The set of all such finite lists is denoted by $L$. We denote a finite list of modifications by $\vec{r} = \langle r_0, r_1, \ldots, r_r \rangle$. For each alphabet $\alpha$, $\beta$ the set of all modifications is denoted by $\text{Mod}(\alpha, \beta)$. The set of all internal actions is denoted by $\text{Chan}$. For each alphabet $\alpha$, $\beta$ the corresponding action set is denoted by $\text{Act}(\alpha, \beta)$. A set $V(\mathcal{R})$ of modifications is the set of all such finite lists of modifications.

By $\cdot$ we denote the set of all modifications.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.

For each channel $\tau$, a set $\text{Chan}$ of channel names and actions is the set of all visible actions $\text{Chan}$ and not visible from outside.
That is, modifications are executed sequentially from left to right.

\[ \mathbf{i} = \mathbf{ℓ} \]

There are edges \( \mathbf{i} \mathbf{j} \) such that \( \mathbf{i} \mathbf{j} \in E \) and \( \mathbf{i} \mathbf{j} \in E^\ast \).

A valuation \( V \) from location \( \mathbf{ℓ} \) to location \( \mathbf{j} \) is a valuation of the variables, \( V \) is a valuation of the variables, and integer expressions from \( D \) are edges \( \mathbf{t} \mathbf{u} \) such that \( \mathbf{t} \mathbf{u} \in E \) and \( \mathbf{t} \mathbf{u} \in E^\ast \).

\[ \mathbf{t} \mathbf{u} \in E^\ast \]

A valuation \( V \) from location \( \mathbf{ℓ} \) to location \( \mathbf{j} \) is a valuation of the variables, \( V \) is a valuation of the variables, and integer expressions from \( D \) are directed edges \( \mathbf{t} \mathbf{u} \) such that \( \mathbf{t} \mathbf{u} \in E \) and \( \mathbf{t} \mathbf{u} \in E^\ast \).

\[ \mathbf{t} \mathbf{u} \in E^\ast \]

A valuation \( V \) from location \( \mathbf{ℓ} \) to location \( \mathbf{j} \) is a valuation of the variables, \( V \) is a valuation of the variables, and integer expressions from \( D \) are directed edges \( \mathbf{t} \mathbf{u} \) such that \( \mathbf{t} \mathbf{u} \in E \) and \( \mathbf{t} \mathbf{u} \in E^\ast \).

\[ \mathbf{t} \mathbf{u} \in E^\ast \]

A valuation \( V \) from location \( \mathbf{ℓ} \) to location \( \mathbf{j} \) is a valuation of the variables, \( V \) is a valuation of the variables, and integer expressions from \( D \) are directed edges \( \mathbf{t} \mathbf{u} \) such that \( \mathbf{t} \mathbf{u} \in E \) and \( \mathbf{t} \mathbf{u} \in E^\ast \).

\[ \mathbf{t} \mathbf{u} \in E^\ast \]
The structural model just consists of the set of variables and the locations of the model. The computation path of the model is said to have a deadlock if and only if there is a valuation function such that for any configuration of the model, the function returns true for the deadlock condition. The (possibly partial) function is to decide whether for a network the model-checking problem is decidable. The network is a finite description of (finite or infinite) sets of a (possibly partial) function. The network is a computation path of the form

\[ C \cdot \ell \cdot \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

where \( C \cdot \ell \cdot \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \) is to decide whether any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

is reachable. Any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

is any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

The network is said to have a transition sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

where \( \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \) is any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

and

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

is any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

The network is reachable if and only if there is a valuation function such that for any configuration of the model, the function returns true for the reachability condition.

The network is said to have a transition sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

where \( \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \) is any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

and

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

is any (in)finite sequence of the form

\[ \ell, \nu \cdot \pi \cdot \llbracket \cdot \rrbracket \]

The network is reachable if and only if there is a valuation function such that for any configuration of the model, the function returns true for the reachability condition.
Recall the two examples.

Would be too easy...

How are we supposed to implement that?

• There is non-determinism in the upper automaton,
  • internal transitions can interleave, one interleaving leads to a deadlock.

• We are not!

• We define deterministic CFA, a greedy semantics for internal transitions, and only implement deterministic CFA using the greedy semantics.
Greedy CF A Semantics

The communicating finite automaton $L, B, V, E, \ell = (A, E, \ell, I, F, \lambda, vini)$ is called Greedy CF A semantics if and only if there is no edge with an input action starting at $E$.

1. $vini \geq 0$ and $\ell$.

2. $vini > 0$ and $\ell$.

3. $vini < 0$ and $\ell$.

Example.

Let each automaton in the network

- have pairwise (logically) disjoint guards.
- have pairwise different as source location have pairwise different.
- have locally deadlock-free.
- be deterministic.

The communicating finite automaton $L, B, V, E, \ell = (A, E, \ell, I, F, \lambda, vini)$ is called

1. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

2. Greedy CF A Semantics: cannot be extended by an internal transition.

3. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

Let each automaton in the network

- be deterministic.
- have pairwise (logically) disjoint guards.
- have pairwise different as source location have pairwise different.

The communicating finite automaton $L, B, V, E, \ell = (A, E, \ell, I, F, \lambda, vini)$ is called

1. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

2. Greedy CF A Semantics: cannot be extended by an internal transition.

3. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

Let each automaton in the network

- be deterministic.
- have pairwise (logically) disjoint guards.
- have pairwise different as source location have pairwise different.

The communicating finite automaton $L, B, V, E, \ell = (A, E, \ell, I, F, \lambda, vini)$ is called

1. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

2. Greedy CF A Semantics: cannot be extended by an internal transition.

3. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

Let each automaton in the network

- be deterministic.
- have pairwise (logically) disjoint guards.
- have pairwise different as source location have pairwise different.

The communicating finite automaton $L, B, V, E, \ell = (A, E, \ell, I, F, \lambda, vini)$ is called

1. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.

2. Greedy CF A Semantics: cannot be extended by an internal transition.

3. Deterministic CF A semantics: implementable (i) and (ii) can be checked syntactically.