System-Level Design
2.01.334, Summer 2024

Exercise 2
Models of Computation

Assigned: April 15, 2024
Due: April 22+29, 2024

General Instructions

1. Please submit your solutions via mail to your advisors (mail addresses can be found at the end of this document). Submissions should include a single typewritten PDF file with the writeup and a single Zip or Tar archive for any supplementary files (e.g. source files, which has to be compilable by simply running `make` and should include a README file with instructions for running each model).

2. You may discuss the problems with your classmates but make sure to submit your own independent and individual solutions. You are allowed to work in groups with up to three members. In this case, please put the full names of all three group members on the cover page of you submitted solutions.

3. Some questions might not have a clearly correct or wrong answer. In general, try to write down your arguments and reasoning how you have arrived at your solutions.

Task 1: Non-determinism

a) What is non-determinism?

b) How might non-determinism arise? (give one example not discussed in class)

c) What are the advantages and disadvantages of having non-determinism in a language or model, i.e. in what circumstances might it be positive/desired or negative/undesired?

Task 2: Synchronous Dataflow (SDF) Synthesis

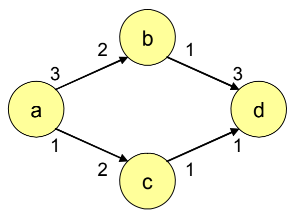


Figure 1: SDF graph

For the SDF graph in Figure :

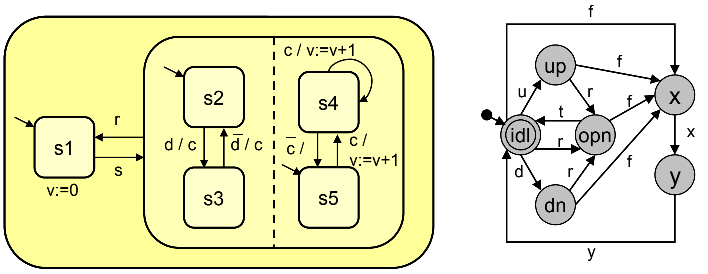
a) Show that the graph is consistent and that it has a valid schedule.

b) List all possible minimal periodic static schedules.

c) Find the periodic schedule with the lowest token buffer usage. What is the minimum buffer usage?

Task 3: State Machines

In class, we have discussed the concepts of extended state machines (FSMs with data), hierarchy (OR state) and concurrency (AND state) for managing complexities in FSMD and HCFSM models.



(a) HCFSMD example

(b) HCFSM example

Figure 2: Finite State Machines

a) Convert the HCFSM(D) in Figure 2(a) into an equivalent FSMD. Note that the concurrent (AND) composition of states is communicating through signal c, and that the HCFSM has Mealy semantics. Transitions for unspecified input conditions default to remaining in the same state, where unspecified outputs default to absence of producing any event. Absence is otherwise indicated as negated conditions on signals. What functionality does the state machine actually perform?

b) Try to reduce the complexity of the FSM in Figure 2(b) by converting it into a HCFSM using hierarchy (OR states) and/or concurrency (AND states) wherever possible. What can you generally say about the complexity (number of states and transitions) of an FSM as a function of the size of an equivalent HCFSM?

Task 4: Models of Computation and Languages

In class, we learned about the different Models of Computation (MoCs): KPN, SDF, FSM(D), HCFSM, PSM.

a) What is the relationship between MoCs and languages?

b) Can SpecC support all these MoCs? If so, briefly sketch for each MoC that you think can be supported how you would represent a corresponding model in SpecC.

c) Is a SpecC specification model deterministic? If yes, why? If not, list possible sources of non-determinism and how they can be avoided (i.e. how a SpecC model can be made deterministic?).

Task 5: Discrete-Event Semantics

For each of the following code examples, what is the value of `myB` printed at the end of execution and at what simulated time does the program terminate? You are free to run the code on top of the SpecC simulator and observe the program output, but you need to provide an explanation and reasoning of why the program is behaving as it is (e.g. sequence of events happening during simulation):

```
behavior A(int myB) {
  void main(void) {
    myB = 10;
  }
};


behavior B(int myB) {
  void main(void) {
    myB = 42;
  }
};


behavior Main(void) {
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

Listing 1: **a)**

```
behavior A(int myB) {
  void main(void) {
    waitfor 42;
    myB = 10;
  }
};

behavior B(int myB) {
  void main(void) {
    myB = 42;
  }
};


behavior Main(void) {
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

Listing 2: **b)**

```
behavior A(int myB) {
  void main(void) {
    myB = 10;
    waitfor 42;
  }
};

behavior B(int myB) {
  void main(void) {
    waitfor 10;
    myB = 42;
  }
};

behavior Main(void) {
  int myB;

  A a(myB);
  B b(myB);

  int main(void) {
    par { a; b; }
    printf("%d", myB);
    return 0;
  }
};
```

Listing 3: **c)**

```
behavior A(int myA, event e) {
  void main(void) {
    myA = 10;
    notify e;
    myA = 11;
    notify e;
    waitfor 10;
  }
};

behavior B(int myA, int myB, event e) {
  void main(void) {
    wait e;
    myB = myA;
  }
};

behavior Main(void) {
  int myA;
  int myB;
  event e;

  A a(myA, e);
  B b(myA, myB, e);

  int main(void) {
    par { a; b; }
    printf ("%d", myB);
    return 0;
  }
};
```

Listing 4: **d)**

```
behavior A(int myA, event e) {
  void main(void) {
    myA = 10;
    notify e;
    waitfor 10;
    myA = 11;
    notify e;
  }
};

behavior B(int myA, int myB, event e) {
  void main(void) {
    wait e;
    myB = myA;
  }
};

behavior Main(void) {
  int myA;
  int myB;
  event e;

  A a(myA, e);
  B b(myA, myB, e);

  int main(void) {
    par { a; b; }
    printf ("%d", myB);
    return 0;
  }
};
```

Listing 5: **e)**

f) What code has to be inserted at the beginning of behavior B (line 13) in (e) to change the output of the program? Give two different options. What must not appear there for the program not to deadlock?

g) Why do SpecC events have a semantic in which they can get lost? Under what condition do SpecC events get lost? What type of channel/communication could not be modeled if delivery would always be guaranteed?

Administrative

Advisors: Kim Grüttner <kim.gruettner@dlr.de>
Jörg Walter <joerg.walter@offis.de>
Henning Schlender <henning.schlender@dlr.de>
Sven Mehlhop <sven.mehlhop@offis.de>