Chapter 1, part 3: Embedded Computing

High Performance Embedded Computing
Wayne Wolf
Models of computation

- Models of computation affect programming style.
- No one model of computation is good for all algorithms.
- Large systems may require different models of computation for different parts.
  - Models must communicate compatibly.
Control flow graph

- Commonly used to model program structure.

\[
x = a
\]

\[
i = 0?\]

\[
x = a - b
\]

\[
y = c + d
\]
CDFG properties

- Finite state model.
- Single thread of control.
- Can handle subroutines.
Finite state machine

- State transition graph and table are equivalent:

<table>
<thead>
<tr>
<th></th>
<th>s1</th>
<th>s2</th>
<th>s3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>s1</td>
<td>s2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>s1</td>
<td>s1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>s2</td>
<td>s2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>s2</td>
<td>s3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>s3</td>
<td>s3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>s3</td>
<td>s1</td>
<td>1</td>
</tr>
</tbody>
</table>
Classical automata

- Moore-automata:
  \[ Y = \lambda(Z); \quad Z^+ = \delta(X, Z) \]
- Mealy-automata
  \[ Y = \lambda(X, Z); \quad Z^+ = \delta(X, Z) \]

Next state \( Z^+ \) computed by function \( \delta \)
Output computed by function \( \lambda \)

Moore- + Mealy automata = finite state machines (FSMs)
Nondeterministic FSM

- Several transitions out of a state for a given input.
  - Equivalent to executing all alternatives in parallel.
- Can allow ε moves---goes to next state without input.
Deterministic FSM from nondeterministic FSM

- Add states for the various combinations of nondeterminism.

```
s1
  ↓
  c

s4

s1 → s2
a

s3

s4

s12

s1
  ↓
a

s1
  ↓
c

s4

s12

s1
  ↓
c

s4

s3

b

non-deterministic

deterministic
```
Statecharts

- Provided composite states:
  - OR states;
  - AND states.
- Composite states reduce the size of the state transition graph.
- Use combination of state with “diagram” or “chart”
Introducing hierarchy

FSM will be in exactly one of the substates of S if S is active (either in A or in B or ..)
Definitions

- Current states of FSMs are also called **active** states.
- States which are not composed of other states are called **basic states**.
- States containing other states are called **super-states**.
- For each basic state $s$, the super-states containing $s$ are called **ancestor states**.
- Super-states $S$ are called **OR-super-states**, if exactly one of the sub-states of $S$ is active whenever $S$ is active.
Definitions

- **superstate**
- **substates**
- **ancestor state of A, B, C, D & E**

Diagram:
- The diagram shows a transition graph with states labeled A, B, C, D, and E. State A is the ancestor state of A, B, C, D, and E.
- There are transitions labeled g, h, i, j, m, and f between the states.
- Node S represents the superstate.
Default state mechanism

Try to hide internal structure from outside world!

Default state

Filled circle indicates sub-state entered whenever super-state is entered.

Not a state by itself!
For input m, S enters the state it was in before S was left (can be A, B, C, D, or E). If S is entered for the very first time, the default mechanism applies.

History and default mechanisms can be used hierarchically.
Combining history and default state mechanism

same meaning
Example: Answering Machine
Concurrent ways of describing concurrency are required. **AND-super-states:** FSM is in all (immediate) sub-states of a super-state; Example:

![State diagram](image)

- **Lwait**:
  - Transition: Ring → Lproc
  - Transition: Hangup (caller) → Lwait

- **Lproc**:
  - Transition: Ring → Lproc
  - Transition: Hangup (caller) → Lwait

- **Kwait**:
  - Transition: Key pressed → Kproc

- **Kproc**:
  - Transition: Done → Kproc

**Key Events**:
- Key on
- Key off
- Off
Line-monitoring and key-monitoring are entered and left, when service switch is operated.
Types of states

In StateCharts, states are either

- basic states, or
- AND-super-states, or
- OR-super-states.
Timers

Since time needs to be modeled in embedded systems, timers need to be modeled. In StateCharts, special edges can be used for timeouts.

If event a does not happen while the system is in the left state for 20 ms, a timeout will take place.
Using timers in an answering machine

Lproc

4 s

timeout

play text

beep

lift off

talk

return (callee)

dead

8 s

record

timeout

silent

beep
Petri net

- Parallel model of computation.
Firing rule

- A transition is enabled if each place at its inputs have at least one token.
  - A transition doesn’t have to fire right away.
- Firing a transition removes tokens from inputs and adds a token to each output place.
- In general, may require multiple tokens to enable.
Properties of Petri nets

- Turing complete.
- Arbitrary number of tokens.
  - Nondeterministic behavior.
  - Naturally model parallelism.
Task graph

- Used to model multi-rate systems.
Task graph properties

- Not a Turning machine.
  - No branching behavior.
  - May be extended to provide conditionals.
- Possible models of execution time:
  - Constant.
  - Min-max bounds.
  - Statistical.
- Can model late arrivals, early departures by adding dummy processes.
Data flow graph

- Partially-ordered computations:

+ -, *
+, -, *
-, +, *
Data flow graph properties

- Finite state model.
- Basic data flow graph is acyclic.
- Scheduling provides a total ordering of operations.