Goal-Conflict Detection Based on Temporal Satisfiability Checking

Renzo Degiovanni[*], Nicolas Ricci[*], Pablo Castro[*]
[*]Departamento de Computación, Universidad Nacional de Río Cuarto, Argentina

Conference: ASE 2016, Singapore, September 3 - 7, 2016.
http://dc.exa.unrc.edu.ar/staff/rdegiovanni/files/ASE2016.pdf

The derivation of correct software requirements specifications is essential to any reliable software development process. With the ever increasing complexity of software, the importance of rigorous methods in supporting the attainment of correct specifications prior to their implementation, also increases. Much research over the last decades has demonstrated the significant advantages that formal, goal-oriented approaches bring to the generation of correct software requirements specification. Goals are prescriptive statements of how the system should behave. They reflect stakeholders' understanding of what the envisioned system is intended to do, and the criteria upon which it would be evaluated. They are commonly used to: aid the elicitation and elaboration of requirements; guide the refinement and organisation of requirements; and support the derivation of software operations. However, for such tasks to be successfully achieved, the goals themselves must be correct, which is not often the case. Goals are typically too ideal to start with (wavering off the exceptional conditions that may arise within its environment once implemented), partial and imprecise. Ensuring their correctness within the development cycle is of utmost importance.

One of the challenges in specifying correct goals is ensuring their consistency. Inconsistency occurs when two or more goals cannot be satisfied simultaneously, owing to their contradictory nature, non-conformance to standards, or because of restrictions imposed within certain domains, amongst other reasons. They are typically a result of overlapping and conflicting expressions. Detecting and resolving inconsistencies in goals (a process called *inconsistency management*) early on not only helps in avoiding costly software repairs but also supports systematic requirements' elicitation and verification activities. Several approaches have been proposed in the literature for managing inconsistency in goals. Much work has been done on the qualitative end, where the general focus has been on identifying contradictory low-level requirements and computing the degree to which goals are satisficed or denied by them. A weaker notion of conflict (called *divergence*) in goals expressed in Linear Temporal Logic (LTL) has been previously addressed. This latter type of inconsistency is concerned with those goals which are not contradictory (can be simultaneously satisfied), but become inconsistent when certain conditions hold. Consider for instance the following goals from the mine pump controller example: *"the pump shall be on when the water level is above the high threshold"*, and *"the pump shall be off when methane is detected in the mine"*. These goals are not logically inconsistent as they are satisfiable in cases where the water level never reaches a high level or methane is not detected in the mine. They become logically inconsistent only in the case when the water level is high and methane is present at the same time. Situations like the latter can be captured formally as assertions called *boundary conditions*, i.e., declarative formulas that characterise those particular circumstances that lead to inconsistency. Existing model synthesis approaches would not detect this type of inconsistency since there exists at least one model that satisfies such goals, in which the boundary condition never holds. So far, very limited work has been done on automatically finding boundary conditions for goal expressions.

In this oral communication, we present a novel approach to automatically compute boundary conditions for conflicting goals expressed in LTL, using a satisfiability procedure based on tableaux. A tableau for an LTL formula is essentially a finite graph representation of all its satisfying models; it is built by first decomposing the formula whose satisfiability is being analysed, according to decomposition rules that produce, for temporal operators, constraints on the current state and future states, for their satisfaction. The resulting graph explores the possible ways of making the initial formula satisfiable, and in this process, contradictory portions are identified. The second phase of the tableau method removes contradictory portions, as well as parts of the graph that cannot satisfy eventualities, leaving a subgraph, the tableau, that captures *all* models of the formula (when it becomes empty, the formula is unsatisfiable). Intuitively, the tableau indirectly captures "conflicting situations", since any condition not included in the tableau necessarily prevents the formula from being satisfiable. Our approach consists of computing the tableau from a set G of goals, and then exploring it to identify conditions that would "escape" the tableau, thus violating the goals, to produce boundary conditions. Our approach is general, in the sense that it can automatically detect conflicts in goals expressed as any LTL formula. In particular, our technique can automatically produce boundary conditions that are more general than those obtainable through existing previous pattern-based approaches, and can also generate boundary conditions for goals that are not captured by these patterns.