**UNIVERSITY OF CALIFORNIA**
**College of Engineering**
**Department of Electrical Engineering and Computer Sciences**

**EECS 249** **A. Sangiovanni-Vincentelli**
**Fall 2012** **P. Nuzzo**

# Homework 1
# Due Thursday, September 13, 2012

**General remark: since we will mostly be dealing with design problems throughout this class, an answer to a homework problem may not necessarily be right or wrong. Grade is established mostly based on the reasoning that you follow while developing your solution. Therefore, make sure that you justify all your claims. You can use any kind of sources as long as you include references.**

**The goal of HW1 is to provide examples of orthogonalization of concerns and platform-based design.**

1. **A simple producer-consumer example.** Producer and consumer are abstraction of computational blocks that might be very complex: think of two mobile phones for instance. In our case, the producer is very simple; it generates random data, and we assume the existence of a library function `randomdata()`, which returns a random object (where the returned type can be whatever you need it to be). The producer generates data on demand, while the consumer is just a sink. It receives data and has a way of signaling that the reception is completed. Assume also that there is a way of checking whether the received data is correct or not, namely there exists a library function `checkdata(data)` which returns true if the data is correct and false if it is corrupted. Finally, consider two protocols:
    - *Simple dropping*: the consumer checks if the received data is corrupted, in which case the data is dropped and a completion signal is sent back to the producer.
    - *Retransmission*: the consumer checks if the received data is corrupted, in which case it asks the producer to send the data again until the reception is successful and a completion signal is sent back to the producer.

    a) Write the two concurrent processes `producer` and `consumer` in the case of simple dropping protocol. You can use finite state machines, process networks, VHDL, Verilog, C++, Java or pseudocode (as long as it is understandable) to describe them.

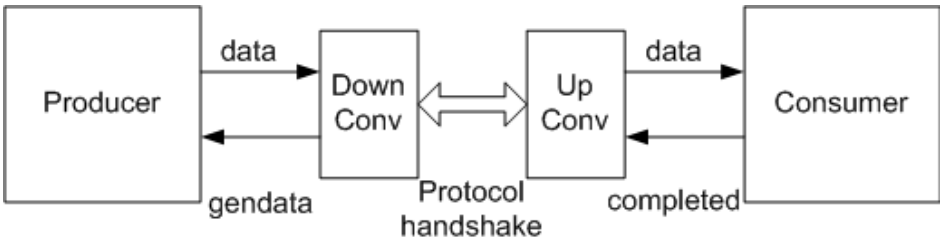
    b) Write now the two concurrent processes `producer` and `consumer` in the case of a retransmission protocol, using the same models of computation or languages as in (a)? How much of the implementation developed in (a) could you actually reuse?

    c) By recalling that producer functionality is to produce data on demand, while consumer functionality is to receive data and ask for another one, write now a process `producer` that has an input named `gendata` and an output named `data`. Producer has to generate a new random data whenever the signal `gendata` is enabled.

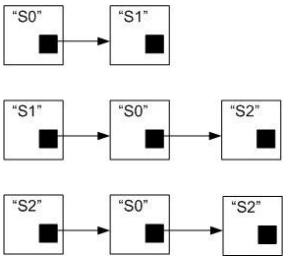
    d) Write a process `consumer` that has an input named `data` and an output named `completed`. Consumer has to enable the `completed` signal each time that a data is received. Note that those two processes in (c) and (d) can be connected together. The composed system is an autonomous system where producer keeps on producing data and receiver keeps on receiving data. We just modeled the functionality of both processes and connected them together.

    e) Now think of the protocol between the two components as two new blocks, as in the following figure, where two other processes are introduced, a *down* converter and an *up* converter. Write

the up and down converter in the case of simple dropping and retransmission. You can define the protocol handshaking as you like. Do you notice any difference between the approach followed in (c)-(d)-(e) and the one in (a)-(b)?




2. **Graphs and Breadth First Search (BFS).** We want to represent a graph by using its adjacency list. Our building block is then a data structure containing a label to name each node and a pointer to a list of adjacent nodes. The adjacency list representation of a graph is used in many computer programs to store a graph in an internal data structure. An example of a directed graph implemented using this method is shown in the figure below.




a) The figure above shows an implementation of a *specification* using a particular *architecture*. What is the *specification*? (Hint: recall the definition of a graph).
b) We now want to implement the classical breath first search (BFS) algorithm using the adjacency list representation of a graph described above. Also, we assume that a queue data structure is available to store nodes that are visited. An implementation of the BSF is shown in the figure below. What is now its purely functional description?
c) Based on the answer to part (b), can you identify a set of library elements (functions) that an *architecture* should have in order to implement BFS? Using these components, can you implement other algorithms?

---

**Algorithm 1** BFS

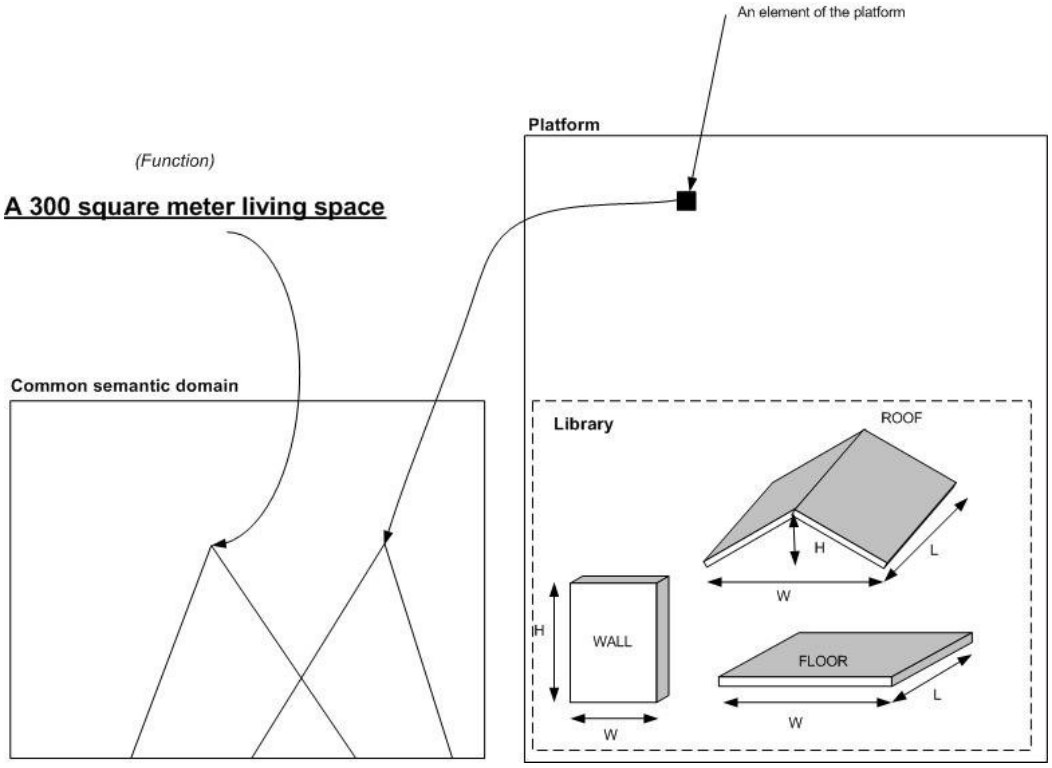
```
1:  BFS(G,s)
2:  for all u ∈ V(G) do
3:      color[u] ← WHITE
4:      π[u] ← NIL
5:  end for
6:  color[s] ← GRAY
7:  π[s] ← NIL
8:  Q ← {s}
9:  while Q ≠ ∅ do
10:     u ← head[Q]
11:     for all v ∈ Adj[u] do
12:         if color[v] = WHITE then
13:             color[v] ← GRAY
14:             π[v] ← u
```

15: *ENQUEUE*(*Q, v*)
16: **end if**
17: **end for**
18: *DEQUEUE*(*Q*)
19: *color*[*u*] ← *BLACK*

20: **end while**

3. **Platform Based Design (PBD).** We want to apply the PBD concept to an example taken from civil engineering (note that this is an example and it is not the way they follow to build houses). Our setting is shown in the figure below.
The function requires us to build a 300 square meter of living space. The platform consists of the following library of components:
- Walls: they are characterized by height and a width;
- Floors: they are characterized by width and length;
- Roofs: they are characterized by width, length and height.

We have the following composition constraints:
(a) Floors have to lie on walls and must be horizontally placed;
(b) Walls have to lie on floors and must be vertically placed;
(c) Every platform instance has to start with a floor and terminate with a roof.
The cost of the platform instance is the total surface of its components. Note that surface is an additive quantity like energy.
We have the following design constraint to satisfy: given any two points of the living space, their distance should not be greater than 10 meters. Our objective is to find a minimum cost solution that satisfies the constraint. The common semantic domain is composed of all living spaces obtained as juxtaposition of parallelepipeds (either horizontal or vertical).



a) Given the functional description above, denote the set that is obtained by mapping the functional description into the common semantic domain.
b) Characterize at least two different platform instances.
c) Given a platform instance, characterize the set obtained by mapping it into the common semantic domain.
d) Is Evans Hall an instance of this platform? And what about Soda Hall?
e) Pick an implementation of the function in the common semantic domain, which satisfies the constraints.
f) Compute the cost of your implementation.
g) Based on the lecture notes and your answers to the questions above, how would you summarize the concept of PBD? Could you give your own example of PBD in an application domain you are interested in?