Questions and Answers for the Lectures on HCI

Module 7

1. Why we need formal dialog representation?

   Answer: We need (formal) techniques to represent dialogs, which serves two purposes: (a) It helps to understand the proposed design better, and (b) formal representation makes it possible to analyze dialogs to identify usability problems (e.g., we can answer questions such as "does the design actually support undo").

2. How we can model complex dialogs with the STN? Where STNs fail?

   Answer: STNs models dialogs in terms of states (of the system) and transitions between states. For complex dialogs, the number of states and transitions becomes very large. Consequently, it becomes very cumbersome, if not impossible, to represent complex dialogs with simple STNs. This problem can be overcome with the use of the hierarchical STNs (see slides 17-18, Lecture 1. Module 7 for more details).

   STNs are not suitable to model concurrent dialogs.

3. Mention the three types of states supported by StateChart notation? What is the significance of the history mechanism in StateChart?

   Answer: The three types of states are: (a) active state, (b) super state, and (c) basic state (see slide 6-7, Lecture 2, Module 7 for details).

   The history mechanism, in a sense, acts as the memory of the StateChart. With this, it is possible to "remember" the basic state the system was in before exiting a super state, so that the next time the same super state is entered, the state transition starts from the same basic state.

4. Mention the elements of a (classical) Petri Net. How Petri Nets work?

   Answer: There are four components of a Petri Net: place, transition, arc and token (see slides 5-9, Lecture 3, Module 7 for more details on the elements).

   Petri Nets work by "firing" of transitions. At any instant, the tokens are distributed over places. This distribution represents the *state* (also known as *marking*) of the system. If all the input places of a transition contains at least one token, the transition is *enabled* and can *fire*. In a firing, each input place of the enabled transition loses one token and each output place gains one, resulting in a redistribution of tokens and consequently a new

state of the system. It may be noted that the Petri Net is non-deterministic: if multiple transitions are enabled at the same time, only one will fire.

5. Discuss the state and action properties that we can check by formally representing dialogs.

   Answer: There are mainly three action properties that a usable system should satisfy

   a) Completeness: it checks if all the transition leads to acceptable/final states or there are some missed arcs, i.e., some transition sequence don't lead to final states. Completeness ensures that a user never gets struck at some state of the dialog, which s/he doesn't want, and can't come out from there (leads to frustration and less satisfaction).

   b) Determinism: it checks if there are several arcs (transitions) for the same action. Lack of determinism introduces confusion and affects learnability and memorability, two important components of usability.

   c) Consistency: it checks whether the same action results in the same effect (state transition) always. An inconsistent interface reduces memorability and learnability, thus reducing the overall usability.

   There are mainly three state properties related to system usability

   a) Reachability: can we get to any state from any other state? Reachability ensures that all features of the system can be used.

   b) Reversibility: can we return to the previous state from the current state? Reversibility ensures that the user can recover from mistakes, thus increasing confidence and satisfaction.

   c) Dangerous states: Are there any undesirable states that lead to deadlock (i.e. no further transitions are possible)? Detection of dangerous states ensures that the user never goes to one, thus avoiding potential usability problems.