
Atomic Actions / Asynchronous Transfer of Control

✉ TO ALL STUDENTS OF ANU/CECS/SoCS/COMP4330, 6433, AND 8140

Abstract – First contact with asynchronous transfer of control methods. Depending on your background you will need to take your time understanding the basic structure and finer details of the provided example. Make sure you feel 100% comfortable with the introduced concepts before proceeding to the next assignment.

1. The background

A cascade of time-out-on-action blacks is employed to implement concurrent atomic actions as introduced in the lecture. Start by reading through the provided code carefully and vary the timing parameters and raised exceptions.

The Atomic Action is invoked by calling the procedure Perform in the package

Generic Atomic Action. When called, this procedure creates all the tasks participating in the Atomic Action, and destroys these tasks after completion. The actual actions performed by the tasks are declared and implemented as procedures in the package

Atomic Action, together with their cleanup procedures. The cleanup procedures have to be called in case of a failure, to roll back to the initial state of the Atomic Action (refer to the definition).

2. This assignment

Modify the example, so that tasks are not created inside the Atomic Action, but that existing tasks can participate. In this scenario, each of the participating tasks calls Perform with its task ID as a parameter. The ID parameter selects the corresponding action for each task, which are defined as in the given example, in the array Action.

This call returns after completion of the whole Atomic Action (all participating tasks have completed their action part), or after the cleanup is finished for all tasks, in case of an error. Make sure that all requirements for Atomic Actions are met.

Also show that your modified Atomic Action can be repeated with the same results.