An Overview on Code Synthesis and Runtime Verification

A Broad Vision of our Goals and Achievements

- Gergely Pintér

Budapest University of Technology and Economics
Department of Measurement and Information Systems
Key Achievements
Presented in my Thesis

- A *formal semantics* for UML state machines
- A method for the *automatic implementation* of UML state machines
- Two *verification methods* for the runtime evaluation of state-based behavior
Key Achievements Present in my Thesis

- A formal semantics for UML state machines
- A method for the automatic implementation of UML state machines
- Two verification methods for the runtime evaluation of state-based behavior

Semantics
- What does a complex statechart actually mean?
Key Achievements
Presented in my Thesis

- A *formal semantics* for UML state machines
- A method for the *automatic implementation* of UML state machines
- Two verification methods for the runtime evaluation of state-based behavior

**Code Synthesis**
- How to *implement* the control structure described by a statechart?
- Demonstrated by code generation for a *μC based device*. 
Key Achievements

Presented in my Thesis

- A formal semantics for UML state machines
- A method for the automatic implementation of UML state machines
- Two verification methods for the runtime evaluation of state-based behavior

Runtime Verification

- How to check that the application actually behaves according to its specification?
- How to add extra timing related requirements to a statechart and check them during runtime?
Key Achievements
Presented in my Thesis

- A *formal semantics* for UML state machines
- A method for the *automatic implementation* of UML state machines
- Two *verification methods* for the runtime evaluation of state-based behavior
Key Achievements
Presented in my Thesis

- A formal semantics for UML state machines
- A method for the automatic implementation of UML state machines
- Two verification methods for the runtime evaluation of state-based behavior

Research Focus:
- Unambiguous specification,…
- …automatic implementation and…
- …runtime verification of…
  complex control structures
Key Achievements
Presented in my Thesis

- A formal semantics for UML state machines
- A method for the automatic implementation of UML state machines
- Two verification methods for the runtime evaluation of state-based behavior

Research Focus:
- Unambiguous specification,
- ...automatic implementation and...
- ...runtime verification of... complex control structures

When talking about “Control Structures”...
- We are talking about control concept of programming and modeling languages (e.g., do-while loops, if-else branches, functions, even processes or threads)...
- ...and not process control concepts like PID controllers, ZOHs, etc.
- ...i.e., “how C/C++/Java/etc. statements are organized into a program”
Let’s Focus a Bit on Automatic Code Synthesis...

- Originally aimed benefits:
  - Substitution of a *labor-intensive error-prone task* with a proven correct *automatic tool*
    - *Reduction of development costs*
      - Human effort, time, maintenance cost
    - *Increase in code quality*
      - Complex, hard to understand parts generated automatically
      - Human focus on key tasks (i.e., atomic activities), boring labor-intensive maintenance of the control structure carried out by a tool
Let’s Focus a Bit on Automatic Code Synthesis

- Originally aimed benefits:
  - Substitution of a labor-intensive error-prone task with a proven correct automatic tool
  - Reduction of development costs
  - Human effort, time, maintenance cost
  - Increase in code quality
  - Complex, hard to understand parts generated automatically
  - Human focus on key tasks (i.e., atomic activities), boring labor-intensive maintenance of the control structure carried out by a tool

These goals are important indeed, but there is a much broader horizon ahead of us:

- We are using ever more computing cores in devices,…
- …these cores may be dedicated to various goals and,…
- …are frequently idle due to “badly written programs”,…
- …while consuming energy
Let’s Focus a Bit on Automatic Code Synthesis…

- Originally aimed benefits:
  - Substitution of a labor-intensive boring task with a proven correct automatic tool
  - Reduction of development costs
  - Human effort, time, maintenance cost
  - Increase in code quality
  - Complex, hard to understand parts generated automatically
  - Human focus on key tasks (i.e., atomic activities), boring labor-intensive maintenance of the control structure carried out by a tool

These goals are important indeed, but there is a much broader horizon ahead of us:
- We are using ever more computing cores in devices,…
- …these cores may be dedicated to various goals and…
- …are frequently idle due to “badly written programs”…
- …while consuming energy
Let's Focus a Bit on Automatic Code Synthesis…

Originally aimed benefits:
- Substitution of a labor-intensive boring task with a proven correct automatic tool
- Reduction of development costs
- Human effort, time, maintenance cost
- Increase in code quality
- Complex, hard to understand parts generated automatically
- Human focus on key tasks (i.e., atomic activities), boring labor-intensive maintenance of the control structure carried out by a tool

But why should a programmer understand the inner details of a multi-core CPU? (That may have not even been manufactured yet…)

These goals:
- We are using ever more cores in devices,…
- …these cores may be dedicated,…
- …are frequently idle due to “badly written programs”,…
- …while consuming energy

Citation form an actual CPU expert
Let’s Focus a Bit on Automatic Code Synthesis…

- Originally aimed benefits:
  - Substitution of a labor-intensive boring task with a proven correct automatic tool
  - Reduction of development costs
  - Human effort, time, maintenance cost
  - Increase in code quality
  - Complex, hard to understand parts generated automatically
  - Human focus on key tasks (i.e., atomic activities), boring labor-intensive maintenance of the control structure carried out by a tool

Idea

- Extend visual control models with information about the most beneficial platform and do the mapping automatically by the control code synthesis tool

These...
Presentation Structure

Wide Context and Future Research Goals

Achievements Until Now

Demonstration
Presentation Structure

- Wide Context and Future Research Goals
- Achievements Until Now
- Demonstration
Warning: The next part of the presentation is mostly brain storming about future research activities. Do not expect proven, fine-tuned solutions! Our goal here is to give a broad overview on our planned work and discuss our and your ideas concerning the subject.
Control Hierarchy

State-Transition Model

Detailed Activity Model

Process/Thread Model

Methods, Statements

Machine Instructions

Micro-Instructions
Control Hierarchy: Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions
Control Hierarchy: Typical Description Forms

- State-Transition Model
  - UML statecharts
  - Matlab/Stateflow diagrams
  - Harel statecharts, etc.
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions
Control Hierarchy: Typical Description

State-Transition Model
- UML statecharts
- Matlab/Stateflow diagrams
- Harel statecharts, etc.

State 1
exit/

State 2
entry/…

State 2A
entry/…

State 2B
entry/…

Top-level organization of activities

State-Transition Model

Detailed Activity Model

Process/Thread Model

Methods, Statements

Machine Instructions

Micro-Instructions
Control Hierarchy: Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Detailed Activity Model
- UML activity diagrams
- Matlab/Stateflow diagrams
Control Hierarchy: Typical Description

State-Transition Model
Detailed Activity Model
Process/Thread Model
Methods, Statements
Machine Instructions
Micro-Instructions

Detailed Activity Model
- UML activity diagrams
- Matlab/Stateflow diagrams
Control Hierarchy: Typical Description Forms

State-Transition Model
Detailed Activity Model
Process/Thread Model
Methods, Statements
Machine Instructions
Micro-Instructions

Process/Thread Model
- Top-level structure of the source code (approximately)

src/
  signal_processing/
    Makefile
    some_dsp_library.c
  io/
    Makefile
    some_io_library.c
main.c
Control Hierarchy:

Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Assignment of activities to program images thus selection of target architecture (Makefiles are shown for a reason)

Process/Thread Model

- Top-level structure of the source code (approximately)

```
src/
signal_processing/
    Makefile
    some_dsp_library.c
io/
    Makefile
    some_io_library.c
Makefile
main.c
```
Control Hierarchy: Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Methods, Statements
- Source code of methods

```c
#ifndef SOME_IO_LIBRARY
#define SOME_IO_LIBRARY

int some_io_method() {
    for (...) {
        if (...) {
            // ...
        } else {
            // ...
        }
    }
}
```
Control Hierarchy:
Typical Description Forms

- State Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Implementation of activities in a high-level language

Methods, Statements
- Source code of methods

```c
#ifndef SOME_IO_LIBRARY
#define SOME_IO_LIBRARY

int some_io_method() {
    for (...) {
        if (...) {
            // ...
        } else {
            // ...
        }
    }
}
```

Control Hierarchy: Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Machine Instructions
- Machine (assembly) language sources

some_io_method:
   pushq %rbp
   movq %rsp, %rbp
   subq $16, %rsp
   movl %edi, -4(%rbp)
   movl $0, -8(%rbp)
   movl %edi, -4(%rbp)
   cmpl -8(%rbp), %eax
   cmpl -4(%rbp), %eax
   jge .L4
Control Hierarchy: Typical Description Forms

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-InSTRUCTIONS
  - Implementation of machine instructions in the CPU
  - Multiple pipelines, ALUs, caches, etc.
Programmers (architects or lead programmers) had to decide about process-core allocation. *This should not be their decision, because they are not CPU experts.*
Control Hierarchy: A Chain of Jobs

- State-Transition Model
- Detailed Activity Model
- Process/Thread Model
- Methods, Statements
- Machine Instructions
- Micro-Instructions

Process organization is here
Computing resources are here
Control Hierarchy:
A Chain of Jobs

I.e., assignment of activities to program images thus architectures thus HW resources

Process organization is here

Computing resources are here
Effects of Task-Core Assignment
Effects of Task Assignment

We will use this *activity model* as an example (actually a mixed form of the state and activity models for simplicity reasons). Vertical bars are *fork/join symbols*, rounded rectangles are *activities*.

We will use this image for representing a *multi-core CPU*.
Effects of Task Assignment

Top-Level Control Structure
- Some mixed view of the state-transition and the activity models
- Consisting of dominantly
  - …fixed point and
  - …floating point steps

Actual CPU
- Two cores, both cores with
  - …fixed point and
  - …floating point ALUs
Effects of Task-Core Assignment

OK, so I have to organize this activity structure into a program...

(Jon, programmer)
Effects of Task-Core Assignment

Hmmm, a single threaded executable will do it, let’s see some good olde’ coding…

(Jon, programmer)
Effects of Task-Core Assignment
...and let the stuff run!
“I paid 2000 bucks for this computer and my game is still crawling.”

(Johnny, Gamer)
“I paid 2000 bucks for this computer and my game is still crawling.”
*(Johnny, Gamer)*

“Our processor is perfect, your program is *badly written*. One of the cores is entirely idle because the entire program is running in a *single thread* on a *single core*.”
*(Jonathan, CPU Expert)*
“I paid 2000 bucks for this computer and my game is still **crawling**.”  
*(Johnny, Gamer)*

“Our processor is perfect, your program is **badly written**. One of the cores is entirely idle because the entire program is running in a **single thread** on a **single core**.”

*(Jonathan, CPU Expert)*

*(Jon, Programmer)*
Effects of Task-Core Assignment

Maybe if I divide the single process into two threads…
Effects of Task-Core Assignment

Hmmm, this will do it, let’s see some good olde’ coding…
Effects of Task-Core Assignment

Diagram showing the effects of task-core assignment with numbered nodes and arrows indicating the flow of tasks.
Effects of Task-Core Assignment

...and let the stuff run!
Wow, reduced the execution time by two!
“I paid 2000$ for this laptop and it is burning a hole in my pants.”

(Johnny, Gamer)
“I paid 2000$ for this laptop and it is *burning a hole in my pants.*”
*(Johnny, Gamer)*

“Our processor is perfect, your program is *badly written.* A *partially loaded* core is running at full clock frequency, however the *power consumption* can be *reduced* by decreasing core speed.”
*(Jonathan, CPU Expert)*
“I paid 2000$ for this laptop and it is burning a hole in my pants.”

(Johnny, Gamer)

“Our processor is perfect, your program is badly written. A partially loaded core is running at full clock frequency, however the power consumption can be reduced by decreasing core speed.”

(Jonathan, CPU Expert)

(Jon, Programmer)
Effects of Task-Core Assignment

Maybe if during step 2 and 4 I decrease the core speed can reduce heat dissipation...
Effects of Task-Core Assignment
Effects of Task-Core Assignment
Effects of Task-Core Assignment

OK, this will do it, let's run the stuff again.
Effects of Task-Core Assignment

Time/Power Consumption

6  6
Effects of Task-Core Assignment

Wow, reduced the power consumption by two!
“Let me draw your attention to our brand new embedded CPU with…

• One general-purpose fixed-point core and…

• Two high speed RISC floating point cores.

You will be able to re-organize your program such way that it will run even faster with lower power consumption…”

(Jonathan, CPU Expert)
Effects of Task-Core Assignment

“To exploit the benefits of the shining new CPU…”
(Jonathan, CPU Expert)
“...you only have to re-organize your threads this way...”

(Jonathan, CPU Expert)
Effects of Task-Core Assignment
“...and look: spared some more time. Just had to re-organize the implementation of the top-level control structure.”

(Jonathan, CPU Expert)
“...and look: spared some more time. Just had to re-organize the implementation of the top-level control structure.”

(Jonathan, CPU Expert)

OK, that's it. I've never been a CPU expert and never wanted to be one. This task allocation magic should be done by somebody else.
“...and look: spared some more time. Just had to re-write the top-level control structure.”
(Jonathan, CPU Expert)

OK, that’s it. I’ve never been a CPU expert and never wanted to be one. This task allocation magic should be done by somebody else.

“Jon is right: we are actually missing a role here!”
(Mr. Johnson, manager)
“...and look: spared some more time. Just had to re-write the top-level control structure.” (Jonathan, CPU Expert)

Idea:

- **Annotate** high-level control structure with *typical resource consumption characteristics* (e.g., mostly FPU-intensive step, mostly IO-intensive step, etc.) and...
- ...do thread-core allocation *automatically*

In practice: extend our already existing code generation solution with “*multi-core awareness*”.

* Citation from a CPU expert
Presentation Structure

Wide Context and Future Research Goals

Achievements Until Now

Demonstration
Presentation Structure

Wide Context and Future Research Goals

Achievements Until Now

Demonstration
Achievements Until Now...

- Understanding complex control structures
  - Unambiguous formal semantics for UML statecharts
    - Mapped to Kripke transition systems
  - Relations of activities expressed by $PERT$-graphs
    - Support for arbitrary complexity
    - This is the entry point for multi-core awareness!
- Automatic implementation of control structures
  - Automatic code synthesis for ANSI-C and Java
    - Demonstrated even on a *Mitmot* device
  - ...actually schedules the precisely calculated activity
    PERT graphs to a single thread...
Achievements Until Now...

- Runtime verification
  - Reference specification:
    - UML Statecharts
    - Temporal correctness criteria (PLTL)
  - Not even mentioned here but at least as important as code synthesis
Achievements Until Now...

Solving Jon’s Problem...

- Formal Semantics for Statecharts
- Control Code Synthesis (Multi-Core Unaware)

Still working...
Achievements Until Now…

Solving Jon’s Problem...

- Formal Semantics for Statecharts
- Control Code Synthesis (Multi-Core Unaware)

By the way, this was my PhD thesis…

Still working…
Achievements Until Now…

By the way, this was my PhD thesis…

Still working…
Presentation Structure

- Wide Context and Future Research Goals
- Achievements Until Now
- Demonstration
Presentation Structure

Wide Context and Future Research Goals

Achievements Until Now

Demonstration
Choose our simulator plug-in's view
Feed in events and check the behavior...
If satisfied, set-up code generation properties…
...and run our code generator to synthesize the implementation of the control structure.
```java
protected static final GPERTGraph idPERTGraph142 = new GPERTGraph(Arrays.asList(new GPERTNode[]{
    new GPERTNode(idWState87)
}));

// GTransitionConglomerate instances
protected static final GTransitionConglomerate idTransitionConglomerateE150 = new GTransitionConglomerate(
    Arrays.asList(new GState[] { idWState87 }),
    Arrays.asList(new java.lang.Class[] { TimerSignal.class }));

// GTransitionConglomerate instances
protected static final GTransitionConglomerate idTransitionConglomerateE125 = new GTransitionConglomerate(
    Arrays.asList(new GConstraint[] { idWPossiblePreviousConfigurationConstraint101 }),
    Arrays.asList(new GStateHierarchyNode[]),
    Arrays.asList(new GStateHierarchyNode[]),
    Arrays.asList(new GStateHierarchyNode[]));

protected static final GTransitionConglomerate idTransitionConglomerateC151 = new GTransitionConglomerate(
    idPERTGraph142,
    Arrays.asList(new GStateHierarchyNode[]),
    Arrays.asList(new GStateHierarchyNode[]));
```

// GTransitionConglomerate instances
protected static final GTransitionConglomerate idTransitionConglomerateC150 = new GTransitionConglomerate(
    Arrays.asList(new GState[] { idWState87 }),
    Arrays.asList(new java.lang.Class[] { TimerSignal.class }));

Properties are not available.
package hu.bme.mit.pinterg.puml_rsa.examples.traffic_light.ui;

import hu.bme.mit.pinterg.puml_rsa.generator.base_classes.BehaviorContext;

class Main {

    /**
     * @param args
     */
    public static void main(String[] args) {
        (new Main()).doIt(args);
    }

    public void doIt(String[] args) {
        TrafficLight trafficLight = TrafficLight.getInstance();
        BehaviorContext behaviorContext = new BehaviorContext(null, trafficLight, null);
        initialize(trafficLight, behaviorContext);
        step(trafficLight, behaviorContext, new LampSwitchSignal());
        step(trafficLight, behaviorContext, new TimerSignal());
        step(trafficLight, behaviorContext, new TimerSignal());
        step(trafficLight, behaviorContext, new CarArrivedSignal());
    }
}
```java
private void initialize(TrafficLight aTrafficLight, BehaviorContext aBehaviorContext)
{
    System.out.println("-- Initialization --");
aTrafficLight.initializationStep(aBehaviorContext);
    System.out.println(" Active states:");
    for (GState gState : aBehaviorContext.configuration)
        System.out.println(" " + gState.representedName);
}

private void step(TrafficLight aTrafficLight, BehaviorContext aBehaviorContext, GSigns gSigns)
{
    System.out.println("-- [Dispatching signal: " + gSignal.getClass() + " --...");
aTrafficLight.triggerProcessingStep(aBehaviorContext, gSigns);
    System.out.println(" Active states:");
    for (GState gState : aBehaviorContext.configuration)
        System.out.println(" " + gState.representedName);
    System.out.println("-- -- -- -- -- -- -- --\n
```
```
package hu.bme.mit.pinterg.puml_rsa.examples.traffic_light.ui;

import hu.bme.mit.pinterg.puml_rsa.generated.base_classes.BehaviorContext;

public class Main {
    public static void main(String[] args) {
        (new Main()).doIt(args);
    }

    public void doIt(String[] args) {
        TrafficLight trafficLight = TrafficLight.getInstance();
        BehaviorContext behaviorContext = new BehaviorContext(null, trafficLight, null);
        initialize(trafficLight, behaviorContext);
        step(trafficLight, behaviorContext, new LampSwitchSignal());
        step(trafficLight, behaviorContext, new TimerSignal());
        step(trafficLight, behaviorContext, new TimerSignal());
        step(trafficLight, behaviorContext, new CarArrivedSignal());
    }
}
The demonstration was for Java but actually we also have ANSI-C ports.
Presentation Structure

- Wide Context and Future Research Goals
- Achievements Until Now
- Demonstration
“I need more processing power with less energy consumption for the same price.”
(Johnny, Gamer)

“Our perfect CPUs deserve better written programs!”
(Jonathan, CPU Expert)

“Let the CPU-specific control structure organization calculated by somebody else.”
(Jon, Programmer)