Structure Analysis for Dynamic Software Architecture

Tingting Han   Taolue Chen   Jian Lu
State Key Laboratory of Novel Software Technology, Nanjing University,
Nanjing, Jiangsu, P.R.China, 210093
{hantt,ctl,lj}@ics.nju.edu.cn

The open and dynamic Internet environment greatly urges software entities that are distributed on different locations to coordinate with each other to accomplish a computing task. *Software architecture* is applied to abstract the software entities to be components and the coordination between them to be connectors and then a model is extracted as the architecture on which the design, analysis and verification are based. Currently, the notion of *dynamic software architectures* that can modify their architecture and enact modifications during the system execution has become one of the most active research areas.

In this paper, we focus on the dynamic evolution of *system structure* other than *coordination mechanisms* (e.g. communication protocols). It is widely recognized that some restrictions should be imposed on the system evolution to ensure that the system structure may remain one style or transform within a scope. These conditions, to a large extent, make the system execute under control as expected.

Our formalization towards structural properties of graphically described software architectures mainly covers three aspects:

1) To *model the system* with a graph-based calculus. The data model is graph-based for the sake of a more natural and intuitionistic way to only record the structural information but omits the behavioral information. Comparing to many of the existing works, we explicitly separate the behavior and the topology aspects, in which one of our novelties lies. Moreover, we model private resources which is becoming aware in many of the applications by means of name hiding notions inspired by the π-calculus.

On the basis of the model, a *modification language* is provided to specialize the dynamic actions that the software architectures might perform in an explicit way. In syntax, its primitive operations include "add" components and connectors, "remove" components and connectors as well as "attach" and "detach" components and connectors. Moreover, the various ways of combining the primitive operations will lead to the changes of the architecture. The formal semantics of the language is presented in an operational style.

2) To *specify the properties* that the system should hold by spatial logic. We tailor *spatial logic* to take on this role, since it has strong expressing power for describing precisely certain properties, especially those hold at a certain location, at some location, or at every location. Concretely, we apply location formulas to perfectly cover the nested subsystems; the private communication between certain components may be dealt with by the restriction formula with modal operators ®, $\oslash$, and quantifiers И, H and recursion has been introduced into spatial logic for clearness and conciseness. Technically, the standard approach to introduce recursion into a modal logic is via fixpoint, as in μ-calculus, however such work is not trivial because of the possible interaction of the rich modalities, such as ®, $\oslash$, И and first-order quantification and the fixpoint operator. In order to deal with such problems, we make a distinction between proposition and predicate, thus these interactions can be nicely solved in the sense that a concise semantics interpretation for logic formulas can be given.

3) To *verify* whether the data model evolves to meet the specification by means of model checking. The verification and automatic detection of errors in software architectures aids to locate errors and increase the reliability of these systems.

Our work is inspired by [1] which uses spatial logic to query graphs. However, it neglects the spatial operators and quantifiers that deal with the private resources. We extend their logic in this paper. And our spatial logic is designed specially to describe dynamic software architecture. Still, a modification language other than a query language is constructed for the purpose of architecture evolution.

As future works, we intend to combine the behavior and structure analysis for software architecture in the same framework which provides a unified specification and verification approach.

[1] L. Cardelli, P. Gardner, G. Ghelli. A spatial logic for querying graphs. In 29th Colloquium on Automata, Languages and Programming (ICALP 2002), Lecture Notes in Computer Science, pages 597-610. Springer-Verlag, 2002.