Use the B method to specify a system capable of handling hotel reservations for customers in a travel agency environment. The system has to manage the following entities: *customer*, *hotel*, *room*, *room type* and *reservation*. Next, is a brief description of each of them:

- A *customer* is recorded together with his/her *name* and some sort of contact information (*postal address*, *email address, phone number*). A *customer* corresponds to a real person.
- A *hotel* is recorded together with its *name*. Each *hotel* contains many *room*s (at least one *room).*
- A *room type* is recorded together with its *name* and *price* (a certain amount of money). Each *room type* belongs to exactly one hotel.
- A *room* is recorded together with its *number* and its *type*. There is no *room* without its *room type*. Each *room* belongs to exactly one *hotel.*
- A *reservation* is recorded together with its *reference* (some sort of human-readable code) and a *time interval* (check-in date & check-out date). A newly created *reservation* is made by exactly one *customer* (but a *customer* can make many *reservation*s) at exactly one *hotel* for exactly one *room type*. Obviously, the system can maintain many *reservation*s for the same *hotel.* Please keep in mind that a "fresh" *reservation* enters an "unallocated" state (it does not have an allocated *room).* Over time, the same *room* can be allocated to many *reservation*s but their time intervals must not intersect.

The system should provide a number of operations for creating/modifying *customer*s*, hotel*s, *room type*s, *room*s, for creating/destroying *reservation*s, for allocating a room to an already existing reservation. Next, there is a number of informally stated laws that the system should satisfy:

1. All *reservation*s that do not obey the *hotel*'s capacity (for the whole time interval) must be rejected.
2. In case of exception for the first rule, an adequate advice should be given (different ro*om type* or maybe different time interval).
3. Allocations for unavailable rooms must be rejected.

The specification should be realized and verified for consistency within AtelierB. All components must pass type-checking. It is not required to prove all proof obligations, but those not automatically proved should be analyzed for validity.