Report: Break Out Session on Guaranteed Execution

Calton Pu (moderator), Jim Johnson, Rogerio de Lemos, Andreas Reuter, David Taylor, Irfan Zakiuddin (scribe)

Abstract

The break out session discussed guaranteed properties during program execution. Using a workflow example application, we discussed several research topics that form part of the guaranteed properties, including declarative specifications, generation of workflow program, generation of invariant guards, automated failure analysis, automated repair, and automated reconfiguration of workflow.

Introduction. This break out session started discussion on Specifying (and Protecting) Guaranteed Properties for Execution. The guaranteed properties would be analogs or cognates of serializability, with a simple API, and still powerful enough for proving interesting properties. Examples include mutual exclusion (monitors in programs), serializability (transactions in databases), and protection against TOCTTOU exploits in file systems. Taking workflow as an example application where atomicity alone is insufficient, we discussed the composition of atomic actions and the kind of properties that can be guaranteed for composite activities. Using a travel booking scenario, we discussed several interesting and challenging research topics and their relationship to atomicity.

The first research topic is a *declarative specification of workflow*, in terms of its integrity constraints and dependencies among components. This research could build on methods such as process algebra, which compose atomic actions. We further assume these constraints can be used as the execution properties and invariants to be guaranteed during the execution of workflow. Since current workflow languages such as BPEL4WS are procedural, a declarative specification seems a very promising direction of research for both workflow and guaranteed executions.

The second research topic is a method to *generate appropriate workflow executions* based on these declarative specifications. This research could build on methods and tools such as constraint verifiers for checking integrity constraints and Colored Petri Nets for checking program execution properties. The goal is to use rigorous methods to translate a declarative specification into procedural workflow languages such as BPEL, to maintain the guaranteed properties. The resulting workflow executions should offer high confidence in guaranteeing properties during program execution.

The third research topic is to use the same specifications to *generate invariant guards* that monitor the workflow execution and detect any violations of the constraints and properties supposed to be guaranteed during execution. This research could build on methods such as event-driven architectures, continual queries, and guarding of invariants for specialized or customized system and applications software. When violations are detected, the guards notify appropriate failure analysis and recovery mechanisms (in analogy to exception handling) to preserve the invariants. These are the next steps during execution and the next research topics.

The fourth research topic is to perform *automated failure analysis* upon property/invariant violation detection. This research could build on methods such as root cause analysis and impact analysis, as well as machine learning. These methods are used to find or isolate the causes for the violation of properties being guaranteed. Once the source of failure has been identified or confined, the recovery mechanisms are invoked.

The fifth research topic is to perform *automated repair*. This research could build on methods such as atomicity (e.g., rollback and restart), roll forward, and compensations. Each of these methods applies to a subset of workflows that satisfy some invariants. For example, the rollback and restart applies to atomic actions. The compensations method applies to workflows that provide compensation actions after committing sub-workflows. These repair actions are used when applicable. After the repair mechanism has brought the system state back to a consistent state (according to the properties being guaranteed), the next step is reconfiguration and resumption.

The sixth research topic is *automated adaptation and reconfiguration*. With the information gathered from the first 5 research topics, this research could build on the methods and tools developed for the previous 5 research topics. First, we need to study the interactions between the repaired consistent state and the declarative specifications, for example, to check whether changes in the specifications are required (first research topic). Second, using the consistent state achieved by the repair methods, we can generate a new workflow by reusing the declarative specifications with a different starting point (second research topic). Third, the new starting point also implies a new set of invariants and guards, which requires the regeneration of the previous or new guards (third research topic). Fourth, we execute the newly generated set of workflow procedures and invariant guards to guarantee the properties as specified. This new execution may proceed successfully to the end, or another violation may happen, in which case we continue to the automated failure analysis, repair, and reconfiguration stages.

Each one of the research topics discussed presents very significant and challenging research problems. They also build on existing methods and tools, making them appear feasible and reasonable. In addition to the methods mentioned above, concepts and techniques discussed include operational semantics. The research topics are independent of the workflow application scenario and can be applied to other application scenarios where guaranteeing execution properties that are weaker than atomicity becomes very useful. Other example applications mentioned during the discussions include automated detection and recovery from intrusions, particularly insider attacks.