Modularization

Modularization = split the problem into parts

Modeling \( G = G_1 \parallel G_2 \parallel G_3 \parallel \ldots \)

Specification \( Sp = Sp_1 \parallel Sp_2 \parallel Sp_3 \parallel \ldots \)

We saw examples of both in example 30.

Easier to do things in smaller steps.

Total specification is obtained with parallel composition

\[
G \parallel Sp = G \parallel (Sp_1 \parallel Sp_2 \parallel Sp_3 \parallel \ldots) = (G \parallel Sp_1) \parallel (G \parallel Sp_2) \parallel (G \parallel Sp_3) \parallel \ldots
\]

All individual specifications are met by \( Sp \).

One may also locate problems by composing each specification individually, for example \( G \parallel Sp_i = \varepsilon \) (=only the initial state) means \( Sp_i \) is too restrictive.
Example 31. Consider a resource shared by two users. A user must send a request, and get a permission to use before he may use the resource. Below an automaton for one user:

Here the event \( r(\text{equest}) \) is uncontrollable, and the events \( p(\text{ermission}) \) and \( f(\text{ree}) \) are controllable. If we compose two users together we get a shuffle (no common events):
We have the following sub-specifications

1. The two users should not access the resource simultaneously.

2. Use a first come, first served policy when both users request it.

3. See to it that p1 takes place before p2. Could be required when User1 produces something that User2 needs.

We want all three specifications to hold simultaneously, if possible, and that the users are granted repeated access to the resource.
1. First specification $S_p1$ is simple, and static. We simply forbid the state U1U2, where both users have access to the resource.
2. Second specification Sp2 means that if and only if r1 happens before r2, then p1 should happen before p2. Similar to the database example, so it could be solved using a state split. Let us first use a different strategy:
It is also possible to use a state split. In this case, we might as well remove the forbidden state here, and get $S_p12$
3. Third specification $Sp3$ is an event alternance:

A candidate supervisor is obtained if we do $Sp12 || Sp3$. $Sp12$ is a copy of the plant, so we do not need $G$. 
We get the same result if we do $Sp_1 \parallel Sp_2 \parallel Sp_3$, only the state names are different:

If we do $G \parallel Sp_12 \parallel Sp_3$ or $G \parallel Sp_1 \parallel Sp_2 \parallel Sp_3$ we get the same automata, but with longer state names. $I1I2.A$ becomes $I1I2.I1I2.A$ and so on, the plant state names become duplicated.

In practice one should include the plant $G$ in the parallel compostions and check for uncontrollable states (using a compare operation), to be defined next.