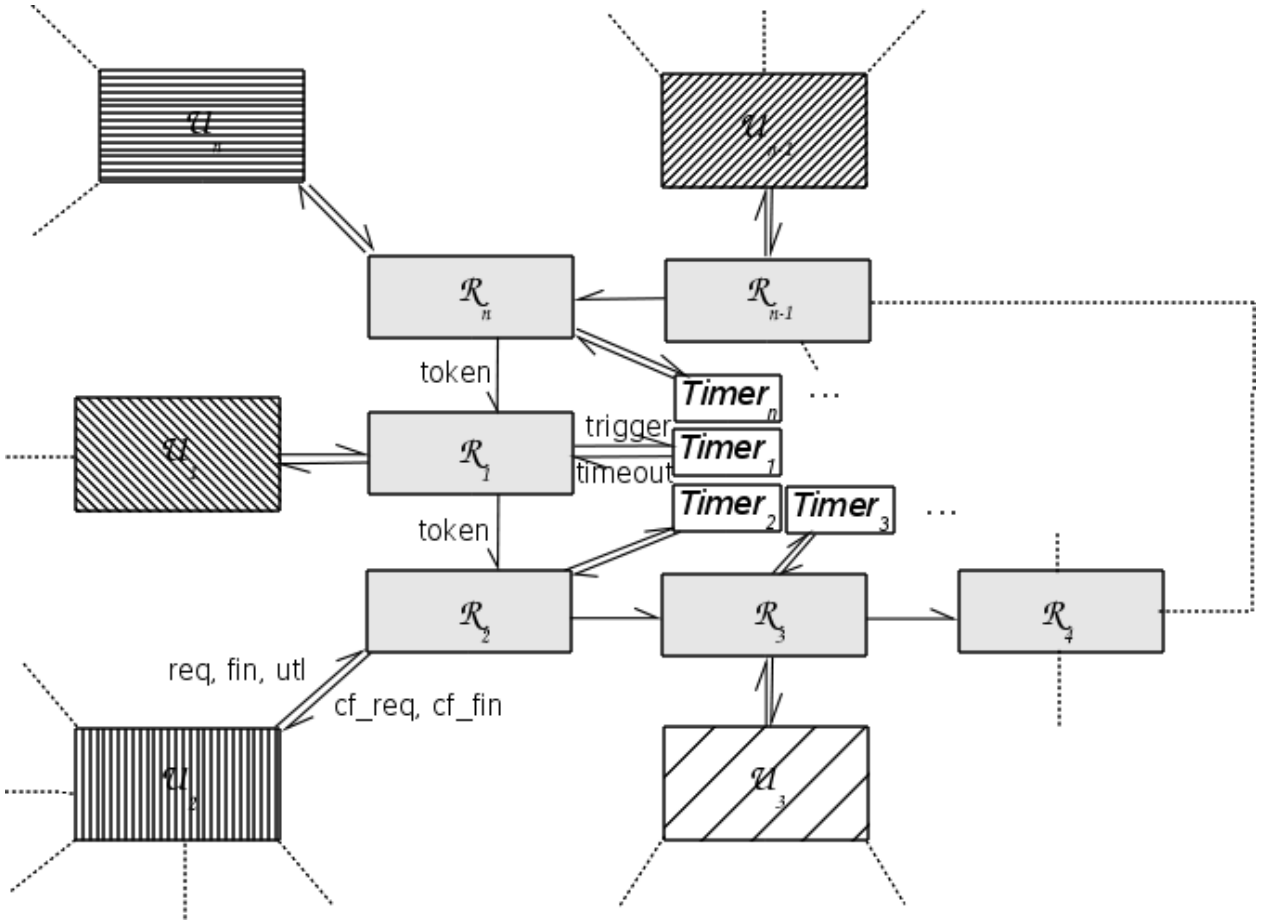
Process synthesis from multiple interaction specifications

Johannes Reich

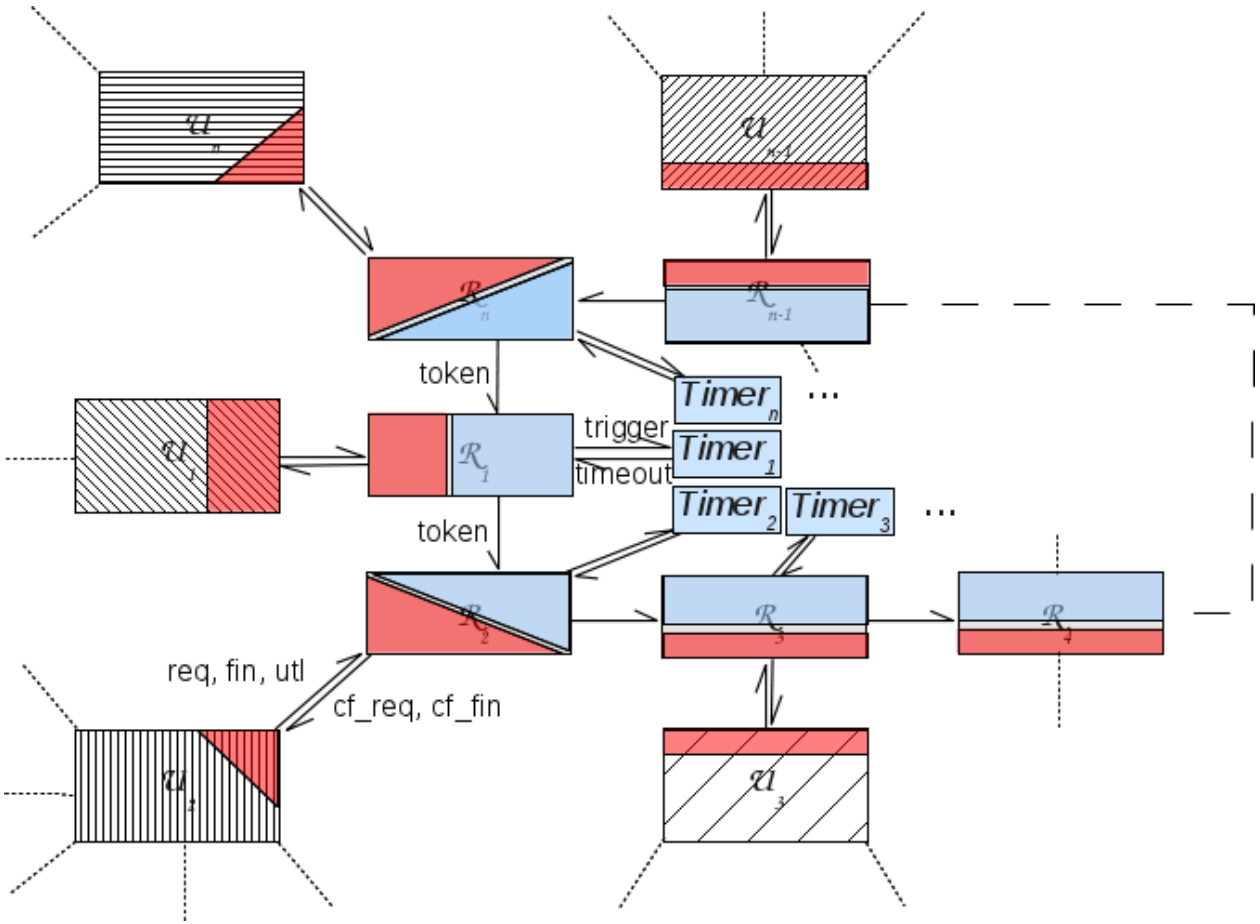
2011-10-04

Table of Contents

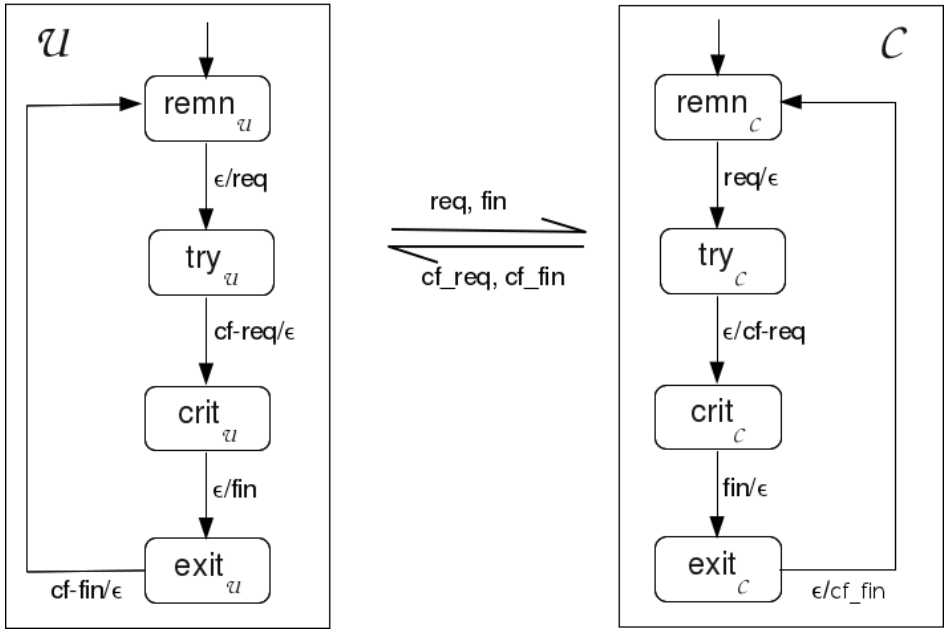
Example of Complex Interacting Systems: A Ring of Resource Manager Processes $\mathcal{R}_i$



Example of Complex Interacting Systems: A Ring of Resource Manager Processes $\mathcal{R}_i$

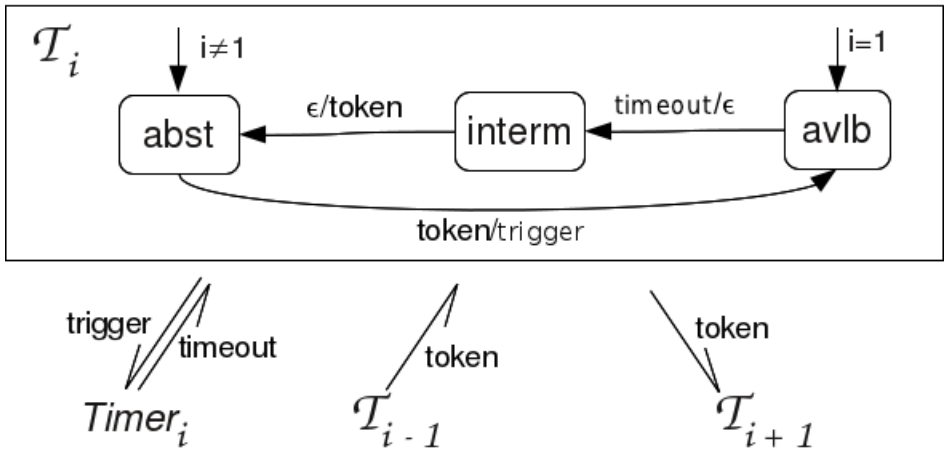


Interaction 1: The Protocol of Mutual Exclusion



A system $\mathcal{U}$ is interacting with a resource manager system $\mathcal{R}$ to get exclusive access to a resource. $\mathcal{U}$ can access the ressource only while it is in its Crit(-ical) state.

Interaction 2: Token Ring Protocol



A ring of resource managers $\mathcal{R}_i$, $i = 1 \ldots n$ coordinates the exclusive access to a resource $\mathcal{U}$ by exchange of a token.

Nondeterministic Finite I/O Automata

Def.: A **nondeterministic finite I/O automaton (NFIOA)** is defined by a tuple $\mathcal{A} = (Q, I, O, q_0, Acc, \Delta)$.

- Q is the non-empty finite set of state values,
- I and O are the finite input and output alphabets where at least one of both is non empty,
- q_0 is the initial state value,
- Acc is the acceptance component and
- $\Delta \subseteq Q \times Q \times I^\epsilon \times O^\epsilon$ is the transition relation.

For finite input sequences:$Acc \subseteq Q$.
For infinite input sequences: the Acc depends on the computational model (Büchi, Rabin, Streett, Muller, etc.). The Muller acceptance condition is to accept all runs in which the set of infinitely often occuring state values is an element of Acc, that is $Acc \subseteq 2^Q$.

Interactions: Synchronizing NFIOAs

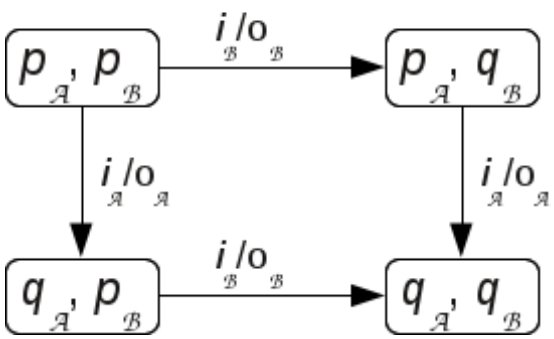
Def.: The **unsynchronized product** of a set of n NFIOAs $\mathcal{A}_k$ is defined by NFIOA $\mathcal{B} = (Q, I, O, \vec{q}_0, Acc, \Delta)$, with $Q_\mathcal{B} = \times Q_k$, $I_\mathcal{B} = \times I_k$, $O_\mathcal{B} = \times O_k$, $\vec{q}_{0\mathcal{B}} = (q_{01}, \ldots, q_{0n})$, $Acc_\mathcal{B} = \bigwedge Acc_k$ is the common acceptance component, $\Delta_\mathcal{B} := \{(\vec{p}, \vec{q}, \vec{i}, \vec{o}) |$ $\vec{p}$ is a reachable state and $\mathcal{A}_k$ provides a transition (p_k, q_k, i_k, o_k) with $\vec{q} = \vec{p}\left[\frac{q_k}{p_k}\right]$ and $\vec{i} = \epsilon[i_k]$ and $\vec{o} = \epsilon[o_k]$ $\}$. I also write $\mathcal{B} = \bigotimes_{i=1}^{n} \mathcal{A}_i$.

"Synchronization" then means to eliminate some transitions from the transition relation $\Delta_\mathcal{P}$ and thereby to eliminate the symmetry of the unsynchronized product

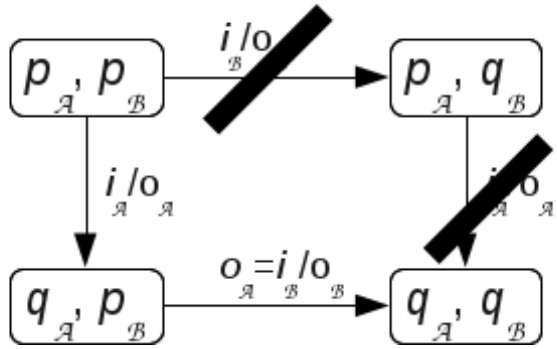
Def.: Be $\mathcal{B} = \bigotimes_{k=1}^{n} \mathcal{A}_k$ an unsynchronized product automaton. An automaton $\mathcal{B}^*$ with all components from $\mathcal{B}$ except a reduced transition set $\Delta_{\mathcal{B}^*} = \Delta_\mathcal{B} \setminus \Delta^e$ with nonempty $\Delta^e \subset \Delta_\mathcal{B}$ is a **synchronized** version of $\mathcal{B}$. I also write $\mathcal{B}^* = \mathcal{B} \setminus \Delta^e$

The Outer Synchronized Product: Channel Mediated System Interaction

Without channel



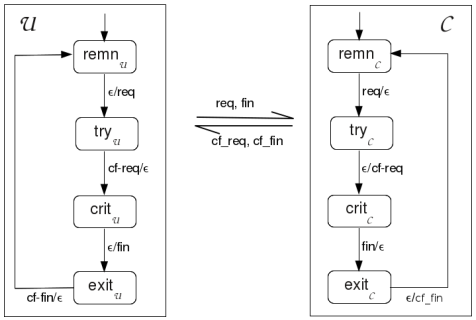
With channel between $\mathcal{A}$ and $\mathcal{B}$



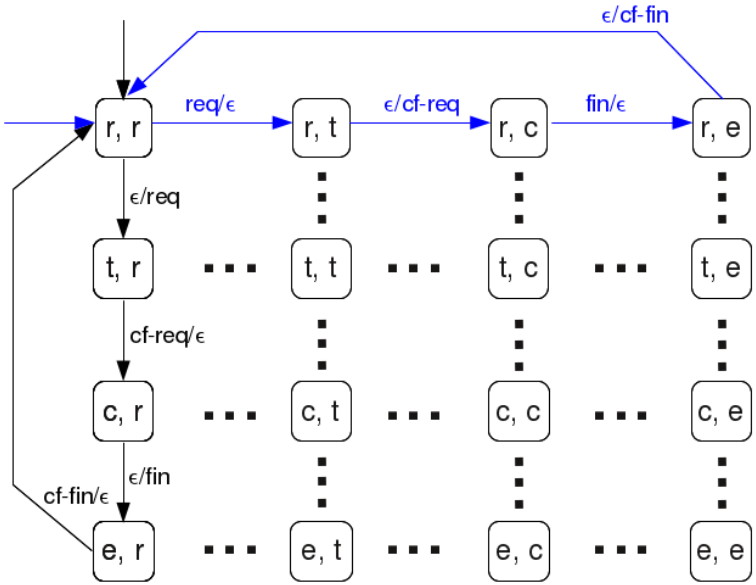
Def.: An outer product of k NFIOAs $\mathcal{P} = \bigotimes_{k=1}^{n} \mathcal{A}_k \setminus \Delta^C$ is called **well formed** if for every channel mediated transition attributable to a sender which sends a character different from ϵ there exists an induced transition of the receiver.

Def.: A well formed outer product is called **consistent** if for each reachable product state value the product acceptance conditions still holds.

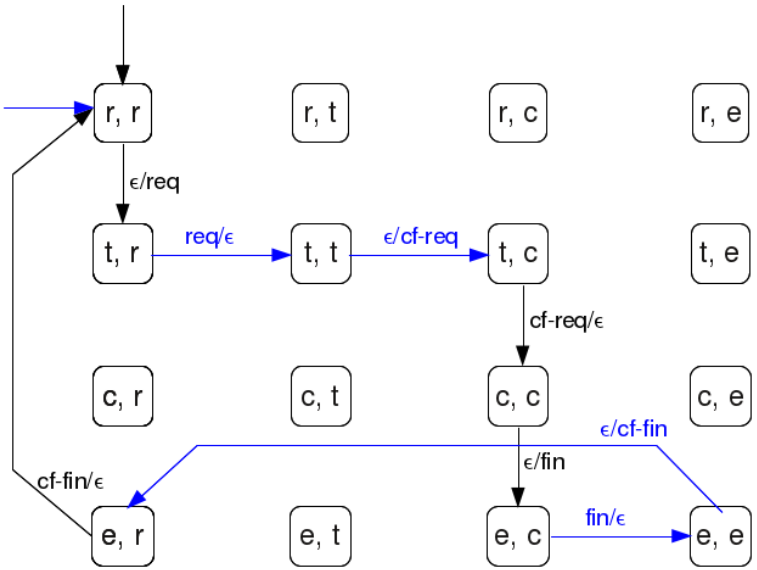
The Outer Synchronized Product: The Protocol of Mutual Exclusion



Unsynchronized product



Channel mediated synchronized product

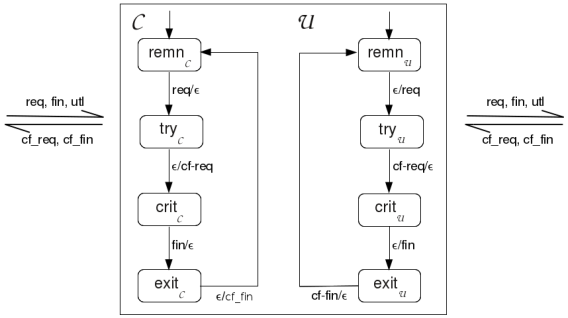


Protocols as a Special Case of the Outer Product

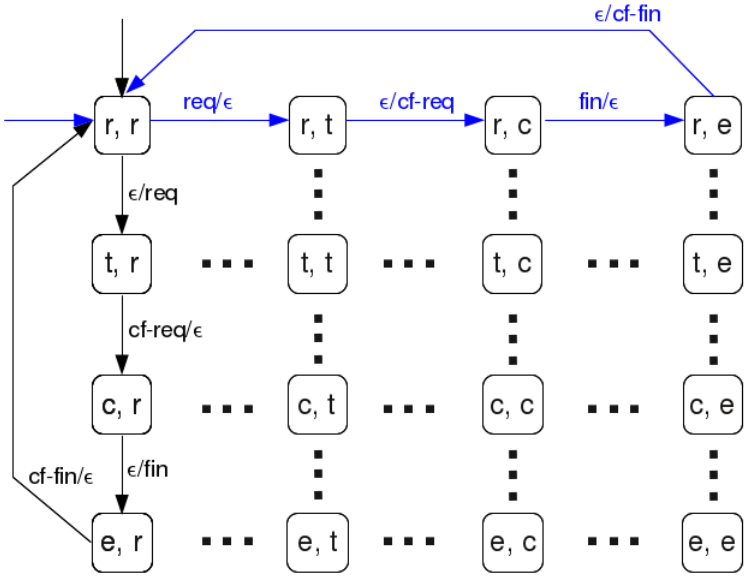
Def.: A **protocol** is the outer product of pairwise different NFIOAs, representing system projections, whose interactions happen completely through a set of channels.

The transition relation of the protocol has to be determined inductively.

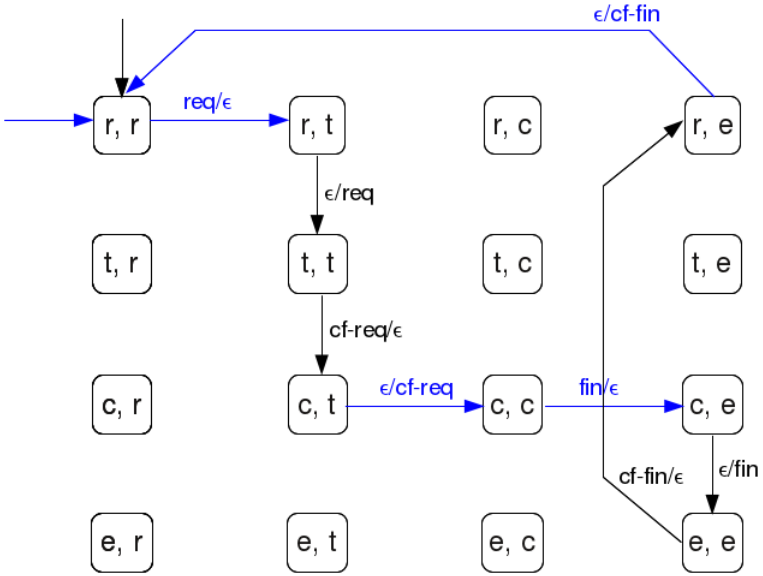
The Inner Synchronized Product: Example "man in the middle"



Unsynchronized product



Inner synchronized product

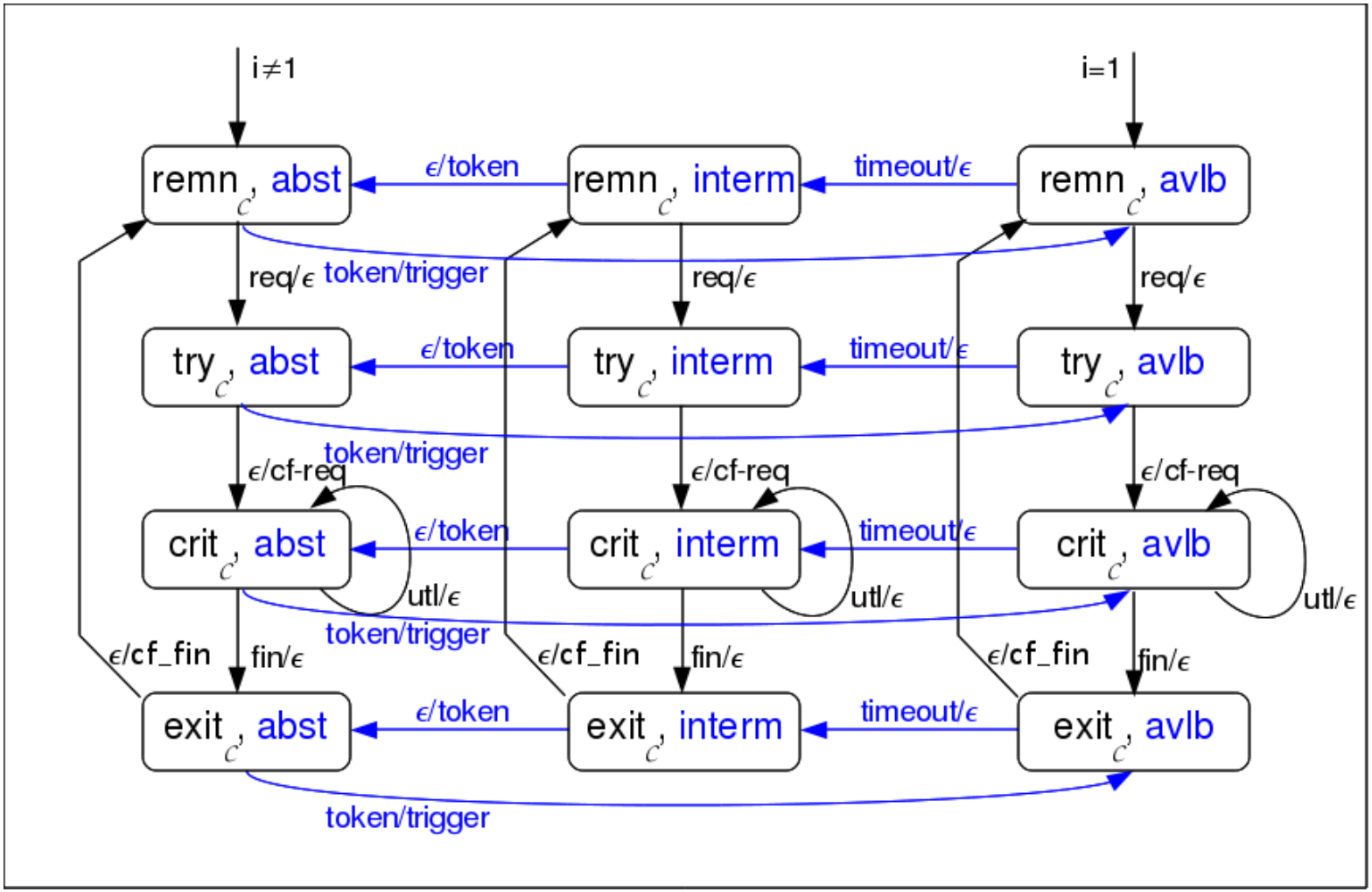


Inner Synchronization by Transition Elimination

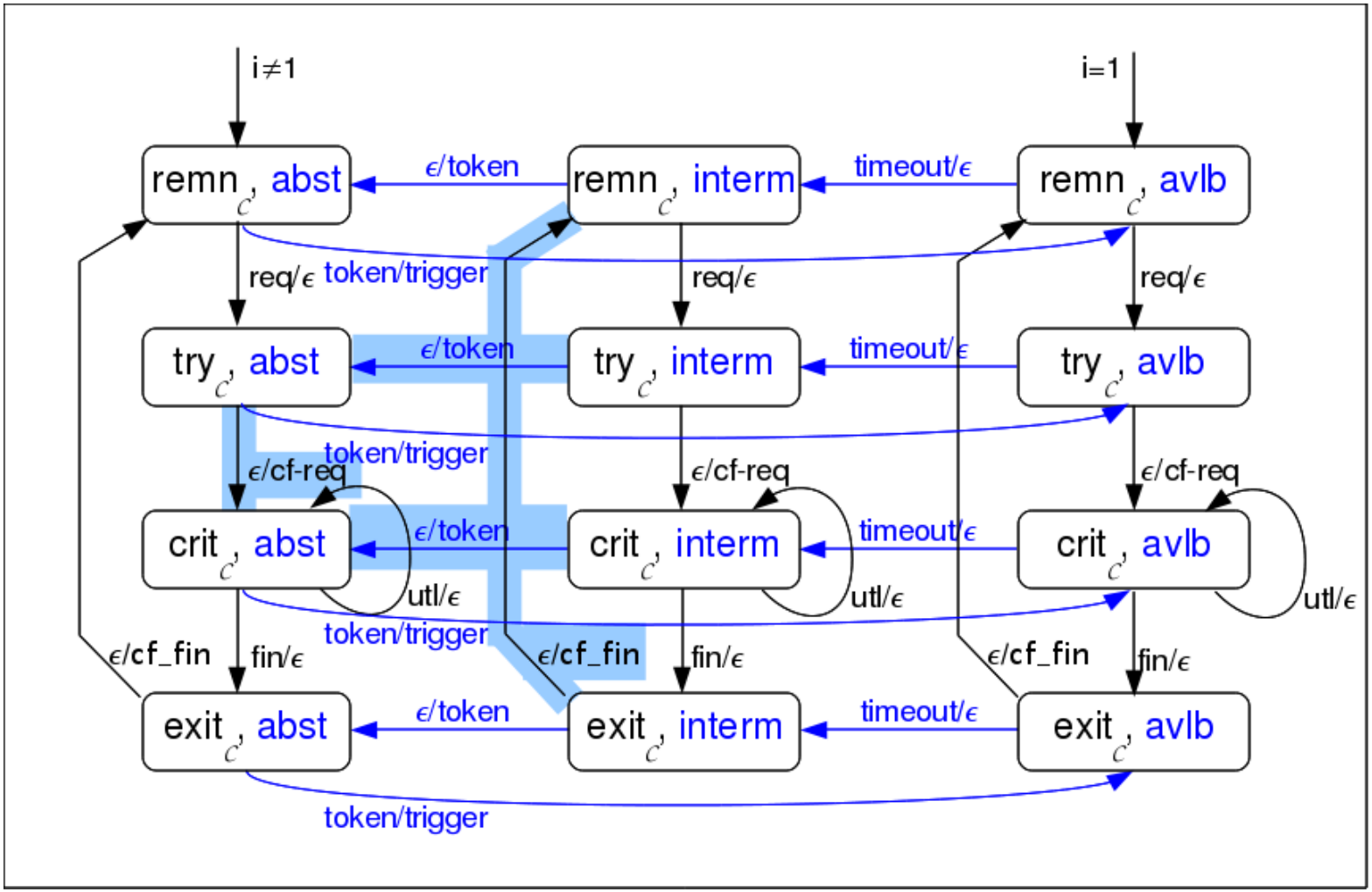
Transitions can be eliminated, if

1. The projection of the synchronized product automaton must result in the original NFIOAs.
2. The common acceptance condition must still be fulfilled,
3. The elimination must provide the coordination semantics of all related interactions.

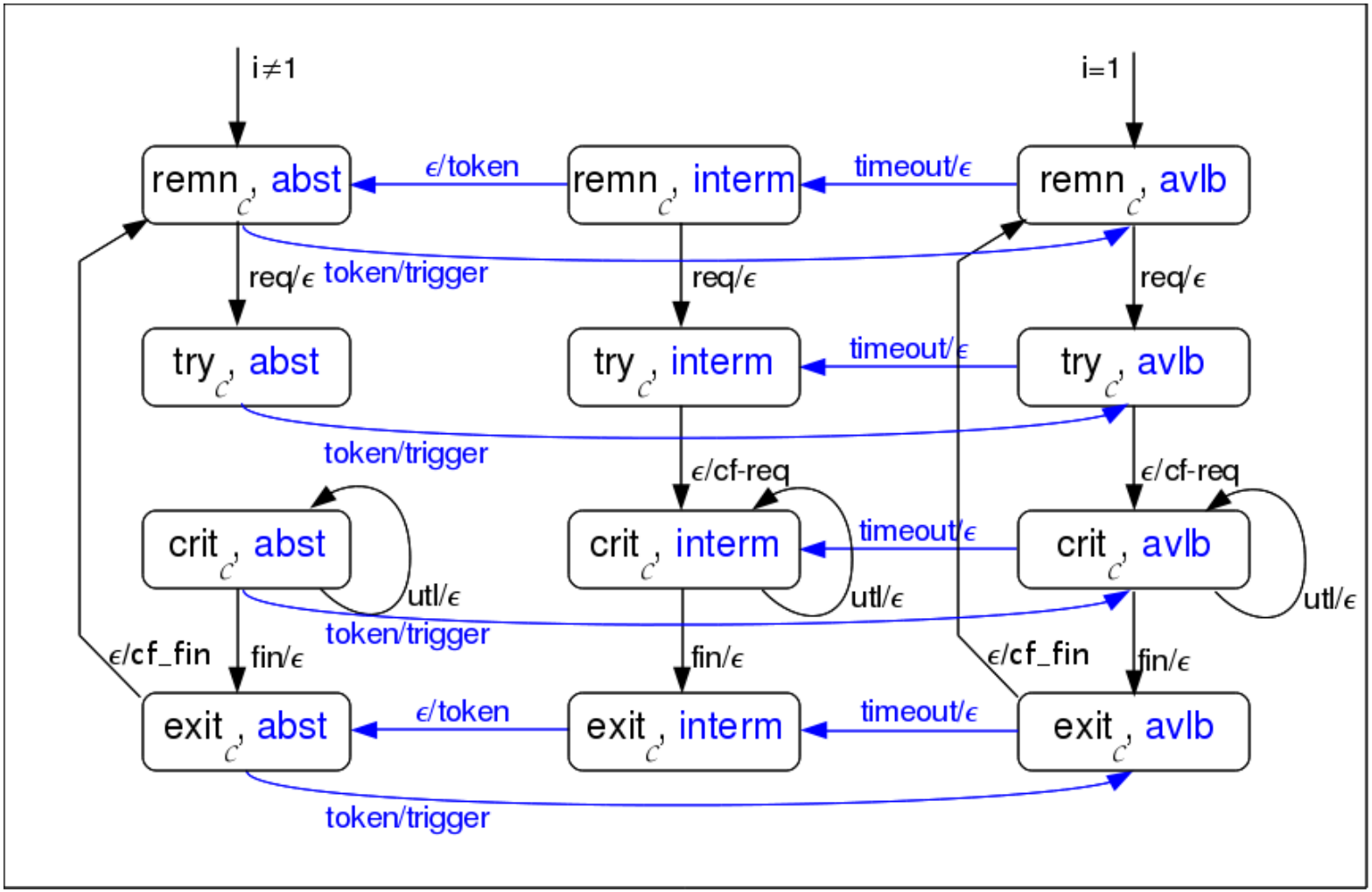
Unsynchronized Product Automaton



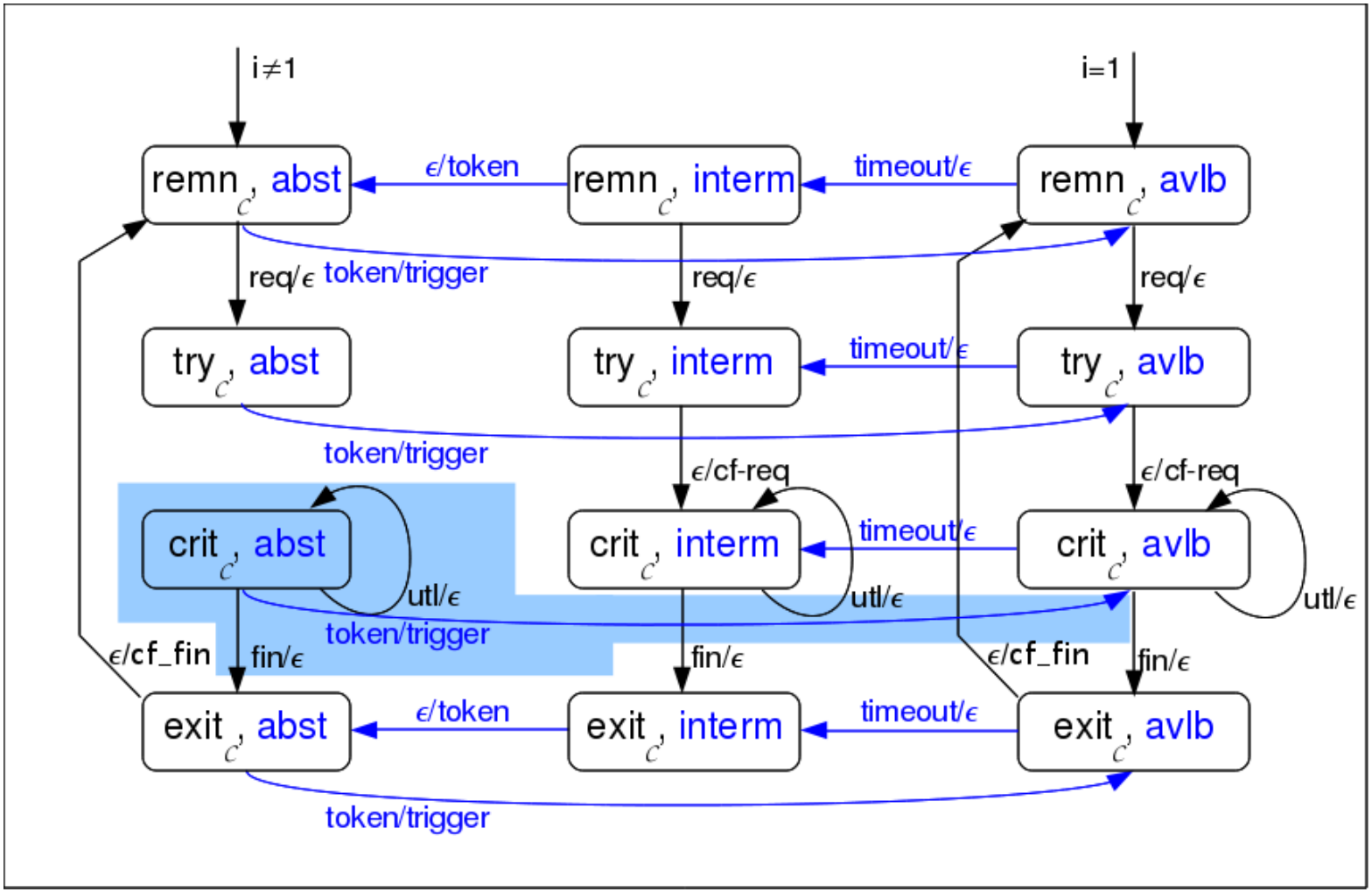
Synchronization, Step 1



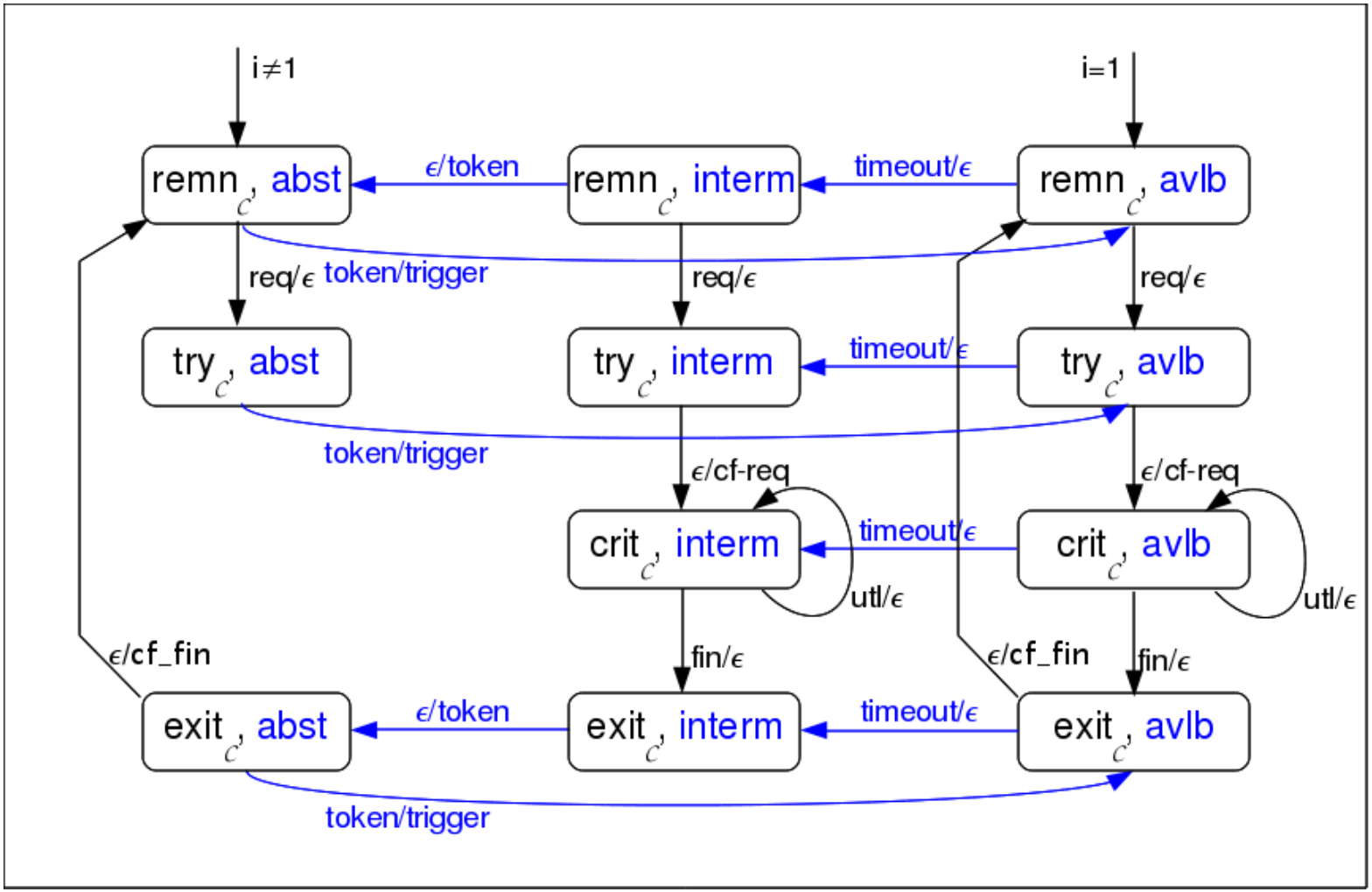
Synchronization, Step 1



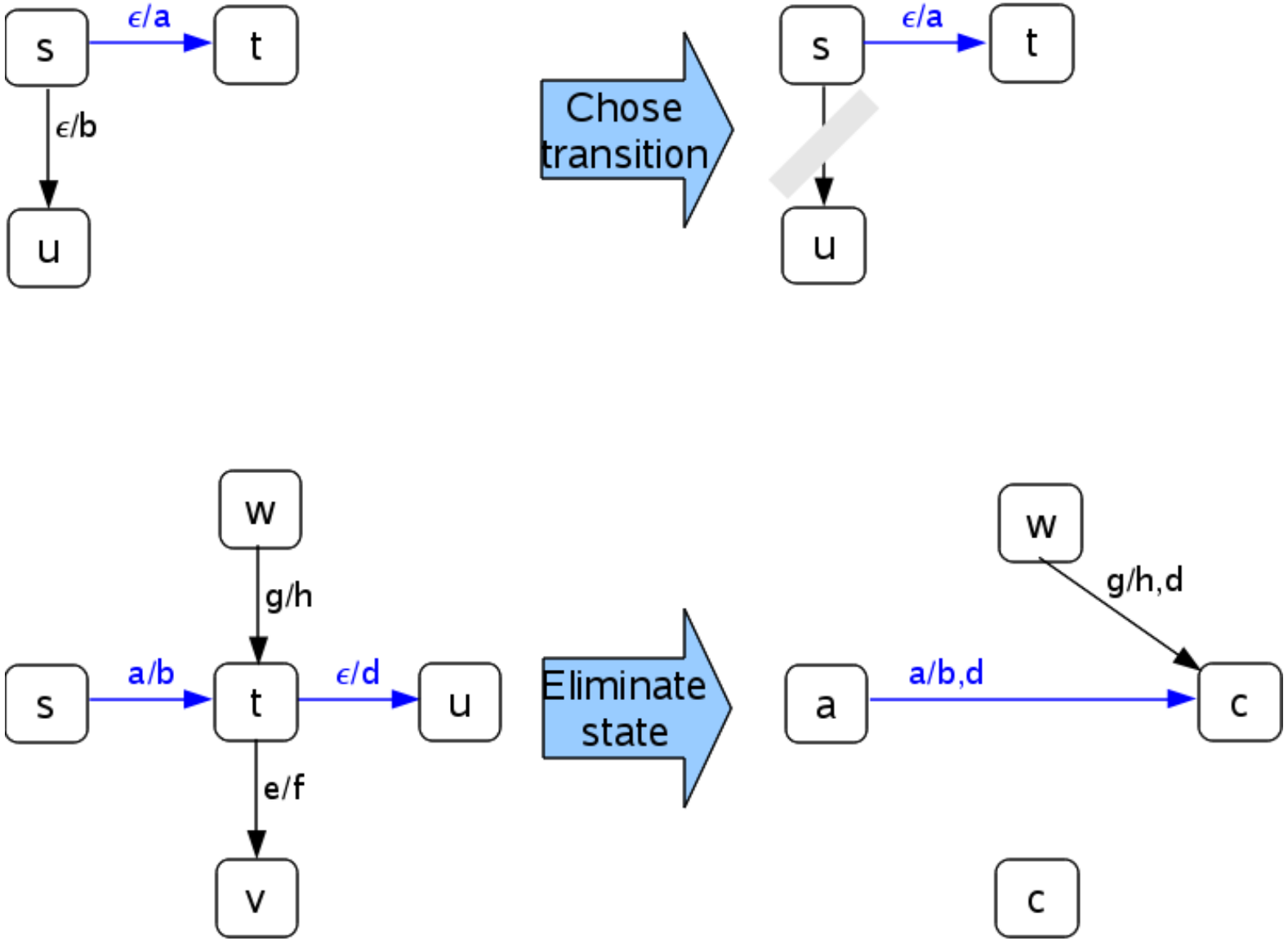
Synchronization, Step 2



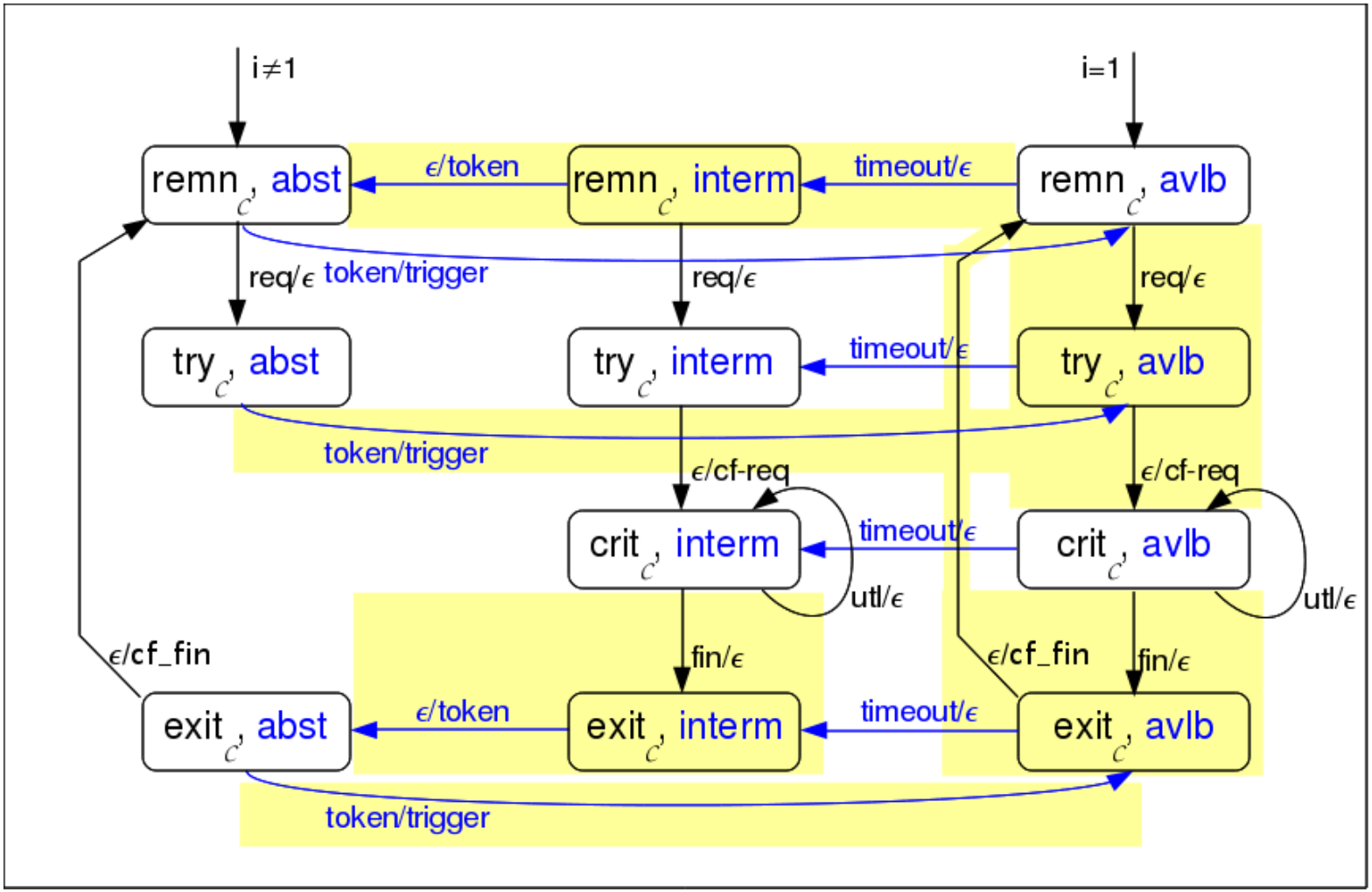
Synchronization, Step 2



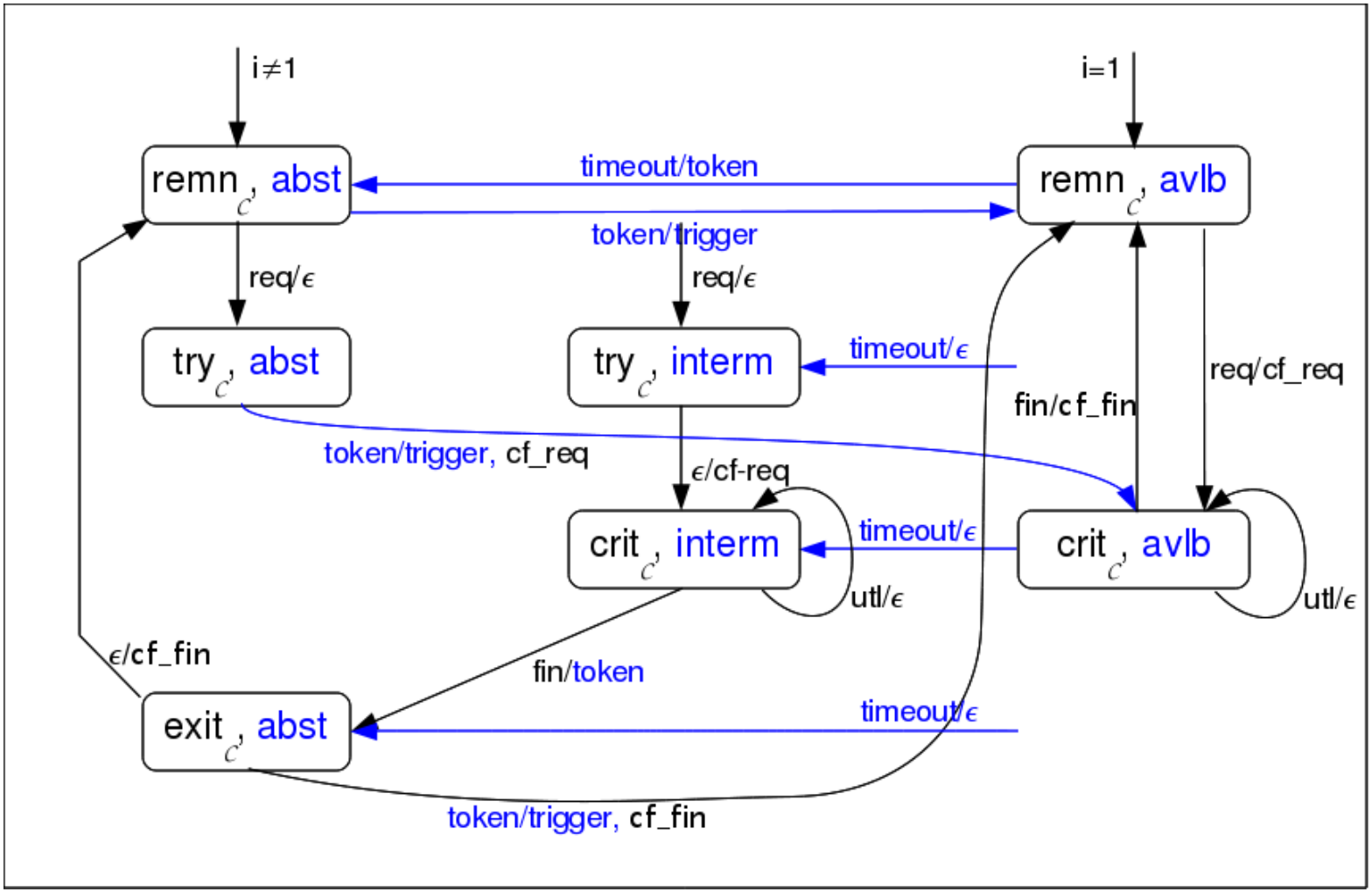
Synchronization and Determination



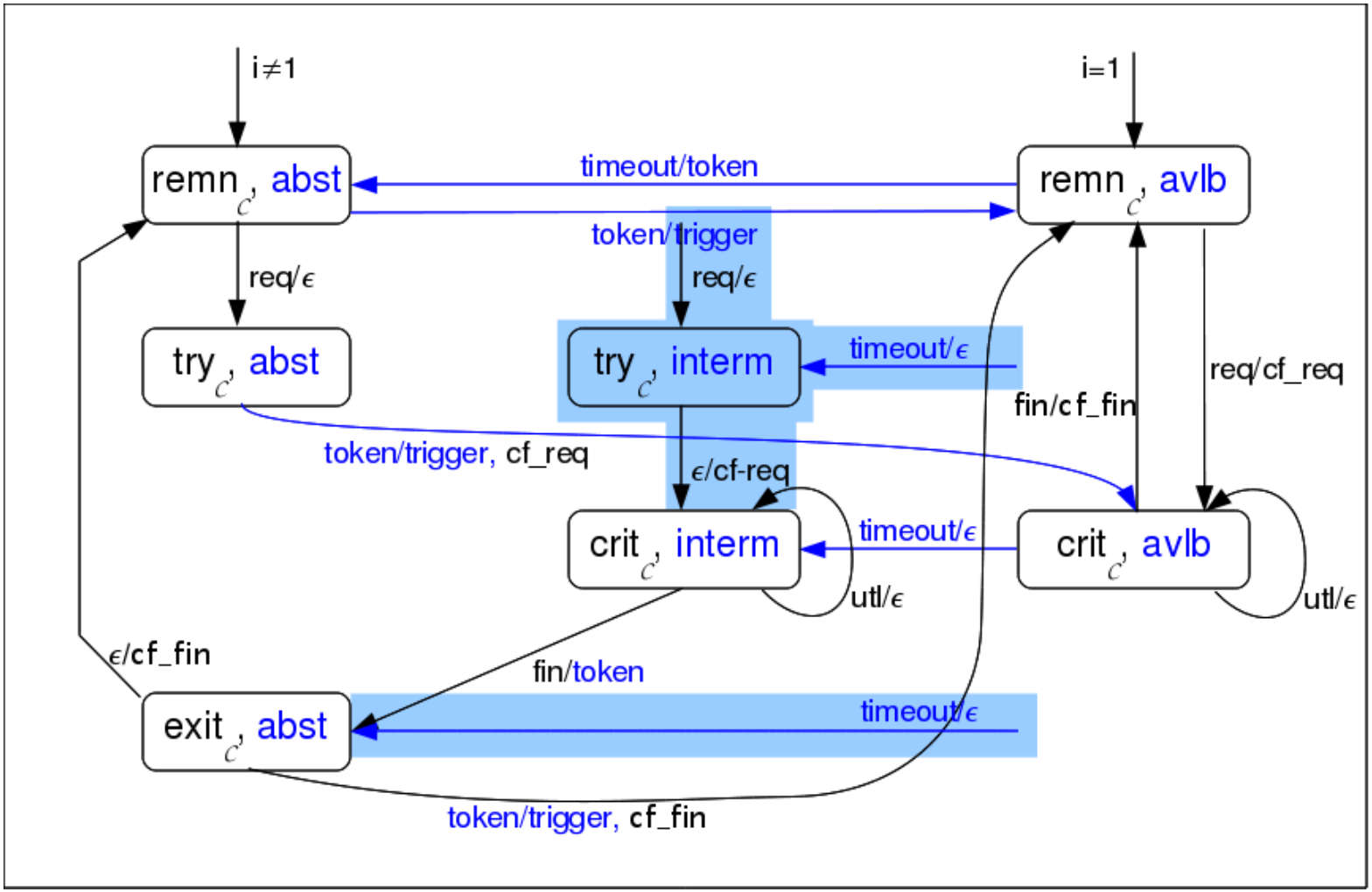
Determination, Step 1



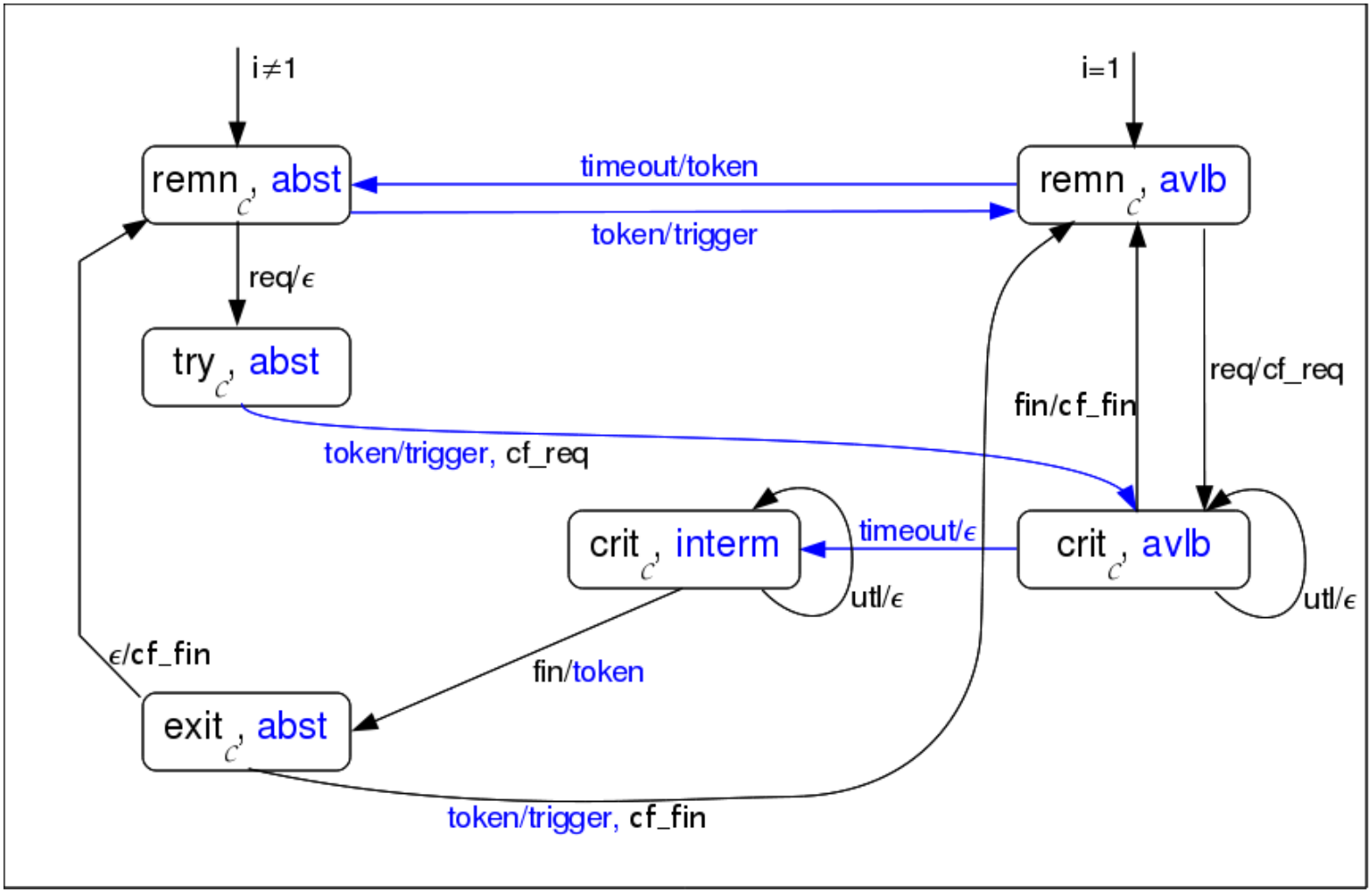
Determination, Step 1



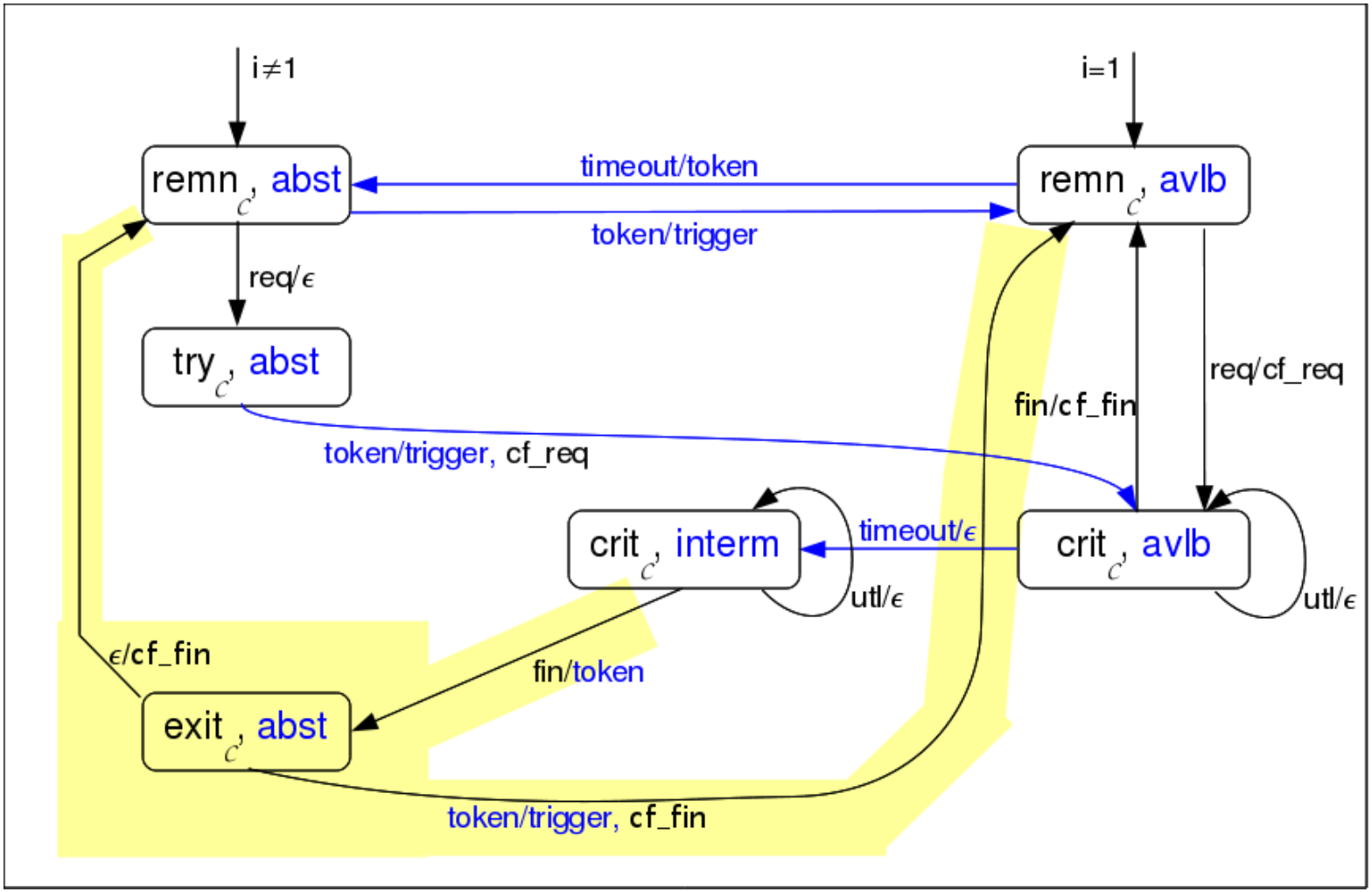
Determination, Step 2



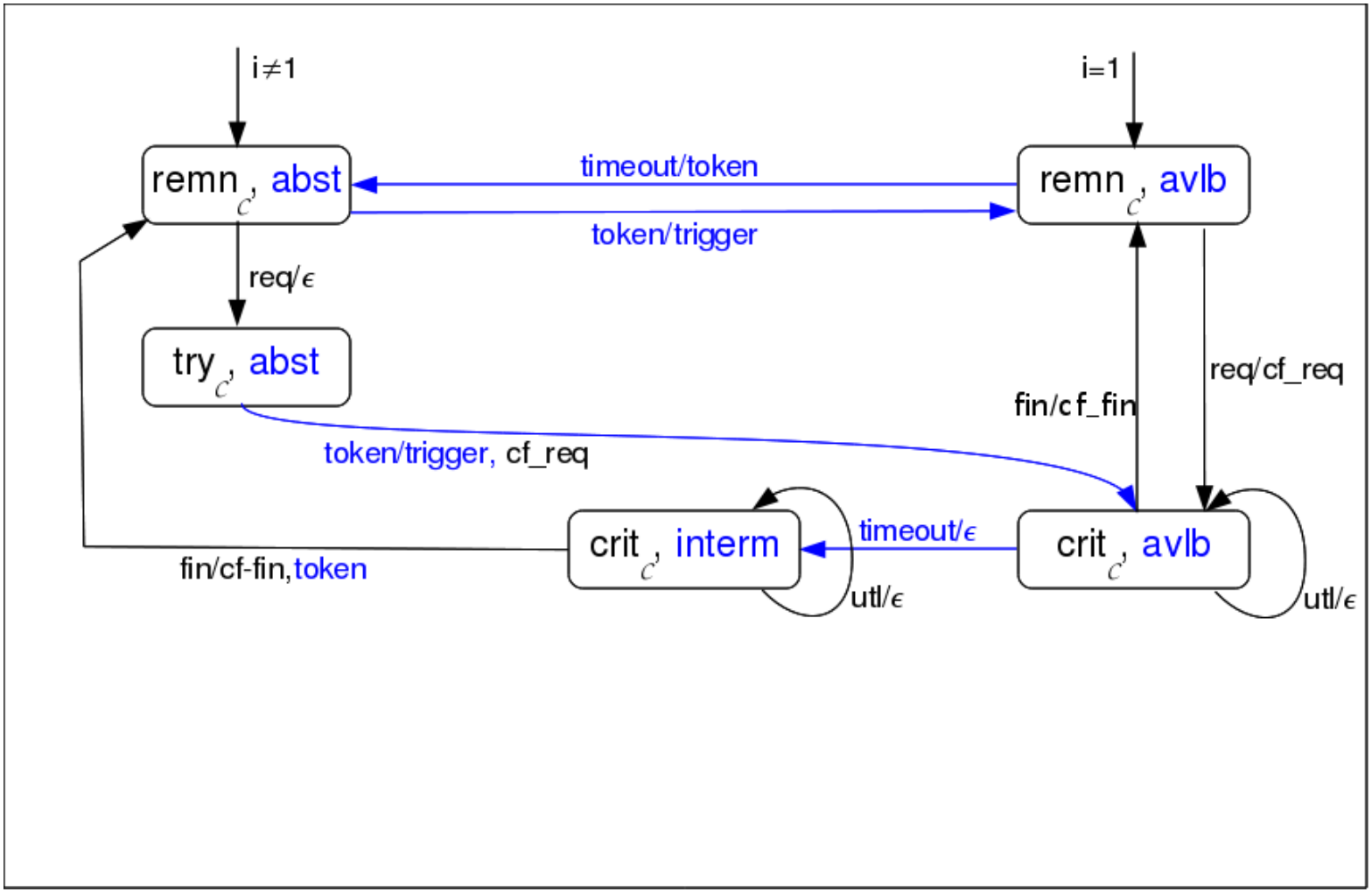
Determination, Step 2



Determination, Step 3



Determination, Step 3



Processes

Def.: Be $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ a set of $n \geq 2$ consistent protocols with a role $\mathcal{A}_{k,i}$ in each protocol P_i which specifies the behavior of a projection of a single system (symbolized by the common index k) such that for all $i \neq j$, $\mathcal{A}_{k,i} \not\subseteq \mathcal{A}_{k,j}$. A fully synchronized product of these roles $\mathcal{R} = \bigotimes_{i=1}^{n} \mathcal{A}_{k,i} \setminus \Delta^e$ is called a **process**.

Thereby a process is again an NFIOA, but - based on its construction - with the important property, that it coordinates at least two different roles. In the special case that a DFIOA results, possible after additional ϵ-merge, a process specifies a finite system.

Summary

- Computer science is a subspecialty of system theory.
- Process theory is more complex than I have thought.
- Network-like interactions are not deterministic, but are nondeterministic. The actions are not determined by the interactions!
- Nondeterminism is not introduced to abstract from details of implementation (C. Hoare, 1985) or only for convenience (N. Lynch, 1996), but nondeterminism seems to be the precondition for an efficient description of interactions between many different systems, where all interactions happen on the same level of abstraction.
- "Outer Synchronization" represents causal relation between the input and output of different systems: a sender and a receiver
- "Inner Synchronization" represents causal relation between the input and output of the same system.

Possible Consequences

- What is "loose coupling"? Abstraction from algorithm (Doug Kaye, 2003) versus sensible interaction of complex systems with few common states (Robert B. Glassmann, 1973)
- Imperative programming languages are formal calculi to describe systems and their interactions. "Soundness": every correct formal description describes an interacting system. "Completeness": every system and interaction can be described.
- Protocols are genuine entities of their own. Their implementation should be much better supported by programming languages ("implements protocol"-annotations, language support for document declarations like data type declarations, ...)
- We can start to think about a "process oriented architecture"

Thank You!

Any questions?

Johannes.Reich@sophoscape.de