Extending Abstract Behavioral Specifications with Erlang-style Error Handling[*]

Georg Göri[1], Bernhard K. Aichernig[1],
Einar Broch Johnsen[2], Rudolf Schlatte[2] and Volker Stolz[2]

[1] University of Technology, Graz, Austria
goeri@student.tugraz.at,aichernig@ist.tugraz.at
[2] University of Oslo, Norway
{einarj,rudi,stolz}@ifi.uio.no

The abstract behavioral specification (ABS) language targets distributed object-oriented systems [3], but does not currently support fault-tolerant features such as adaptability to distribution failures. This work develops an Erlang execution backend for ABS and extends ABS with error handling capabilities à la Erlang. The resulting extension of ABS combines rollback to invariant states at the object-level with Erlang style process linking and supervision.

ABS is a statically typed object-oriented modeling language targeting distributed systems [3]. ABS is based on asynchronous method calls between Concurrent Object Groups (cogs), akin to Actors and concurrent objects. An asynchronous method call creates a new process in the called object. The reply from the asynchronous call is placed in a *future*, a single-assignment global variable that can be shared between objects. Futures support retrieval and checking the availability of a reply. Whereas execution in different cogs happens in parallel, the execution of processes inside a cog is strictly interleaved and controlled by means of cooperative scheduling; i.e., explicit suspension points in the code allow the active process to be suspended and another local process to be activated. Suspension may be conditional; e.g., it can depend on the status of a future. ABS supports a proof theory for concurrent systems by means of local reasoning, based on seeing objects as monitor-like maintainers of class invariants [2]. This proof theory requires that the invariants hold at locally quiescent states; i.e., whenever a process may suspend it must ensure the invariant and whenever a process is scheduled it can assume the invariant.

Erlang is a dynamically typed functional programming language. Concurrency is done by lightweight processes which asynchronously exchange messages but do not share state [1]. Distribution with location transparent message passing is an integral part of Erlang. In addition to a standard exception mechanism, Erlang provides *process linking* to handle distribution and runtime errors. A link is a bidirectional relationship between two processes, which guarantees the delivery of an exit signal to one process in case its partner terminates or becomes unreachable.

Our mapping of ABS to Erlang follows the principle that "everything is a process". While adhering to ABS semantics it provides distribution and scalability. Each cog becomes one Erlang process which controls local scheduling such that at most one ABS process executes at a time. Each ABS process (and the main block) becomes one Erlang process which maintains the local variables. Objects become tail-recursive processes which handle field access via messages.

References

[1] Armstrong, J. *Programming Erlang.* Pragmatic Bookshelf, 2007.

[2] Chang Din, C., Dovland, J., Johnsen, E. B., and Owe, O. Observable behavior of distributed systems: Component reasoning for concurrent objects. *J. of Log. and Alg. Prog.*, 81:227–256, 2012.

[3] Johnsen, E. B., Hähnle, R., Schäfer, J., Schlatte, R., and Steffen, M. ABS: A core language for abstract behavioral specification. In *FMCO*, *LNCS* 6957, pages 142–164. Springer, 2012.

[*]Partially funded by the EU project FP7-610582 ENVISAGE: Engineering Virtualized Services (http://www.envisage-project.eu).