Assignment 2

Introduction to my research project

I am working on a problem in synthesis of controllers for reactive systems. Reactive systems are systems which react to a changing environment. The task of synthesis to design an algorithm which outputs a correct-by-construction program from a given specification. It is important to note that there is a tradeoff between the expressiveness of the logic for specifying the specification and the complexity of the algorithm. There are logics which are expressive enough to encode the halting problem for programs which makes this task undecidable in general. An interesting task is to find logics which are reasonably expressive and yet the synthesis problem tractable.

There is a different approach to this problem arising from control engineering known as "supervisory control theory". In this approach, there is a state-machine with some marked states and the goal is to synthesize a supervisor which restricts the controllable events in such a way that always there is a path to a marked state. This problem is well studied from control engineering perspective. In my PhD project, we are trying to combine techniques from formal methods and control engineering to find better algorithms for synthesis.

1. Automated Software Testing

Software testing is important to ensure reliability. As the name suggests, automated testing is testing done with the help of automation. It is usually done along with manual testing. Automation is done to generate tests automatically from given specifications, to run tests faster, to reduce cost and to increase accuracy. There are many frameworks for testing, e.g.: Data-driven testing, Modularity-driven testing, Model-based testing, etc. Also, there are many modes of testing ranging from Whitebox to Blackbox depending on the access to the source code.

Unit testing is the foundational level of testing, each function can be specified, and tests can be generated by partitions of the input space according to expected behavior. Beyond unit testing, there is service testing where different services are tested for a given system. On topmost level, there is UI testing. In UI testing, overall performance of the system is evaluated.

Testing is a more practical and feasible way of increasing trust in the software compared to verification techniques arising from formal methods. Interestingly, formal method tools such as SAT solvers almost always rely on testing for their credibility. It becomes even more important when an algorithm is not complete but faster on benchmarks than known complete algorithms for the same problem.

In my project, focus is mainly on mathematical guarantees and hence testing does not directly play any role. However, once we have algorithms with mathematical guarantees, it is valuable to implement it and then testing becomes crucial. Testing increases trust in the correctness of implementation. Automated testing can speed up this process.

2. Sustainability

Sustainability is an important aspect of life due to many reasons one of them is limit on the availability of resources. Software engineering is based on two main computational resources: memory space and run-time. Memory space requires hardware chips and run-time corresponds to energy required. Chips manufacturing requires a lot of silicon and energy is needed in almost everything. Efficient software can reduce the energy requirements and the hardware requirements to some degree. Computation is energy costly cause memory chips cause heating and then cooling is required.

Apart from the core computational resources required in running software, there is a component of maintenance of software products. Maintaining software usually requires storage and consistent development as per evolving dependencies and evolving needs. This often requires human resource along with the hardware and energy. One of the ways it can be made more sustainable is by using modular development where possible. Modularity provides access to different components of the software and makes it easier to develop. Efficiency in the software helps in maintaining as well.

Reactive synthesis provides techniques to synthesize correct-by-construction program for reactive systems when specifications can be expressed in fixed point logics such as linear temporal logic. Such constructions usually provide a minimal program in terms of memory space being used, that can lead to construction of efficient programs. Another potential assistance formal method can provide is via the planning problem. Software maintenance can be modelled as a planning problem and then formal method techniques such as game solving, and verification can provide a solution to such problems by designing a proper plan. However, this remains a separate research problem to utilize formal methods for software maintenance.

3. Security and Privacy

Security and privacy are increasingly becoming of concern of society. Security is usually associated with communication protocols, encryption mechanisms and information flow control. The goal there is to ensure that the access to information is limited to appropriate agents. At times there can be software which leak data in an unexpected way. Formal method techniques such as automated theorem proving has been used to verify cryptographic protocols.

Reactive synthesis does not have any direct links with software security, however the techniques used in automated theorem proving and reactive synthesis have a lot in common. A future possibility in this area is the synthesis of cryptographic protocols from specifications. This rather seems impractical in current scenario due to computational cost of the search methods used in automated synthesis.

Privacy is become more of an interest due to learning algorithms coming in the picture. These algorithms might store private data and can make it publicly accessible either directly or indirectly. Apart from accessibility these algorithms can potentially learn about an individual's behavior and choices which could be utilized against the individual in an adversarial way.

The good part is that the techniques involved in information security can be applied to ensure privacy in certain setups. The links with formal methods are hence very similar to that of the links between formal methods and security.

Future trends in Reactive Synthesis

Reactive synthesis is now shifting more towards automated planning. It is useful in conditions where environment can behave in uncertain ways, and we still need to guarantee safety conditions. The problem in reactive synthesis is that it does not scale due to high complexity even for lesser expressive specification logics, i. e. the search space is very large. Some heuristics guided search techniques become useful in this instance. There is a tool called STRIX which is based on heuristic methods, and it performs very well on the benchmarks as well as in practice.

Another potential direction is application of leaning methods in synthesizing the model and then verifying to guarantee correctness. Verification is usually computationally cheaper than synthesis and hence heuristically or learning based model generation and validation by formal techniques could lead to more efficient tools in this direction.

SAT/SMT- solvers have influenced verification tools in a revolutionary way. Verification problems can be encoded as SAT instances and although SAT is NP-complete in theory, SAT solvers scale quite well in practice. The equivalent problem for synthesis is Quantified Boolean Formula (QBF)-SAT where there are quantifications over propositions. QBF-solving is an emerging area, however there are not any QBF-solvers which scale well till date, but techniques as CDCL have been extended to work for QBF.

Game solving is a well-known method in reactive synthesis. One of the games used in reactive synthesis is the Parity games, solving these games are known to be in NP and in co-NP but not known to be in P. That provides a bottleneck in scaling. This is conjectured to be in P. This is an open and flourishing area with many researchers trying to solve this algorithmic problem. In future, if this is solved, it will make reactive synthesis more applicable.

Supervisory control theory provides a synthesis method which can be composed as the specifications are composed. This is not yet known for reactive synthesis; my project is in the direction of building links between the two areas and enabling compositional method for reactive synthesis. If compositional method works, then this can potentially make reactive synthesis scaling.

Another model for reactive synthesis is "behavior trees", these are modular and composable but does not have other structural properties which state machines have. A future direction might be to investigate the appropriate models for reactive synthesis depending on the need.

There is an ongoing project on the synthesis of multi-agent system in distributed computational models. The bottlenecks there come from the high computational requirements for concurrency.

Quantum computing is expected to boost the available computational power. It is still open to investigate if the computational problems arising out of reactive synthesis are efficiently solvable with a quantum or even classical randomized machines. It is expected that in future this will be investigated.

Apart from theoretical issues, there are plenty of application arenas in which reactive synthesis is to be applied and evaluated. Game solving techniques provide a natural link with utility-optimization problem in economics. It remains to see if reactive synthesis can contribute to economics with respect to synthesis of rational agents which can make optimal decisions.