
Partial order reduction

Consider a discrete-state model $(\mathcal{S}, \mathcal{X}_{init}, \mathcal{E}, \{\mathcal{N}_\alpha : \alpha \in \mathcal{E}\})$ where

- \mathcal{S} is the finite set of model states (we can think of either \mathcal{X}_{reach} or \mathcal{X}_{pot})
- $\mathcal{X}_{init} \in \mathcal{S}$ is the set of initial states
- \mathcal{E} is the finite set of model events
- $\mathcal{N}_\alpha : \mathcal{S} \rightarrow \mathcal{S} \cup \{\text{null}\}$ is the **deterministic** next-state function for event $\alpha \in \mathcal{E}$

Event α is enabled in state \mathbf{i} iff $\exists \mathbf{j} \in \mathcal{S}$ s.t. $\mathcal{N}_\alpha(\mathbf{i}) = \mathbf{j}$, we also write this as $\mathbf{i} \xrightarrow{\alpha} \mathbf{j}$

Event α is disabled in state \mathbf{i} iff $\mathcal{N}_\alpha(\mathbf{i}) = \text{null}$

Let $Enabled(\mathbf{i}) = \{\alpha \in \mathcal{E} : \mathcal{N}_\alpha(\mathbf{i}) \neq \text{null}\}$ be the set of events enabled in state $\mathbf{i} \in \mathcal{S}$

The **transition relation** $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ is defined by $\mathcal{N}(\mathbf{i}) = \{\mathbf{j} \in \mathcal{S} : \exists \alpha \in \mathcal{E}, \mathcal{N}_\alpha(\mathbf{i}) = \mathbf{j}\}$

We consider (finite or infinite) evolution paths $\mathbf{i}^{(0)} \xrightarrow{\alpha_0} \mathbf{i}^{(1)} \xrightarrow{\alpha_1} \mathbf{i}^{(2)} \xrightarrow{\alpha_2} \dots$

In concurrent systems, the **state-space explosion** is largely due to the **interleaving** of events

Consider a system with n bits initially set to 0, where each can be set to 1 independently of the others

- there are 2^n states
- any of the $n!$ different sequences of events leads to the same state where all n bits are 1

Q: Can we just examine one of these sequences?

A: It depends on the questions we ask of the model!

Reduced state graph

Build a **reduced state graph** $(\mathcal{X}_{red}, \mathcal{N}_{red})$ with fewer states and state-to-state transitions than $(\mathcal{S}, \mathcal{N})$

GenerateReducedStateGraph()

```

global  $\mathcal{X}_{red} \leftarrow \emptyset$  : set of state;
global  $\mathcal{N}_{red} \leftarrow \emptyset$  : set of (state,event,state);
global  $\mathcal{X}_{stack} \leftarrow \emptyset$  : stack of state;
1 foreach  $\mathbf{i} \in \mathcal{X}_{init}$  do
2   ExpandState( $\mathbf{i}$ );

```

ExpandState(\mathbf{i} : state)

uses depth-first search

```

local  $\mathcal{W} \leftarrow \textit{Ample}(\mathbf{i})$  : event set;

```

how to define $\textit{Ample}(\mathbf{i}) \subseteq \textit{Enabled}(\mathbf{i})$?

```

local  $\mathbf{j}$  : state;

```

```

1  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{i}\}$ ;

```

```

2 Push( $\mathbf{i}, \mathcal{X}_{stack}$ );

```

```

3 while  $\mathcal{W} \neq \emptyset$  do

```

```

4   choose an event  $\alpha \in \mathcal{W}$  and remove it from  $\mathcal{W}$ ;

```

```

5    $\mathbf{j} \leftarrow \mathcal{N}_\alpha(\mathbf{i})$ ;

```

```

6   if  $\mathbf{j} \notin \mathcal{S}$  then

```

\mathbf{j} is a new state, it needs to be expanded

```

7     ExpandState( $\mathbf{j}$ );

```

```

8      $\mathcal{N}_{red} \leftarrow \mathcal{N}_{red} \cup \{(\mathbf{i}, \alpha, \mathbf{j})\}$ ;

```

```

9 end while;

```

```

10 Pop( $\mathcal{X}_{stack}$ );

```

The reduced state graph $(\mathcal{X}_{red}, \mathcal{N}_{red})$ contains a subset of the behaviors of the full model $(\mathcal{S}, \mathcal{N})$

If $Ample(\mathbf{i})$ is properly defined, we can model check $(\mathcal{X}_{red}, \mathcal{N}_{red})$ instead of $(\mathcal{S}, \mathcal{N})$

Issues:

- $Ample(\mathbf{i})$ must be “large enough” for the approach to be correct
- Using $Ample(\mathbf{i})$ instead of $Enabled(\mathbf{i})$ should result in a much smaller state graph
- $Ample(\mathbf{i})$ should be easily and efficiently computable

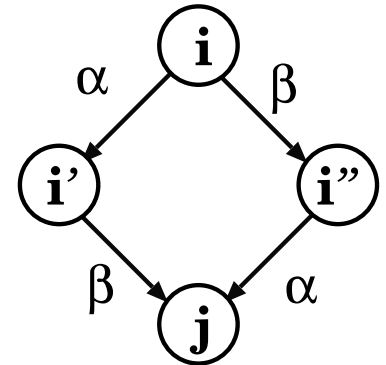
Define a symmetric antireflexive **independence** relation $\mathcal{I} \subset \mathcal{E} \times \mathcal{E}$ s.t. $\forall \mathbf{i} \in \mathcal{S}$ and $\forall (\alpha, \beta) \in \mathcal{I}$:

- **Enabledness**

If $\alpha, \beta \in \text{Enabled}(\mathbf{i})$, then $\alpha \in \text{Enabled}(\mathcal{N}_\beta(\mathbf{i}))$ and, by symmetry, $\beta \in \text{Enabled}(\mathcal{N}_\alpha(\mathbf{i}))$
 \Rightarrow α and β can never disable each other... ...but they are allowed to enable each other

- **Commutativity**

If $\alpha, \beta \in \text{Enabled}(\mathbf{i})$, $\mathcal{N}_\alpha(\mathcal{N}_\beta(\mathbf{i})) = \mathcal{N}_\beta(\mathcal{N}_\alpha(\mathbf{i}))$
 \Rightarrow applying α and β in any order leads to the same state: **diamond rule**



The **dependence** relation is the complement: $\mathcal{D} = (\mathcal{E} \times \mathcal{E}) \setminus \mathcal{I}$

Q: If α and β are independent, can we just arbitrarily apply them in a single given order?

A: No, because:

- The temporal logic property might depend on whether we follow $\mathbf{i} \xrightarrow{\alpha} \mathbf{i}' \xrightarrow{\beta} \mathbf{j}$ or $\mathbf{i} \xrightarrow{\beta} \mathbf{i}'' \xrightarrow{\alpha} \mathbf{j}$
- State \mathbf{i}' or \mathbf{i}'' could have other successors in addition to \mathbf{j} , which may need to be explored

Let \mathcal{A} be the set of atomic proposition and $\mathcal{L} : \mathcal{S} \rightarrow 2^{\mathcal{A}}$ the labeling function for the states

An event $\alpha \in \mathcal{E}$ is **invisible** w.r.t. a set of atomic propositions $\mathcal{A}' \subseteq \mathcal{A}$ if

$$\forall \mathbf{i}, \mathbf{j} \in \mathcal{S}, \quad \mathbf{j} = \mathcal{N}_{\alpha}(\mathbf{i}) \Rightarrow \mathcal{L}(\mathbf{i}) \cap \mathcal{A}' = \mathcal{L}(\mathbf{j}) \cap \mathcal{A}'$$

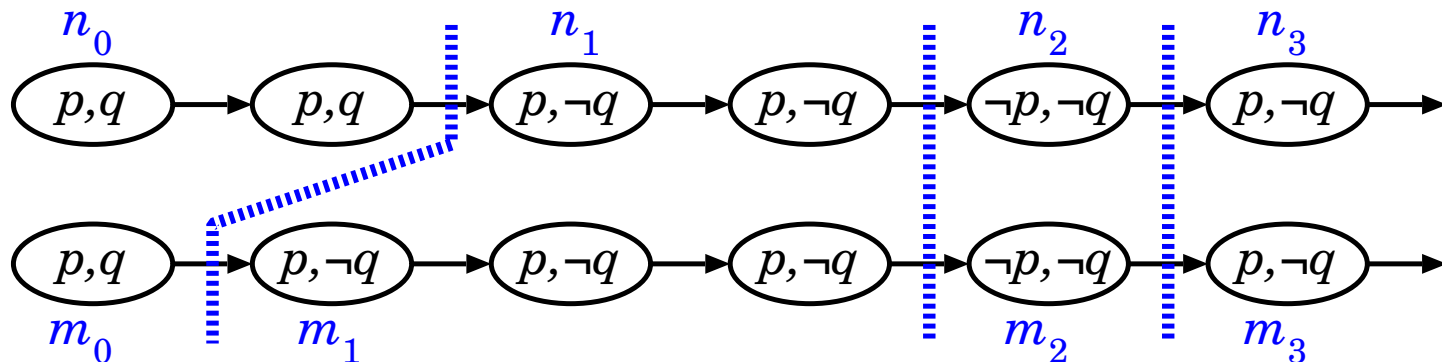
In other words, if the occurrence of α cannot change the value of the atomic propositions in \mathcal{A}'

An event is **visible** w.r.t. \mathcal{A}' if it is not invisible

Infinite paths $\sigma = \mathbf{i}^{(0)} \xrightarrow{\alpha_0} \mathbf{i}^{(1)} \xrightarrow{\alpha_1} \dots$ and $\rho = \mathbf{j}^{(0)} \xrightarrow{\beta_0} \mathbf{j}^{(1)} \xrightarrow{\beta_1} \dots$ are **stuttering** equivalent, $\sigma \sim_{st} \rho$, if:

there are two infinite sequences $0 = n_0 < n_1 < \dots$ and $0 = m_0 < m_1 < \dots$ s.t.

$$\forall k \geq 0, \quad \mathcal{L}(\mathbf{i}_{n_k}) = \mathcal{L}(\mathbf{i}_{n_k+1}) = \dots = \mathcal{L}(\mathbf{i}_{n_{k+1}-1}) = \mathcal{L}(\mathbf{j}_{m_k}) = \mathcal{L}(\mathbf{j}_{m_k+1}) = \dots = \mathcal{L}(\mathbf{j}_{m_{k+1}-1})$$



An LTL formula Af is **invariant under stuttering** iff $\forall \sigma, \rho, \sigma \sim_{st} \rho \Rightarrow (\sigma \models f \Leftrightarrow \rho \models f)$

Let LTL_{-X} be the subset of LTL formulas that do not contain the next-step “X” operator

Theorem: Any LTL_{-X} formula is invariant under stuttering

Theorem: Any LTL formula that is invariant under stuttering can be expressed in LTL_{-X}

Two discrete state models M and M' are **stuttering equivalent** iff

- M and M' have the same set of initial states \mathcal{X}_{init}
- For any path σ of M starting from $\mathbf{i} \in \mathcal{X}_{init}$, there is a path σ' of M' from \mathbf{i} s.t. $\sigma \sim_{st} \sigma'$
- For any path σ' of M' starting from $\mathbf{i} \in \mathcal{X}_{init}$, there is a path σ of M from \mathbf{i} s.t. $\sigma' \sim_{st} \sigma$

Theorem: if M and M' are stuttering equivalent, for any LTL_{-X} formula Af and state $\mathbf{i} \in \mathcal{X}_{init}$:

$$M, \mathbf{i} \models Af \Leftrightarrow M', \mathbf{i} \models Af$$

Let's build $Ample(\mathbf{i})$ in such a way that $(\mathcal{X}_{red}, \mathcal{N}_{red})$ is stuttering equivalent to $(\mathcal{S}, \mathcal{N})$

A state \mathbf{i} is **fully expanded** if $Ample(\mathbf{i}) = Enabled(\mathbf{i})$

If we fully expand every state, $(\mathcal{X}_{red}, \mathcal{N}_{red}) = (\mathcal{S}, \mathcal{N})$, thus they are trivially stuttering equivalent

But we want to do better than that

Given an $LTL_{\neg X}$ formula Af with atomic propositions \mathcal{A}' , $Ample(\mathbf{i})$ must satisfy:

- **C0:** $Ample(\mathbf{i}) = \emptyset$ iff $Enabled(\mathbf{i}) = \emptyset$
- **C1:** For any path in $(\mathcal{S}, \mathcal{N})$ starting at \mathbf{i} and for any event α that depends on $\beta \in Ample(\mathbf{i})$, α can occur along the path only after an event in $Ample(\mathbf{i})$ occurs
- **C2:** If \mathbf{i} is not fully expanded, every $\alpha \in Ample(\mathbf{i})$ is invisible w.r.t. \mathcal{A}'
- **C3:** $(\mathcal{X}_{red}, \mathcal{N}_{red})$ contains no cycle \mathcal{C} where α is enabled but not in $Ample(\mathbf{i})$ for any $\mathbf{i} \in \mathcal{C}$

Theorem: If C1 holds, events in $Enabled(\mathbf{i}) \setminus Ample(\mathbf{i})$ are independent of events in $Ample(\mathbf{i})$

Proof: Assume $\gamma \in Enabled(\mathbf{i}) \setminus Ample(\mathbf{i})$ and $\delta \in Ample(\mathbf{i})$, s.t. γ and δ are dependent.

Then, there is a path from \mathbf{i} starting with γ , contradicting C1

\Rightarrow it's OK to postpone the firing of any event $\alpha \in Enabled(\mathbf{i}) \setminus Ample(\mathbf{i})$

If C1 holds, any path in $(\mathcal{S}, \mathcal{N})$ starting from \mathbf{i} has one of these two forms:

(P1) It starts with a prefix $\beta_1\beta_1 \cdots \beta_m\alpha$, where each β_k , for $1 \leq k \leq m$, is independent of each event in $Ample(\mathbf{i})$, and $\alpha \in Ample(\mathbf{i})$

(P2) It is an infinite sequence $\beta_1\beta_2 \cdots$, where each β_k is independent of each event in $Ample(\mathbf{i})$

If C1 holds and $\mathbf{i} \xrightarrow{\beta_1} \cdots \xrightarrow{\beta_m} \mathbf{j}$ and β_1, \dots, β_m are not in $Ample(\mathbf{i})$, then $Ample(\mathbf{i}) \subseteq Enabled(\mathbf{j})$

(P1): the path $\mathbf{i} \equiv \mathbf{i}^{(0)} \xrightarrow{\beta_1} \mathbf{i}^{(1)} \xrightarrow{\beta_2} \cdots \xrightarrow{\beta_m} \mathbf{i}^{(m)} \xrightarrow{\alpha} \mathbf{j}$ **is not** in $(\mathcal{X}_{red}, \mathcal{N}_{red})$ but, if C2 holds,

this path can be ignored, since it is stuttering equivalent to $\mathbf{i} \xrightarrow{\alpha} \mathbf{j}^{(1)} \xrightarrow{\beta_1} \mathbf{j}^{(2)} \xrightarrow{\beta_2} \cdots \xrightarrow{\beta_m} \mathbf{j}^{(m)} \equiv \mathbf{j}$

which either **is** in $(\mathcal{X}_{red}, \mathcal{N}_{red})$ or has a stuttering equivalent path in $(\mathcal{X}_{red}, \mathcal{N}_{red})$

(P2): the path $\mathbf{i} \equiv \mathbf{i}^{(0)} \xrightarrow{\beta_1} \mathbf{i}^{(1)} \xrightarrow{\beta_2} \cdots$ **is not** in $(\mathcal{X}_{red}, \mathcal{N}_{red})$ but, if C2 holds,

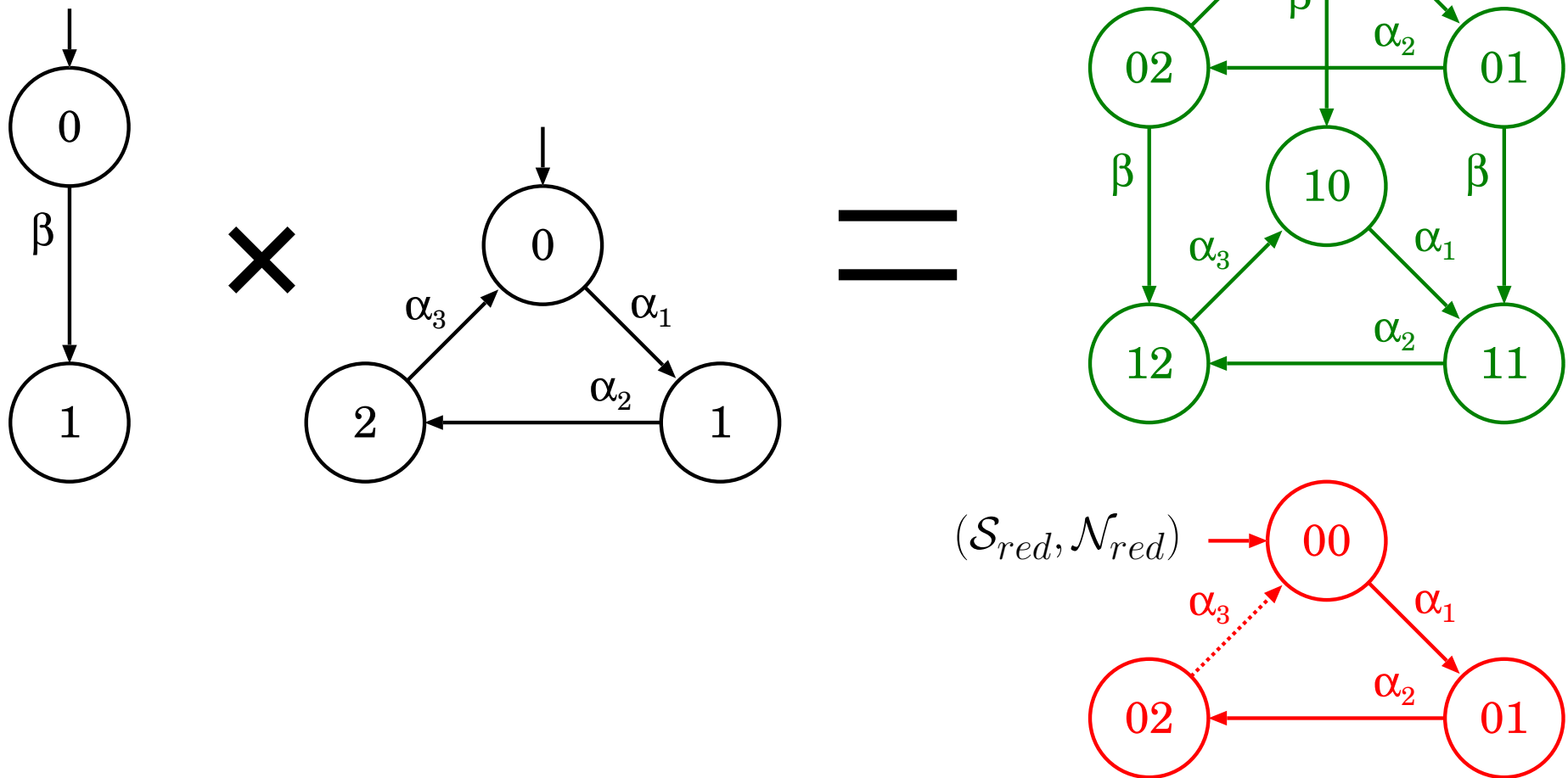
this path can be ignored, since it is stuttering equivalent to $\mathbf{i} \xrightarrow{\alpha} \mathbf{j}^{(1)} \xrightarrow{\beta_1} \mathbf{j}^{(2)} \xrightarrow{\beta_2} \cdots$

which either **is** in $(\mathcal{X}_{red}, \mathcal{N}_{red})$ or has a stuttering equivalent path in $(\mathcal{X}_{red}, \mathcal{N}_{red})$

There is still a problem: a visible event β may be delayed forever

Consider a model $(\mathcal{S}, \mathcal{N})$ where α_1 , α_2 , and α_3 are invisible, but β is visible

$(\mathcal{S}, \mathcal{N})$ is the parallel composition of two smaller systems



To prevent this possibility, we require C3 to hold

How do we check for conditions C0, C1, C2, and C3 in practice?

- **C0:** $Ample(\mathbf{i}) = \emptyset$ iff $Enabled(\mathbf{i}) = \emptyset$

C0 can be easily checked

- **C1:** For any path in $(\mathcal{S}, \mathcal{N})$ starting at \mathbf{i} and for any event α that depends on $\beta \in Ample(\mathbf{i})$, α can occur along the path only after an event in $Ample(\mathbf{i})$ occurs

C1 is very difficult to check, it refers to future states which might not even be in \mathcal{X}_{red}

Theorem: checking condition C1 for a state \mathbf{i} and a set $\mathcal{E}' \subseteq Enabled(\mathbf{i})$ is at least as hard as checking the reachability of an arbitrary state in $(\mathcal{S}, \mathcal{N})$

\Rightarrow use safe, efficient, but sub-optimal heuristics to compute $Ample(\mathbf{i})$

- **C2:** If \mathbf{i} is not fully expanded, every $\alpha \in Ample(\mathbf{i})$ is invisible w.r.t. \mathcal{A}'

C2 can be easily checked

- **C3:** $(\mathcal{X}_{red}, \mathcal{N}_{red})$ contains no cycle \mathcal{C} where α is enabled but not in $Ample(\mathbf{i})$ for any $\mathbf{i} \in \mathcal{C}$

C3 is expensive to check, even it refers just to cycles in $(\mathcal{X}_{red}, \mathcal{N}_{red})$, not in $(\mathcal{S}, \mathcal{N})$

Lemma: C3 holds if at least one state is fully expanded in each cycle

\Rightarrow use a stronger but easier-to-check condition than C3

- **C3':** if state \mathbf{i} is not fully expanded, no event in $Ample(\mathbf{i})$ can reach a state on \mathcal{X}_{stack}

C3' can be easily checked

Heuristics to compute $Ample(\mathbf{i})$ have been proposed for various formalisms, including

- Petri nets
- Communicating Sequential Processes (CSP)

Let's see a heuristic for concurrent/distributed software models where:

- there are M processes P_1, \dots, P_M
- each process has a program counter, let $pc_m(\mathbf{i})$ be the program counter of P_m in global state \mathbf{i}
- processes can communicate via **shared variables** or **rendezvous** or **buffered message passing**

Definitions of important sets of events

- $Pre(\alpha) = \{\beta : \exists \mathbf{i}, \alpha \notin Enabled(\mathbf{i}) \wedge \alpha \in Enabled(\mathcal{N}_\beta(\mathbf{i}))\}$, the events that may enable α
- $Dep(\alpha) = \{\beta : (\beta, \alpha) \notin \mathcal{I}\}$, the events that depend on α
- \mathcal{E}_m , the events of process P_m
- $\mathcal{E}_m(\mathbf{i}) = \mathcal{E}_m \cap Enabled(\mathbf{i})$, the events of process P_m enabled in global state \mathbf{i}
- $Current_m(\mathbf{i}) \supseteq \mathcal{E}_m(\mathbf{i})$, the events of P_m enabled in some state \mathbf{j} s.t. $pc_m(\mathbf{i}) = pc_m(\mathbf{j})$

The particular modeling formalism determines how easily we can build these sets of events

To satisfy C0: let $Ample(\mathbf{i}) = \mathcal{E}_m(\mathbf{i})$, for some m s.t. $\mathcal{E}_m(\mathbf{i}) \neq \emptyset$, if one exists

All events in $\mathcal{E}_m(\mathbf{i})$ are dependent on each other, as they change $pc_m(\mathbf{i})$ and local variables

C1 is violated if a sequence σ of events independent of $\mathcal{E}_m(\mathbf{i})$ can be executed in $(\mathcal{S}, \mathcal{N})$, s.t. σ eventually enables an event α dependent on some event in $\mathcal{E}_m(\mathbf{i})$

We need to ensure that condition C1 holds:

- the events of σ cannot be in $\mathcal{E}_m(\mathbf{i})$, since all events in $\mathcal{E}_m(\mathbf{i})$ are dependent on each other
- consider two possibilities: either $\alpha \in \mathcal{E}_n$ for some $n \neq m$, or $\alpha \in \mathcal{E}_m$
- if $\alpha \in \mathcal{E}_n$ for some $n \neq m$, then $Dep(\mathcal{E}_m(\mathbf{i}))$ contains a transition of P_n
 \Rightarrow **C1'**: check that $Dep(\mathcal{E}_m(\mathbf{i})) \subseteq \mathcal{E}_m$
- if $\alpha \in \mathcal{E}_m$, let \mathbf{j} the state in which it is executed
 then $\mathbf{i} \xrightarrow{\sigma} \mathbf{j}$ and all the events in σ are independent of $\mathcal{E}_m(\mathbf{i})$, thus belong to other processes
 then, $pc_m(\mathbf{i}) = pc_m(\mathbf{j})$ and $\alpha \in Current_m(\mathbf{i})$ but $\alpha \notin \mathcal{E}_m(\mathbf{i})$, thus $\alpha \in Current_m(\mathbf{i}) \setminus \mathcal{E}_m(\mathbf{i})$
 also, $\alpha \notin Enabled(\mathbf{i})$, thus σ contains an event in $Pre(\alpha)$
 \Rightarrow **C1''**: check that $Pre(Current_m(\mathbf{i}) \setminus \mathcal{E}_m(\mathbf{i})) \subseteq \mathcal{E}_m$

If there is a process P_m for which $\mathcal{E}_m(\mathbf{i})$ satisfies C0, C1', C1'', C2, and C3', let $Ample(\mathbf{i})$ be $\mathcal{E}_m(\mathbf{i})$

Otherwise, be conservative and fully expand state \mathbf{i} , i.e., let $Ample(\mathbf{i})$ be $Enabled(\mathbf{i})$