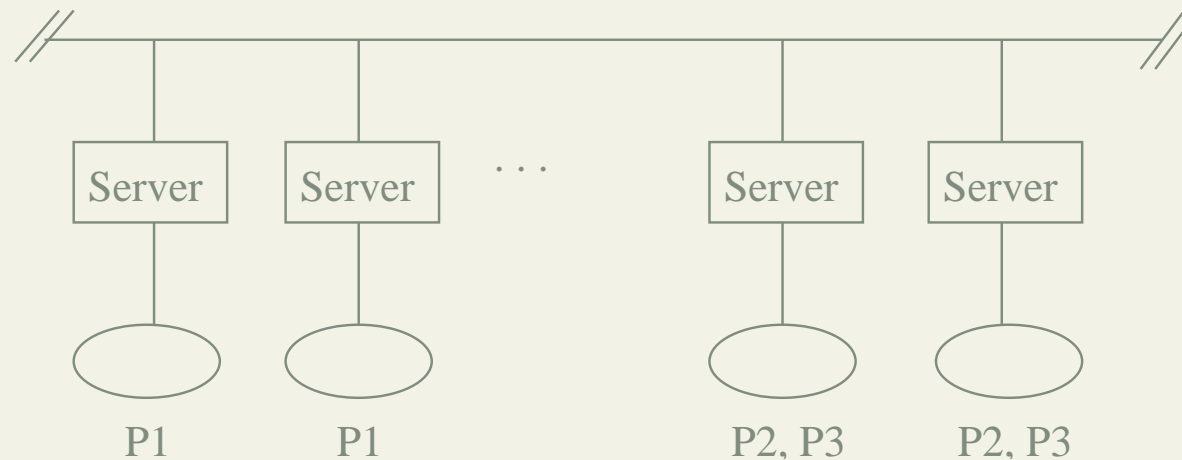
- ▷ Partitionen eines hierarchischen Dateisystems



▷ Striktes Client-Server-Modell



▷ Verteilte File-Server-Modell



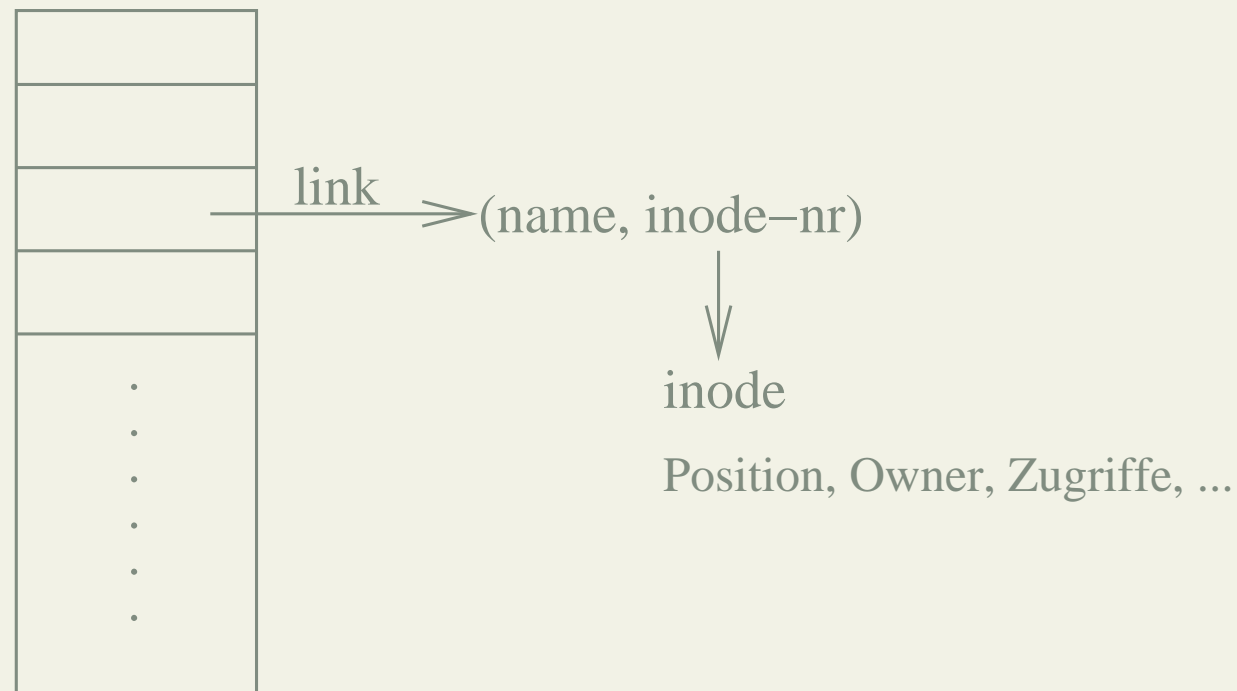
- Entwurfsziele
  - ▷ Transparenz
  - ▷ Performance
  - ▷ Fehlertoleranz
- Zustandslose vs. zustandsbehaftete Server
  - ▷ Ein **zustandsbehafteter Server** verwaltet Informationen über seine Clients, so daß für die Dauer der Dienstleistung eine logische Verbindung zwischen Client und Server besteht. (gute Performance, großer Aufwand)
  - ▷ Ein **zustandsloser Server** speichert keine Daten über seine Clients. Bei jedem Zugriffswunsch muß der Client sämtliche Parameter übersenden. (gringer Aufwand, schlechte Performance)

- Naming

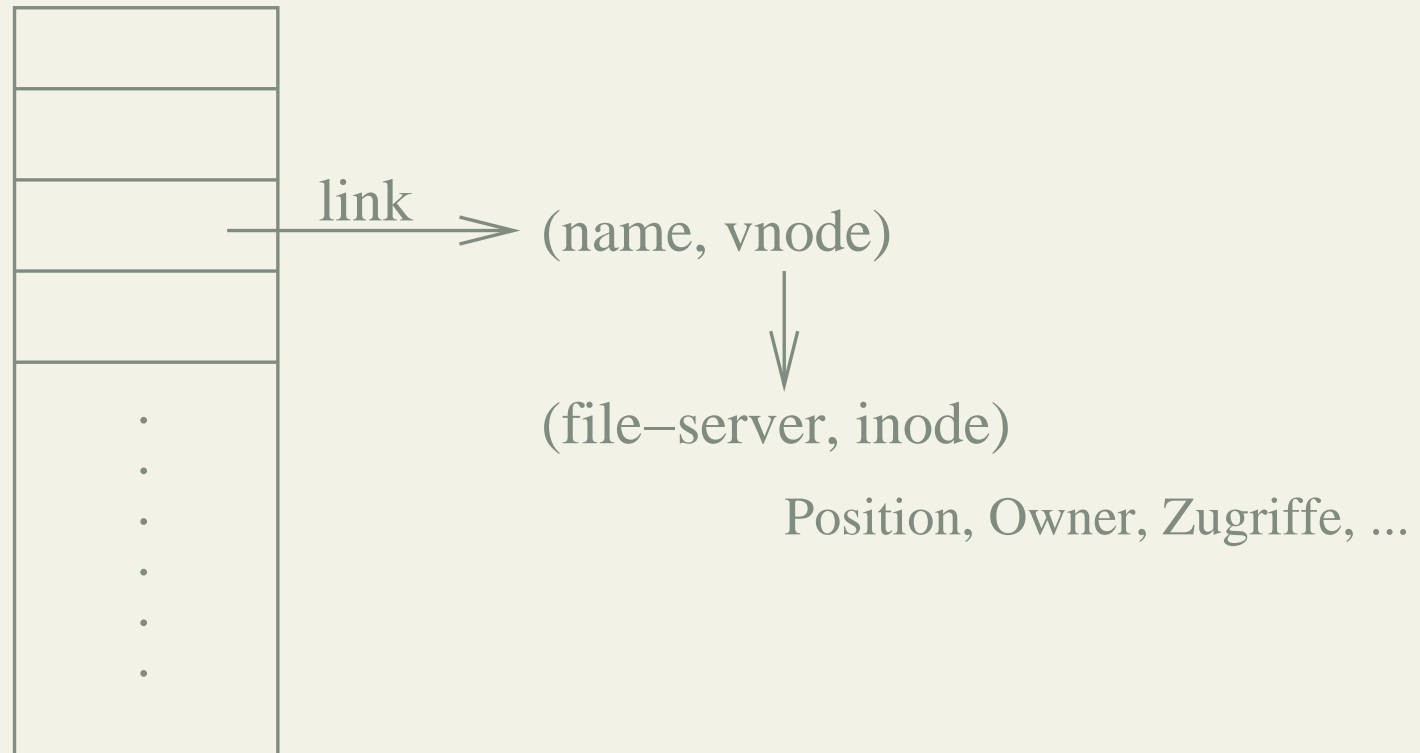
- ▷ Mit **Naming** bezeichnet man das Abbilden der logischen Datenobjekte (Dateinamen) auf physikalische Objekte (Adresse auf dem Speichermedium).

- ▷ Klassische UNIX-Dateisysteme

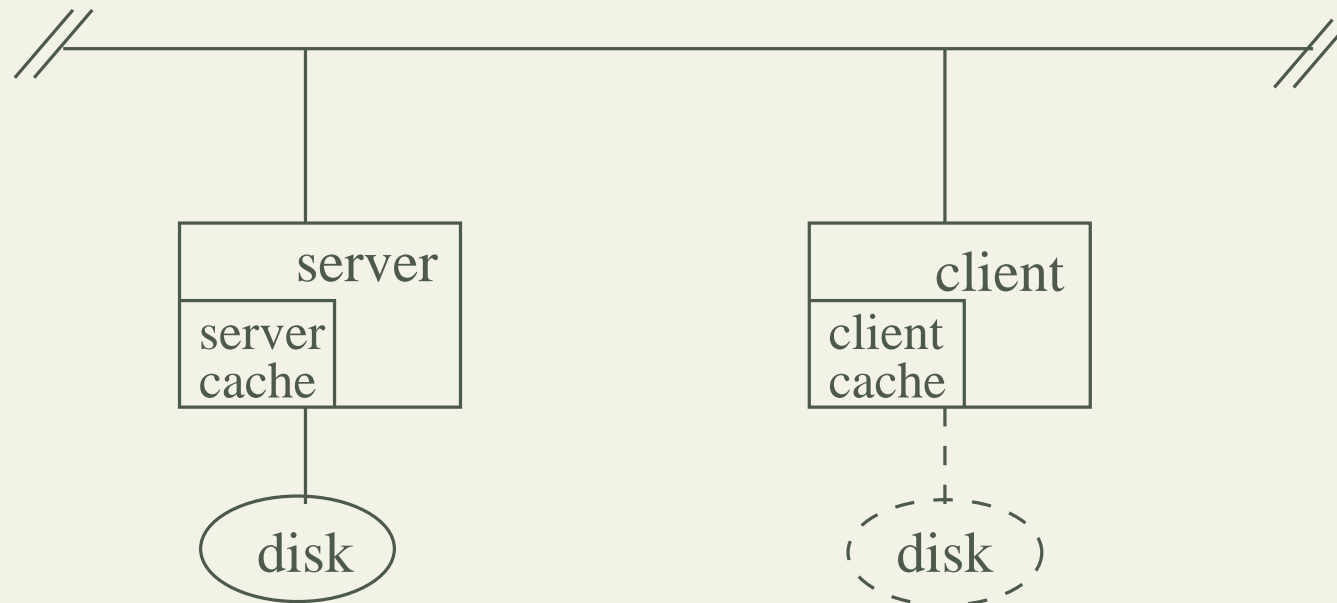
Dateiverzeichnis



▷ Verteilte Dateisysteme  
globales  
Dateisystem



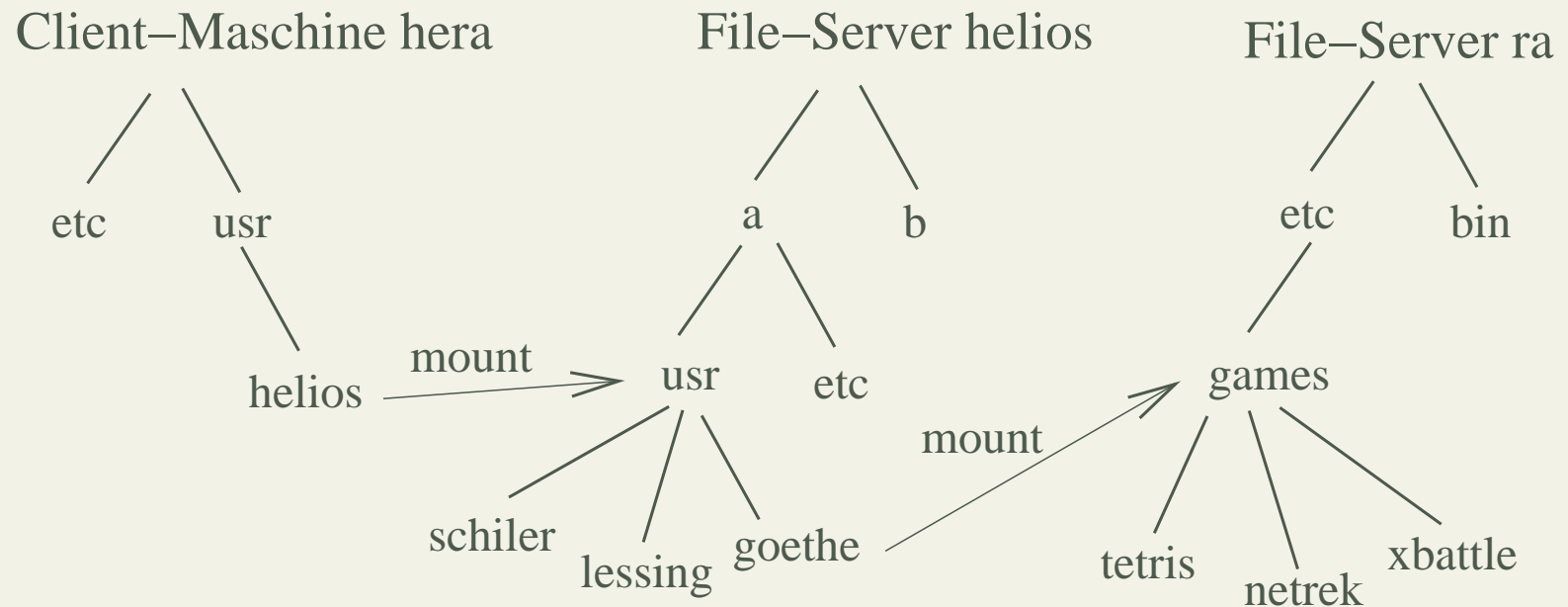
- Caching



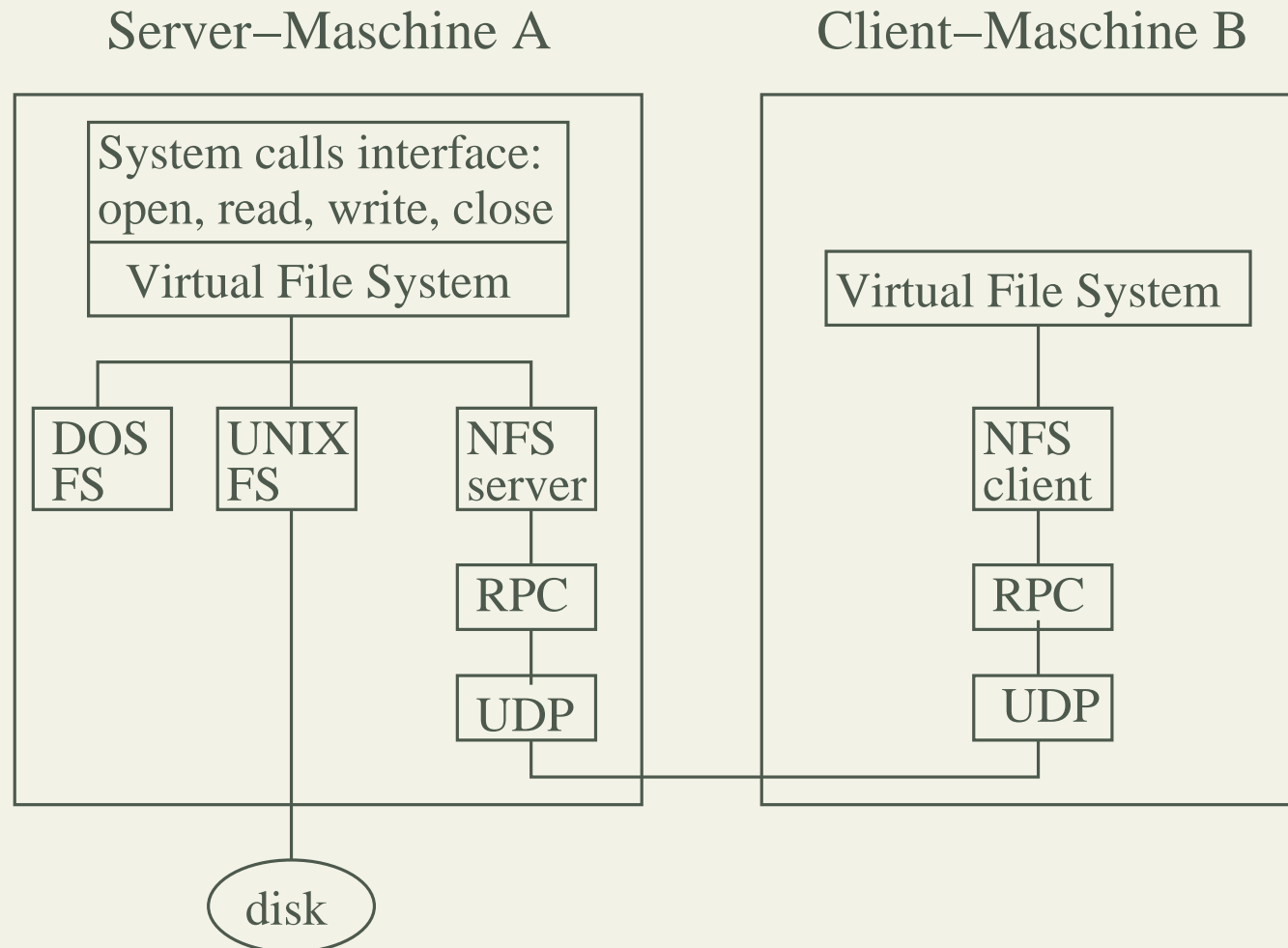
- ▷ **Write-Through**: es werden nur Daten im Cache gespeichert, auf die lesend zugegriffen wird.
- ▷ **Delayed-Write**: veränderte Blöcke werden zunächst in den Cache geschrieben und bleiben dort eine gewisse Zeit, bevor sie zum Server zurückgeschrieben werden.

# • NFS-Protokoll

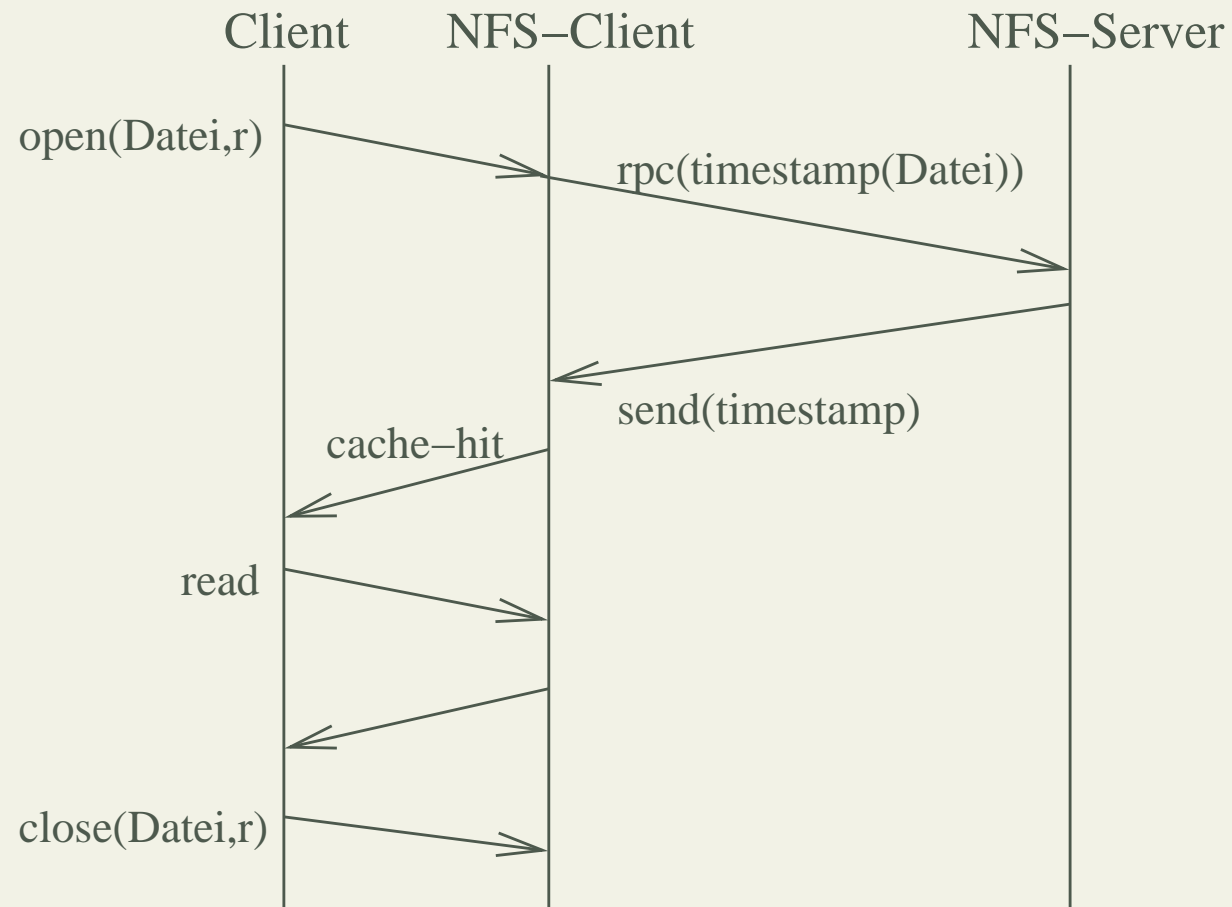
## ○ Mount-Mechanismus



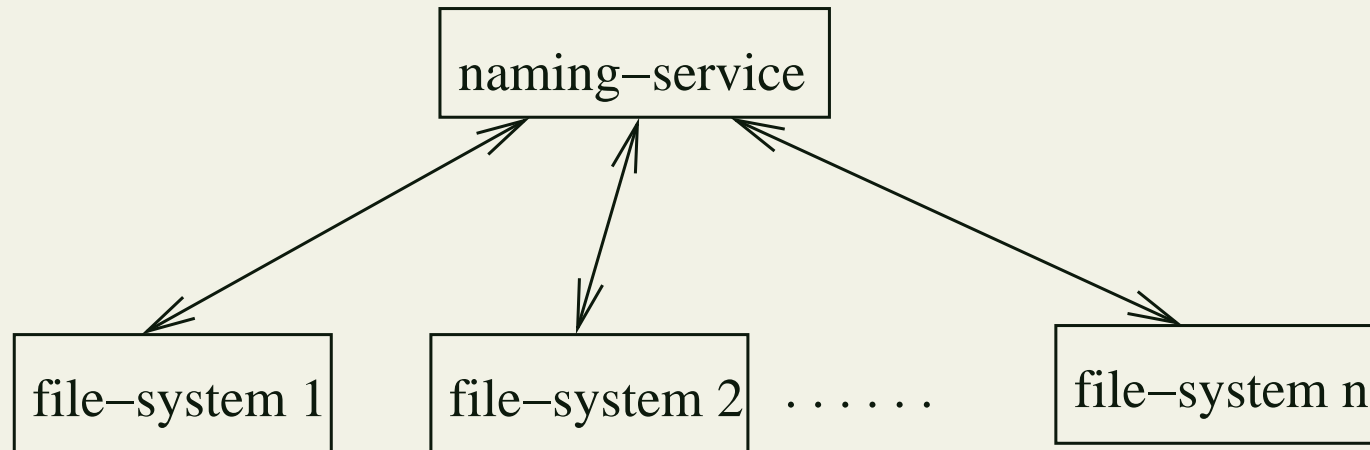
- NFS-Architektur



- Caching



- Ein vereinfachtes NFS-Protokoll

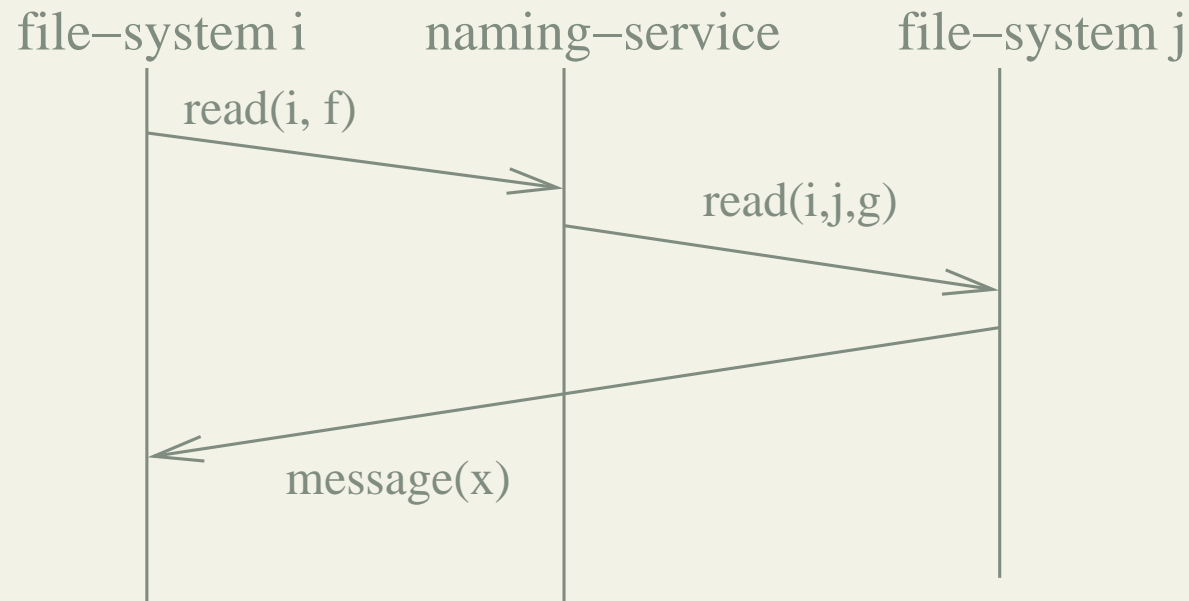


- Vereinfachte Dateisysteme

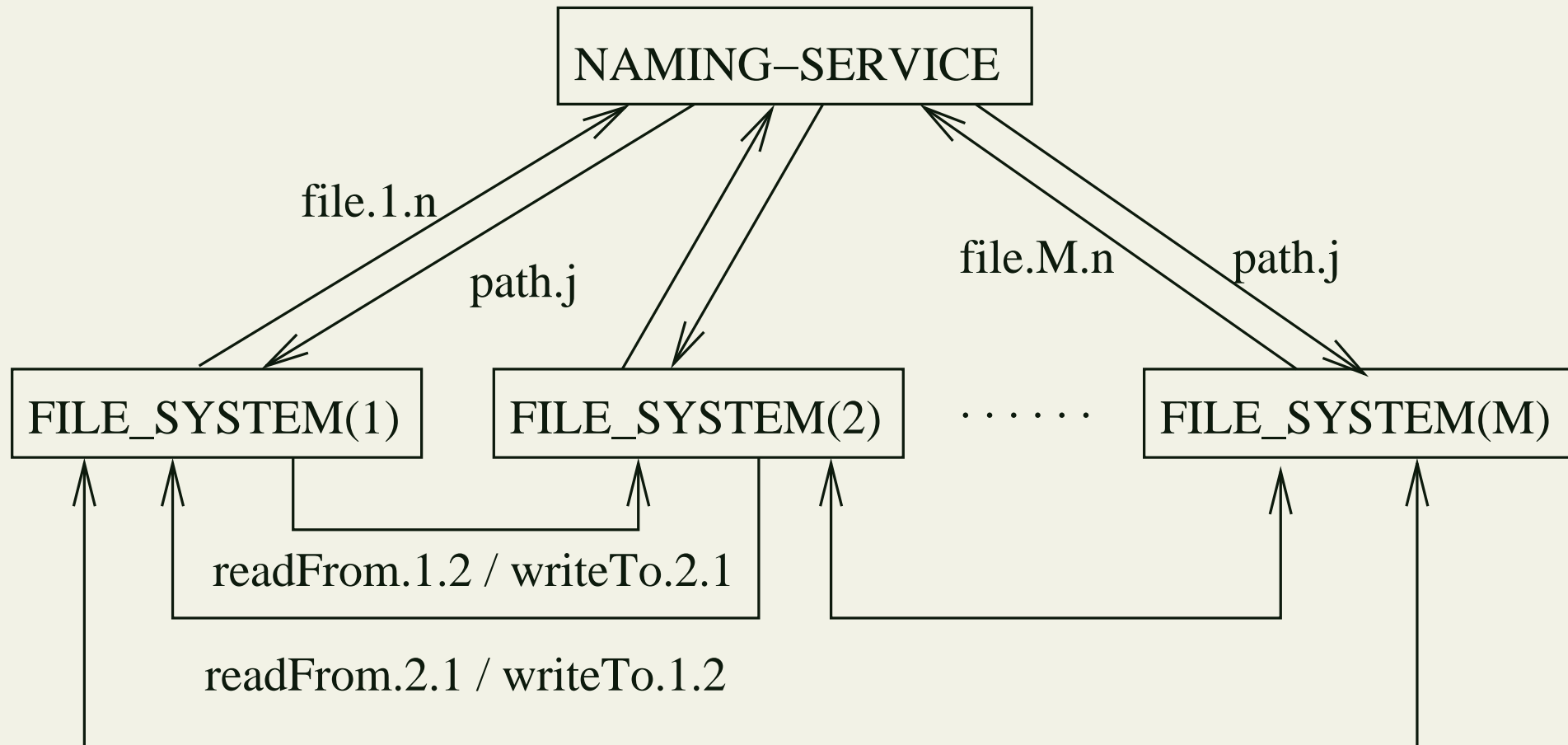
- ▷ Ein Dateisystem besteht aus einer Menge von Dateien
- ▷ Jede Datei eines Dateisystems kann von allen Benutzern gelesen und verändert werden.
- ▷ Sei  $DS_1$  und  $DS_2$  zwei Dateisysteme, dann haben sie keine gemeinsame Datei.

- Naming-Service

- ▷ Der Naming-Service leitet Anfragen eines Dateisystems an das entsprechenden Dateisystem weiter.
- ▷ Das angeforderte Dateisystem bearbeitet die Anfragen und schickt ggf. eine Antwort an das erste Dateisystem zurück.



# Specification with CSP



# Verification with FDR (Deadlock- und Livelock-Freiheit)

- Ein FDR-Spezifikation
  - Festlegen der Anzahl von Datei-Systeme
  - Festlegen der Anzahl von Dateien innerhalb eines Datei-Systems
  - Festlegen der Zustände einer Datei
- Eine abstrakte FDR-Spezifikation
  - Abstraktion der Zustände einer Datei
  - Abstraktion der Dateien innerhalb eines Datei-Systems
  - Abstraktion der Datei-systeme