

Interleaving

Given two automata, we have a choice about how to compose them. Let

$$\begin{aligned}\mathcal{A} &= \langle A, a_0, R \subseteq A \times A, L: A \rightarrow P \rangle \\ \mathcal{B} &= \langle B, b_0, T \subseteq B \times B, R: B \rightarrow Q \rangle\end{aligned}$$

We can form a *product automaton*, $\mathcal{A} \times \mathcal{B}$ whose states are pairs (a, b) from $A \times B$, with initial state (a_0, b_0) . A state (a, b) has labeling $L(a) \cup L(b)$, supposing that P and Q are disjoint sets (If not, we need to make special provisions for contradictory labels).

It remains to determine what the transition relation should be for $\mathcal{A} \times \mathcal{B}$, and SMV provides two forms:

- *Synchronous interleaving.* There is an edge from (a, b) to (a', b') in if and only if both corresponding edges $(a, a') \in R$ and $(b, b') \in T$ are present in \mathcal{A} and \mathcal{B} .
- *Asynchronous interleaving.* Whenever there is an edge $(a, a') \in R$ there are edges from (a, b) to (a', b) for all $b \in B$, in $\mathcal{A} \times \mathcal{B}$; and whenever there is an edge $(b, b') \in T$ there are edges from (a, b) to (a, b') for all $a \in A$, in $\mathcal{A} \times \mathcal{B}$.

The sense of a synchronous interleaving is that both component automata make a simultaneous transition. The sense of an asynchronous interleaving is that one of the component automata makes a transition while the other remains in its current state.

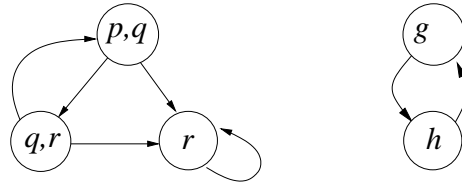
Fairness

Most applications of model checking to software are based on asynchronous interleaving. They model the idea that it is indeterminate how many instructions a process may execute while it is in control. However, one usually wants to exclude the case when some process is indefinitely blocked from execution any of its instructions. This notion that every process gets a chance to execute is called *fairness*. It has many variations.

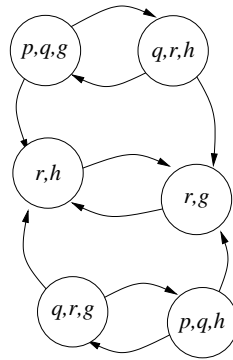
The FAIRNESS clause in SMV, when included, says that the model checker should consider only those paths in which the fairness condition holds infinitely often—or equivalently, only those cycles in the model are considered where the condition holds at least once. The special condition **running** says that a process inevitably takes a step.

However, it is difficult to capture in SMV every notion of fairness that has arisen in the research literature, and it continues to be a topic of interest.

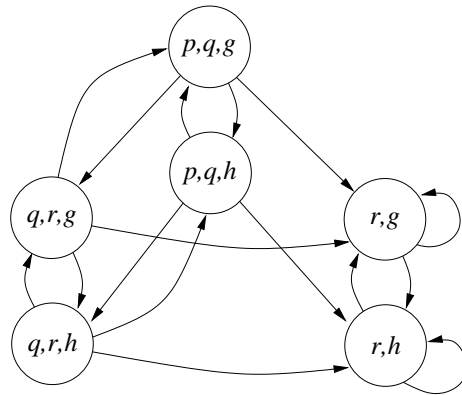
First Example



Synchronous Interleaving

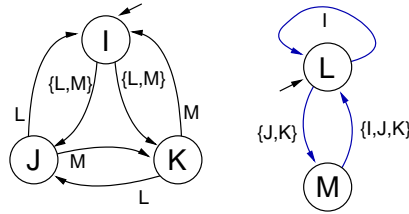


Asynchronous Interleaving

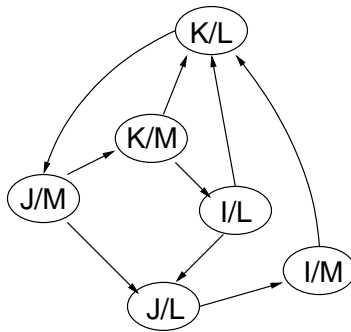


Second Example

In this example, the edge labels say that one process can make a transition when the other process is in the named state. This is more suggestive of the modeling expressions in SMV. A process that is not enabled to make a transition is assumed to remain in the state it is in.



Synchronous Interleaving Each process takes a step.



Asynchronous Interleaving One process takes a step.

