AMETIST Project: Scientific Summary for Year 1

Oded Maler

VERIMAG

Grenoble, France

Brussels, 19 June 2003

Plan

The AMETIST vision

Modeling

Algorithmics (Analysis)

Tools

Relation to Case-Studies

Conclusions

The AMETIST Vision: the State of the Art

Observation: there are many similar problems in diverse domains.

In each domain they are solved using different approaches and mathematical tools.

Building a **unifying framework** whose applicability crosses the boundaries of the application domains is something worth trying (imagine a different arithmetics used for summation of money, apples or liquid quantities).

It is evident that such an initiative will be met with skepticism by practitioners and theoreticians in all those domains.

But we have patience.

The Class of Problems

The class of problems we are concerned with are often called problems of **scheduling**, **planning** etc.

The essential features of all these problems are:

1) You want to do several things (tasks) to achieve your goals. Each of these tasks **takes some time** to be executed.

2) These tasks have complex mutual **inter-dependencies**, most notably precedence (task A can be done only after task B) and triggering (A has to be done if the result of B satisfies some condition).

3) The execution of the tasks requires some **resources** whose quantity is **bounded**. Hence there are sometimes **conflicts** that have to be resolved by a **scheduling policy**.

The Application Domains

Scheduling production lines in factories to optimize costs and delays.

Scheduling of computer programs in real-time systems to meet deadline constraints.

Scheduling of micro instructions inside a processor with a bounded number of registers and processing units.

Scheduling of trains (or airplanes) over limited quantities of railway tracks and crossroads.

Mission planning for autonomous robots on spacecrafts.

...

The Different Features

Each problem is unhappy in its own way. Some characteristic features:

Time-scales (micro seconds vs. days and weeks).

Level of information known a-priori, uncertainty, online vs. offline.

Nature of the processes (material vs. information), preemption, etc.

Optimization vs. meeting constraints.

Limitations on the resources that the scheduler may use.

...

Sketch of some Approaches

The **Operation Research** approach: reduce the problem into one or more well-known optimization (mathematical programming) problems.

Real-time Operating Systems approach: use some priority-based policy implemented in the operating system. Becomes more complicated when tasks also occupy some other resources.

Many other ad-hoc approaches invented and re-invented in each application domain.

We want to do something different, clean but useful (hopefully not an oxymoron).

The Essence of the AMETIST Approach

Components of the system in question are modeled as **dynamical systems** with state-space and well defined dynamics.

All that can happen in the system is expressed in terms of **behaviors** that can be generated by the above model. This is the **semantics** of the problem.

Verification, optimization, synthesis and other design activities explore and modify system structure so that the resulting behaviors are correct, optimal, etc.

Do not let the limitation of currently known computational solutions to influence modeling too much. After semantics is understood, then simplifications and specialization due to computational considerations can intervene.

The Joy of Timed Automata

Such an approach underlies the successful domain of **formal verification** and our mission is to extend it to the scheduling and other time-related problems.

The mathematical carrier of this methodology is the **timed automaton** model, a model that combines discrete events and passage of time.

In addition to discrete state variables, state information is encoded into **clock variables** that measure the time since certain events.

Already toward the end of the previous project VHS, we saw that this model is adequate for expressing such scheduling problems.

Some tools for analyzing timed automata already exist.

Proof of Concept

In order to show the viability of the approach we promised to work on:

Modeling: show the expressive power of timed automata for modeling the type of problems we are interested in.

Algorithmics: develop efficient algorithms to solve such problems (analysis of TA is considered very hard).

Tools: implement these algorithms in existing and new tools; improve existing ones.

Case-studies: reality check with some applications.

Modeling I

While working on the SIDMAR steel plant case-study of the VHS project it was realized that many scheduling problems can be modeled as problem of **finding shortest-paths in timed automata.**

In the current project the modeling framework was extended to treat more complex scheduling situation, in particular:

Scheduling with **preemption** using stopwatch automata (timed automata where some clocks can be stopped and resumed).

Scheduling of task graphs (scheduling parallel programs with precedence constraints on parallel machines) with release times and deadlines.

Modeling II

Scheduling with **temporal uncertainty** concerning task durations (both set-theoretical and probabilistic uncertainty).

Using cost functions richer than the total elapsed time (cost associated with transitions, different staying costs at different states).

Extension of job-shop scheduling to treat different machines with different throughputs that can do the same tasks. First steps toward modeling value chains and combination of temporal and resource constraints.

Modeling: Work in Progress

Scheduling of acyclic programs that combine **parallelism** (several machines and precedence constraints) and **choice** (if-then-else constructs with conditions that are computed after the termination of some tasks). Of extreme importance for embedded systems.

Efficient modeling of realistic probability distributions of task durations.

Defining the appropriate performance measures for scheduling of cyclic tasks (the quantitative analogue of moving from automata to ω-automata).

Rigorous translation of subsets of languages used by the AI community for defining and solving planning problems.

Systematic ways to express directives for guiding the search in timed automata, without having to hack with the semantics.

Modeling: Assessment

As promised richer planning and scheduling situations are progressively modeled.

The approach so far: start with the **simplest problem** (e.g. deterministic job-shop without preemption) and add features.

Toward the end of the project a systematic classification of these models will be done, including the computational implications of each feature.

Analysis I

The standard verification of TA is hard (state and clock explosion) and we try to tackle this situations by improvements in data-structures and algorithms:

Better (state-specific) bounds abstraction for the clocks in a TA.

Loop acceleration techniques to reduce significantly the number of verification steps.

A symmetry reduction technique for more efficient verification of systems having several identical components has been lifted to treat zones by permuting the dimensions.

Storage policies: a procedure for selective storage of reachable symbolic states has been developed. It reduces memory consumption significantly.

Analysis II

A restricted form of controller synthesis using verification. Synthesis under partial observation.

Partial-order reductions: new techniques have been developed for avoiding the explosion associated with the interleaving of independent actions.

Non-lazy runs: it turned out that for certain optimization and synthesis problems, only a **finite subset** of the runs of the TA needs to be explored. Consequently, most of the heavy machinery of TA (zones) need not be used. This leads to a huge improvement in time and memory.

Bounded model-checking for timed automata via a SAT solver for difference logic (see below).

Analysis: Work in Progress

For problems of adversarial scheduling one needs to use some kind of strategy synthesis algorithms.

The problem has been solved in the past for TA using backward value iteration (dynamic programming).

Since the transition graph is already exponential in the number of components, this approach is limited.

For the deterministic setting one can use forward “best first” algorithms that explore only a small fraction of the search tree.

To do this for adversarial problems one needs to adapt game (AND-OR) tree searching algorithms to TA.

Analysis: Fundamental Studies I

"Dynamic" and "static" approaches to verification:

The former is based on searching over the TA runs in an order that corresponds to the execution (forward or backward).

The latter is what is called "bounded model checking": the existence of a run of a given length is expressed as a fomrula in propositional logic whose satisfiability is checked using some general-purpose SAT solver.

Same classification applies to quantitative problems such as optimal scheduling: the dynamic approach corresponds to searching paths in the timed automaton. The static approach is the one used traditionally in the domain: formulate the optimal schedule as a solution to a constrained optimization problem.

Analysis: Fundamental Studies II

To apply the static approach to general verification of TA, we have identified the corresponding logic: DL (difference logic), propositional logic augmented with difference constraints of the form $x - y < c$.

We have shown how to express bounded reachability problems for TA as formulae in DL.

We have designed and implemented a prototype **SAT solver for DL.** Some work is still needed to make it competitive.

Analysis: Fundamental Studies III

In fact, after having formulated the problem as a DL formula, there are still two basic design choices concerning the way the mixed constraint satisfaction/optimization can be solved.

The problem is mixed because it involves Boolean and numerical constraints.

The approach dominant in OR gives “priority” to the numerical constraints (which, without the logic can be solved using good old linear programming). Hence Boolean variables are considered as integer and using MILP techniques are “relaxed” to real, B&B, etc.

The approach for problems dominated by logic is to treat Boolean variables as they are and use constraint propagation techniques.

In inter-cultural collaboration we compare these approaches.

Analysis: Assessment

A lot of serious work ranging from applying standard techniques used in untimed verification to the TA setting, up to development of new algorithms for new problems.

Attempt to extend the bounded model-checking approach to TA via SAT.

Comparison with MILP and other optimization methods (should get a CPLEX/Solver license from ILOG).

Tools

Most of the above has been implemented and sometimes integrated into existing tools.

In addition:

Translation from the PDDL language (AI planning) to Uppaal.

A prototype tool for posing scheduling problems as TA and solving using non-lazy runs;

A prototype tool for automatic abstraction of TA models of large acyclic circuits.

A translator from TA to DL.

Integration of schedulability analysis into a Java compiler.

Case-Studies and their Relevance

Terma: computer scheduling problem, can be modeled using our approach. First version solved, waiting for new features.

Bosch: another real-time computing application. Involves a lot of hybrid aspects that should be abstracted somehow. Too early to know.

Cybernetix: no timing but an interesting synthesis problem.

Axxom: an ideal application of our methodology. First instance has been solved successfully using the scheduling algorithms just developed.

Conclusion

The project pushes forward the frontiers of our understanding of timed systems.

Still a significant effort is needed in the second year to enlarge the scope of applicability of the TA approach.

With some convincing applications dissemination will become easier. The AI planning community seems to be ready to adopt some of the TA methodology.