Project Report

On

Formal Verification of Partograph Application

by

Ronak Khandelwal (2013CS50295)

&

Deepankar Mishra(2013CS50282)

Supervisor:

Prof Sanjiva Prasad

Department of Computer Science and Engineering

Indian Institute of Technology Delhi

2015

Acknowledgement

We would like to acknowledge and extend our heartfelt gratitude to Prof. Sanjiva Prasad for his invaluable guidance and support given to us during our project. We would also like to thank Shibashis Ghua for his constant help and useful suggestions. We would like to express sincere thanks to our family members and friends for their constant encouragement.

Abstract

The project requires us to do the formal verification of an android based mobile application which will be used for the medical purposes in the future. The app is used to analyze the situation of the patient during labor period of pregnancy based on the readings entered by the user in the app. It includes certain notifications and pop-ups which appears depending upon various parameters. Our job is to verify that these pop-ups and notifications come irrespective of the value of other variables by creating a model of what the application is doing.

We have used UPPAAL (model verification software) to model the application and then used it verify certain important properties regarding the correctness of the application. We have modelled few parameters like Cervical Dilation, Maternal Pulse and Fetal Heart Rate. There are other properties but they can be modelled in the similar fashion. We have verified key properties using UPPAAL and writing them in CTL (verification language).

Contents

1. Introduction to Android Application 2
2. Modelling of Cervical Dilation 3
   2.1 Modelling Information 3
   2.2 Good and Bad Scenarios 5
   2.3 Important transitions involved in the model 5
   2.4 An example simulation 6
   2.5 Properties and their verification 7
3. Modelling of Other Parameters (Pulse and Fetal heart rate) 9
   3.1 Modelling Information 9
   3.2 Good and Bad Scenarios 10
   3.3 Important transitions involved in the model 10
   3.4 An Example Simulation 11
   3.5 Properties and their verification 12
   3.6 Putting everything together 12
4. Conclusion 14
   4.1 Summary 14

List of Figures

Figure 2.1: Cervical Dilation 3
Figure 2.2: User 4
Figure 3.1: Pulse and Fetal Heart Rate 9

Chapter 1

Introduction to Android Application

The android application is named Partograph which is built by IIT Delhi students. The app is made to be used by the people during the labor hours of pregnancy. The people can put in the values of various parameters and know their condition and also the actions which need to be taken depending upon the condition of the patient.

The application starts with the creation of the new user asking them various information like name of the women, name of her husband etc. It then goes to second page where the user is required to enter various readings like pulse, fetal heart rate etc. Then it moves on to another page which is also is about taking certain initial reading. If the user enters any parameters which is out of bounds, the app shows an indication by making the text box in which the user is entering green.

After this page comes the page in which the user needs to take the reading of Cervical Dilation after every one hour and also the reading of various parameters like pulse, fetal heart rate etc. half an hour before taking the reading for the Cervical Dilation. According to the readings of the Cervical Dilation, a graph between Cervical Dilation and time is plotted. There are three regions in the graph. Normal region (colored green) indicates that the patient is in the normal state. Then there is Alarm region (colored yellow) which indicates that patient condition is getting bad. After that if the patient’s condition becomes even worse, then there is an emergency region (colored red). Whenever the plot between the cervical dilation and time passes from one region to another, notification is given to the user.

Chapter 2

Modelling of Cervical Dilation

2.1 Modelling Information

In this, we have the modelled the part of the application in which the user has to enter the value of the cervical dilation after every 1 hour. The pop-up for entering the value of the cervical dilation remains there for 15 minutes. If the user enters the value, it gets plotted on the graph otherwise the pop-up goes away and again comes after 1hour.

Our model of cervical dilation consists of 8 locations. There are transitions from one location to another depending on the nature shown by the application. There is a local clock which keeps track of 60 minutes and there is an integer variable which keeps track of how much hours have passed by.

In the model, we have allowed the user to enter the readings of the Cervical Dilation from 4 to 11 in which 4 to 10 are admissible and after 10, the readings become inadmissible.

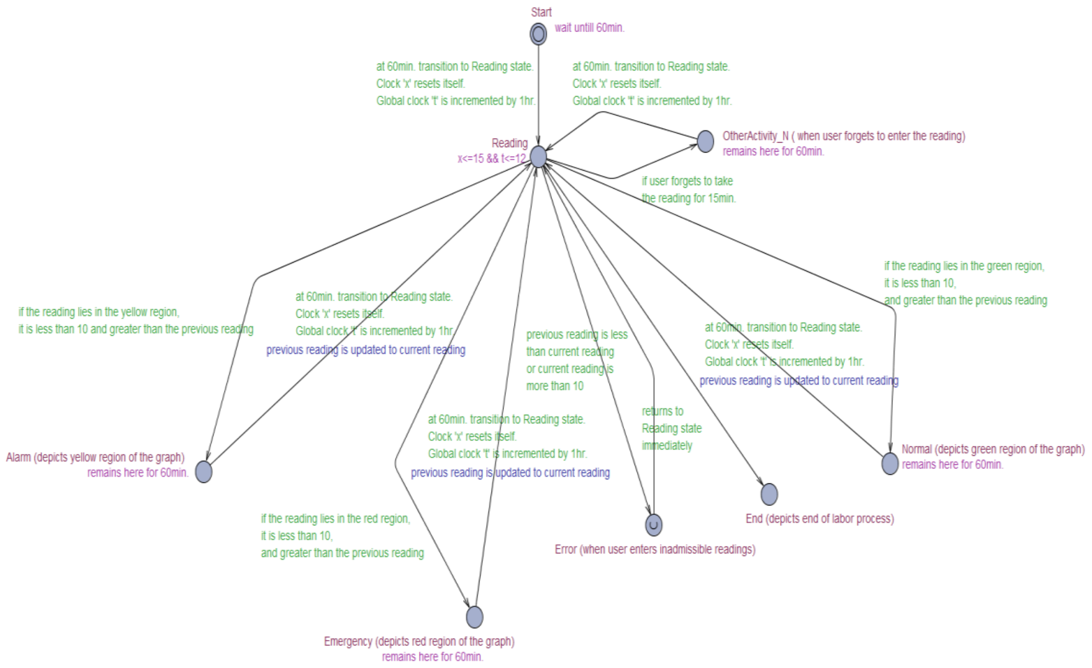
Start
wait untill 60min.
at 60min. transition to Reading state.
Clock 'x' resets itself.
Global clock 't' is incremented by 1hr.
at 60min. transition to Reading state.
Clock 'x' resets itself.
Global clock 't' is incremented by 1hr.
Reading
x<=15 && t<=12
OtherActivity_N (when user forgets to enter the reading)
remains here for 60min.
if user forgets to take
the reading for 15min.
if the reading lies in the green region,
it is less than 10,
and greater than the previous reading
if the reading lies in the yellow region,
it is less than 10 and greater than the previous reading
at 60min. transition to Reading state.
Clock 'x' resets itself.
Global clock 't' is incremented by 1hr.
previous reading is updated to current reading
previous reading is less
than current reading
or current reading is
more than 10
at 60min. transition to Reading state.
Clock 'x' resets itself.
Global clock 't' is incremented by 1hr.
previous reading is updated to current reading
returns to
Reading state
immediately
at 60min. transition to Reading state.
Clock 'x' resets itself.
Global clock 't' is incremented by 1hr.
previous reading is updated to current reading
Alarm (depicts yellow region of the graph)
remains here for 60min.
Normal (depicts green region of the graph)
remains here for 60min.
End (depicts end of labor process)
Error (when user enters inadmissible readings)
if the reading lies in the red region,
it is less than 10,
and greater than the previous reading
Emergency (depicts red region of the graph)
remains here for 60min.

Figure 2.1: Cervical Dilation

Start: Initial location which signifies that the user has just entered into the Cervical Dilation page for the first time after creating a new user.

Reading: This location depicts the pop-up being raised after every hour. In this state of the model, the user is supposed to enter the value of the Cervical Dilation. The automaton remains in this location for at-most 15 units of time.

Other_Activity_N: If the user forgets to enter the reading within this time period, then the automaton moves to this location and resides here for 60 minutes before going into Reading state.

Normal, Alarm and Emergency state are attained depending upon where the point is marked on Cervical Dilation graph.

For deciding the transitions to these states, we have used basic equations which gives a positive or negative result depending upon whether the point lies to the left or the right of the line when we put the point in the equation of the lines dividing the graph into three regions named normal, alarm and emergency

Error: If the user enters the value of Cervical Dilation that is less than the previous value entered or if the value is greater than 10, then it attains error state.

End: If the time of labor has become more than 12 hrs, then the automaton moves to the end state which depicts that the labor period has ended.

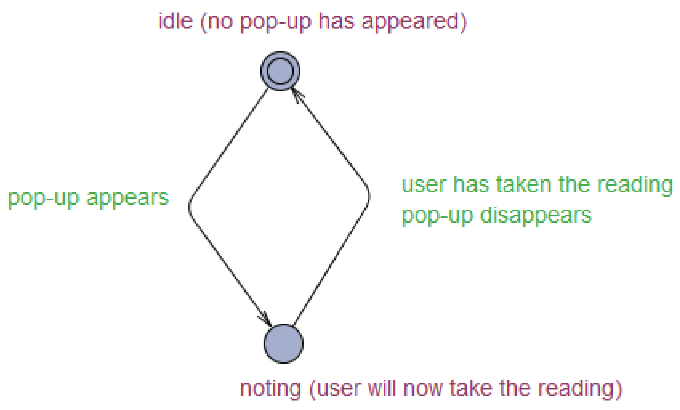
idle (no pop-up has appeared)
pop-up appears
user has taken the reading
pop-up disappears
noting (user will now take the reading)

Figure 2.2: User

The model also contains a timed automaton named **User**. It consists of two states named **Idle and Noting**. Idle (initial location) state signifies that no pop-up is there for taking the readings. The Noting state signifies that the user is noting the value of cervical dilation .This automaton depicts whether the user is taking the reading or user is sitting idle. It is also used to denote the reading of the cervical dilation that the user is going to enter in the application after each hour. The readings taken by the user is decided during the simulation time in UPPAAL.

2.2 Good and Bad Scenarios

The good scenarios involved in the application are that the pop-up for taking the reading of the Cervical Dilation appears after every hour irrespective of whether the person enters the reading or forgets to enter the reading. Also it appears irrespective of what the readings are. Also the graph of Cervical Dilation goes from green region to yellow region and then to red region depending on the values of the Cervical Dilation at a particular moment indicating the condition of the patient. Also the reading of the Cervical Dilation should be between 4 and 10 and it should be greater than or equal to the previous reading of the Cervical Dilation.

The bad scenarios are the ones in which an error notification is given to the user. In the case of Cervical Dilation, the error notification is given if the value entered by the user is greater than 10 or the present value entered is less than the previous value of Cervical Dilation.

2.3 Important transitions involved in the model

Initially the model is present in the **start** state.

1. After spending 1hr in start state, it will move on to **reading state** in which the user takes the reading.

2. The model remains in the reading state for another 15 minutes.
   a. If the user takes the reading in this span of time, it goes to either normal, alarm, emergency or error state depending upon the readings
   b. If the user doesn't take the reading, then it goes **from reading to Other_Activity_N state** (meaning the pop-up for taking the reading disappears). It stays in this state for 60min before going to Reading state again and then the same process is repeated.

3. If the reading is less than the previous reading, it goes **from reading to error state** (notification is raised in the app), and it again moves to reading state without spending any time in the error state.

4. If 12hrs have been passed in taking the readings, then the model moves **from reading to end state** (meaning the labor process has ended) irrespective of whatever the reading user has entered.

5. If the reading is such that the marking in the graph remains in the green region only (condition has been taken care off as a guard in the transitions from reading to normal state), then it moves **from reading to normal state**. It spends the remaining time(time at which 1hr is completed from the point when it has moved to reading state before coming to normal state) in the normal state, and then again moves to the reading state.

6. If the reading is such that the point in the graph is in the yellow region, transition is made **from reading to alarm state**, from where it moves to reading state as in the case of normal state.

7. If the reading is such that the point in the graph is in the red region, transition is made **from reading to emergency state**, from where it moves to reading state as in the case of normal state.

2.4 An example simulation

The model starts from Start state. In this example simulation, we are showing transitions until 6hrs.

1. After 1hr, model makes transition from start to reading state. (t=1hr)

2. After taking 4 as the reading of cervical dilation, the model moves to normal state. (t=2hr)

3. After 1hr, it moves to reading state. It then spends 15min in the reading state and the user forgets to take the reading, therefore it moves to Other_Activity_N state. (t=3hr)

4. After 1hr, it moves to reading state again, then it repeats the same transition and it again moves to reading state after 1hr. (t=4hr)

5. When the user enters 11 as the reading of cervical dilation, it moves to error state from which it again moves to reading state, without spending any time in the error state.

6. After that it moves to normal state when the user enters 4 in the cervical dilation and after 1hr, it moves to reading state again.(t=5hr)

7. When the user again enters 4 as the reading of the cervical dilation, it moves to alarm state due to the guard conditions, and after 1hr, it moves to reading state again. (t-6hr)

Similarly, if we do the simulation for more time, we can show it moving to emergency state, and after t=12hrs it moves to end state which is the deadlock state of the model.

2.5 Properties and their verification

The various properties needs to be verified in order to prove the correctness of the application, so that it can be used for medical purposes. The properties are as follows:-

1. If the present reading is less than the previous reading, then the model is in the error state.
   CTL language: A[] forall (i :readings) User(i).idle and (i<Cervical_Dilation.pd) imply Cervical_Dilation.Error
   Explanation: In the above property specification, i is for the present reading and pd is for the previous reading of Cervical Dilation. User(i).idle means the user has selected i as the reading of Cervical dilation.

2. If the model is in the error state, then it is always the case that present reading of Cervical Dilation is less than the previous reading of the Cervical Dilation. (this property is the converse of the property stated above)
   CTL language: A[] forall (i:readings) User(i).idle and Cervical_Dilation.Error imply i<Cervical_Dilation.pd
   Explanation: Similar to the previous explanation

3. If the reading is marked in the green region of the graph of Cervical Dilation, then at that time it is always in the normal state.
   CTL language: A[] forall (i : readings) User(i).idle and (i>=Cervical_Dilation.pd) and (i<10) and (i>=Cervical_Dilation.t) and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0) imply (Cervical_Dilation.Normal)

4. If the reading is marked in the yellow region of the Cervical Dilation, then at that time it is always present in the alarm state.
   CTL language: A[] forall (i : readings) User(i).idle and (i>=Cervical_Dilation.pd) and (i<10) and (i<Cervical_Dilation.t) and (i+2>=Cervical_Dilation.t) and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0) imply (Cervical_Dilation.Alarm)

5. If the reading is marked in the red region of the Cervical Dilation, then at that time it is always present in the emergency state.
   CTL language: A[] forall (i : readings) User(i).idle and (i>=Cervical_Dilation.pd) and (i<10) and (i<Cervical_Dilation.t - 2) and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0) imply (Cervical_Dilation.Emergency)

The converse of the three properties are stated as follows in their respective order:

6. If the model is present in the normal state, then the reading of Cervical Dilation is marked in the green region of the graph.
   CTL language: A[] forall(i:readings) User(i).idle and Cervical_Dilation.Normal imply ((i>=Cervical_Dilation.pd) and i<10 and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0) and (i>=Cervical_Dilation.t))

7. If the model is present in the alarm state, then the reading of Cervical Dilation is marked in the yellow region of the graph.
   CTL language: A[] forall (i:readings) User(i).idle and Cervical_Dilation.Alarm imply ((i>=Cervical_Dilation.pd) and (i<10) and (i<Cervical_Dilation.t) and (i+2>=Cervical_Dilation.t) and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0))

8. If the model is present in the emergency state, then the reading of Cervical Dilation is marked in the red region of the graph.
   CTL language: A[] forall (i:readings) User(i).idle and Cervical_Dilation.Emergency imply ((i>=Cervical_Dilation.pd) and (i<10) and (i<Cervical_Dilation.t - 2) and (Cervical_Dilation.t<=12 and Cervical_Dilation.t>0))

9. After every 1hr, the pop-up for taking the reading appears irrespective of present state of the application. This property is proved by the combination of five properties which are written in the CTL language as follows
   a. **CTL language:** Cervical_Dilation.Start --> (Cervical_Dilation.Reading and Cervical_Dilation.x ==0)
   b. **CTL language:** Cervical_Dilation.Normal --> (Cervical_Dilation.Reading and Cervical_Dilation.x == 0)
   c. **CTL language:** Cervical_Dilation.Alarm --> (Cervical_Dilation.Reading and Cervical_Dilation.x == 0)
   d. **CTL language:** Cervical_Dilation.Emergency --> (Cervical_Dilation.Reading and Cervical_Dilation.x ==0)
   e. **CTL language:** Cervical_Dilation.OtherActivity_N --> (Cervical_Dilation.Reading and Cervical_Dilation.x ==0)

   Explanation: The clock variable x resets itself after every 60 units of time. Therefore, x=60 and x=0 are one and the same thing. The property states that whatever the state the model is in, it eventually moves to reading state and at that time, value of the clock variable is 0 or 60. Therefore it eventually moves to reading state from any state, when x=60 and since clock x is periodic with a period of 60 units, therefore it is proven that it moves again and again to reading state after every 1hr. Therefore the pop-up for taking the reading comes after every 60min.

Chapter 3

Modelling of Other Parameters (Pulse and Fetal heart rate)

3.1 Modelling Information

The other parameters involve pulse, fetal heart rate, blood pressure etc. We have modelled pulse and fetal heart rate only as other parameters like blood pressure, temperature can be modelled in the similar way.

For the purpose of the model, the pulse reading can vary from 0 to 10. The correct value ranges from 3 to 7. Other values are considered to be out of range.

Similarly, fetal hart rate reading can vary from 10 to 20. The correct value ranges from 13 to 17. Other values are considered to be out of range.

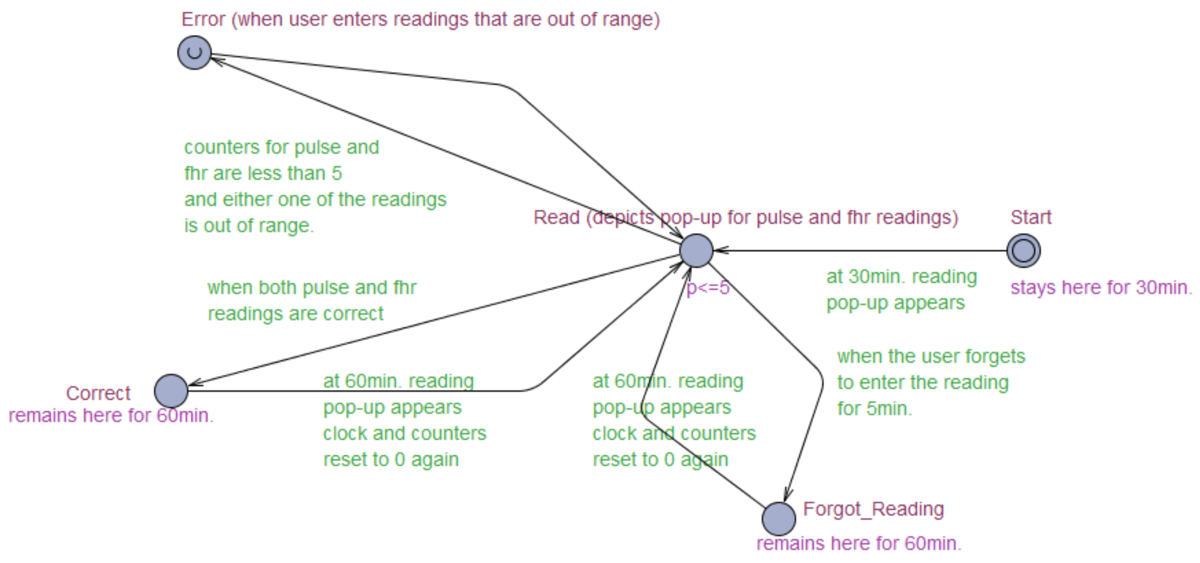
Error (when user enters readings that are out of range)
counters for pulse and fhr are less than 5 and either one of the readings is out of range.
Read (depicts pop-up for pulse and fhr readings)
Start
p<=5
at 30min. reading pop-up appears
stays here for 30min.
when both pulse and fhr readings are correct
Correct
remains here for 60min.
at 60min. reading pop-up appears clock and counters reset to 0 again
at 60min. reading pop-up appears clock and counters reset to 0 again
when the user forgets to enter the reading for 5min.
Forgot_Reading
remains here for 60min.

Figure 3.1: Pulse and Fetal Heart Rate

In this model, we have made 5 locations which are as follows:

Start: It is the initial location of the model. It remains here for 30min.
Read: From start state, it goes to Read state after 30min. which depicts the reading pop-up for pulse and fetal heart rate has appeared.
Forgor_Reading: If the user doesn't enter the reading for 5min, then the pop-up disappears and the model moves to this state.
Correct: If both the reading entered are correct, then it moves to Correct state where it remains for 60min. before going to Read state again.
Error: If either of the readings entered is out of range, then it moves to this state and it goes into Read state again without spending any time in the Error state.

3.2 Good and Bad Scenarios

When the value entered for parameters like pulse, fetal heart rate, blood pressure etc. are in the range as specified by the medical practitioner, no indication is shown which means that the application is in the good scenario.

But when the value of these parameters is out of bounds, then the respective text box is made green, which indicates that the application has gone to bad scenario and the user has entered wrong value of the parameter. For example, the range for pulse reading is 60 to 100. So if the person enters the value of the pulse less than 60 or greater than 100, then the text box related to pulse reading is made green, and thus the person is notified. This is the bad scenario in the application related to other parameters.

3.3 Important transitions involved in the model

Initially, the model is present in the **Start** state.

1. After 30 minutes it moves **from Start to Read state** where it remains for 5 minutes.
   a. If the user takes the reading within 5 minutes it moves to Correct or Error state.
   b. If the user forgets to take the reading in 5 minutes, the pop-up for taking the reading disappears and the model goes **from Read to Forgot_Reading state**.

2. If the readings of pulse and foetel heart rate are correct, then it moves to **Correct state** where it stays for 60min and then it again moves to Read state where the user needs to take the readings again.

3. If any one of the readings in out of range, then it moves to **Error state**, and again moves to Read state without spending any time in this state

It can go into the error state only for 5 times. Even after that, if the user enters incorrect reading, then it moves to Forgot_Reading state and counters get reset when it again comes to Read state from either the correct or Forgot_Reading state.

3.4 An Example Simulation

The simulation starts from Start state.

1. After 30 units of time, it moves to Read state. This transition signifies the appearance of pop-up after 30 minutes.

2. The user then enters 16 as pulse and 25 as fhr (Fetal heart rate) reading. Since both the readings are in the proper range, the model makes a transition from Read to Correct state.

3. From Correct state, it again moves to Read state after 60 units of time. In the Read state, the user enters 13 as pulse and 21 as fhr reading. Fhr reading is out of bounds, therefore it moves to Error state, from where it again moves to Read state without wasting any time in the Error state.

4. After this, the user forgets to take the reading within 5 units of time, therefore the model moves to Forgot_Reading state where it spends the remaining of 1hr time.

5. After spending the remaining time, it again moves to Read state where the user enters the readings again.

6. The user now enters 19 as pulse and 26 as fhr. Now the pulse reading is out of bounds, therefore it moves to Error state, from where it again moves to Read state without wasting any time in the Error state.

7. The user now enters 10 as pulse and 30 as fhr. Now both the readings are out of bounds, therefore it moves to Error state, from where it again moves to Read state without wasting any time in the Error state.

3.5 Properties and their verification

The various properties needs to be verified in order to verify the application for correctness, so that it can be used for medical purposes. The properties are as follows:-

1. If the values of counters for the pulse and fetal heart rate is less than or equal to 5, and if one of the readings out of pulse or fetal heart rate is out of bounds, then the model is always present in Error state.

   CTL language: A[] Other_readings.Error imply (Other_readings.count_pulse<=5 and Other_readings.count_fhr<=5 and ((Other_readings.pul<13 or Other_readings.pul>17) or (Other_readings.fhr<23 or Other_readings.fhr>27)))

   Explanation: count_pulse and count_fhr are the counters for pulse and fetal heart rate. pul and fhr denotes the reading of pulse and fetal heart rate respectively.

2. If the model is present in the Error state, then it is always the case that the counters are less than or equal to 5 and either one of the readings out of pulse or fhr is out of bounds (This property is the converse of the above property).

   CTL language: A[] Other_readings.Error imply (Other_readings.count_pulse<=5 and Other_readings.count_fhr<=5 and ((Other_readings.pul<13 or Other_readings.pul>17) or (Other_readings.fhr<23 or Other_readings.fhr>27)))

These two properties also verify that if the user enters the correct readings, then it moves to Correct state.

We can model other properties of the application using UPPAAL in a similar way and verify the properties related to those models.

3.6 Putting everything together

Now when the models for all the parameters are put together and simulated at the same time, then

1. After half hour, a pop-up for taking the readings for pulse, fetal heart rate etc. appears i.e. the Other_Readings model moves to Read state. The model then works in the same way as it would have been if simulated alone. This then repeats itself after every 1 hour.

2. After another 30min, pop-up for taking the readings of the cervical dilation appears i.e. the Cervical_Dilation model moves to Reading state where the user enters the reading according to which the next state in the model is chosen. This repeats after every hour.

3. The overall model works for 12hrs and then the Cervical_Dilation model moves to the End state (deadlock state) which signifies the ending of the labor process.

The safety and liveness properties that have been verified for the model taken one at a time also holds when the models are combined. Therefore, now we can say that all the properties have been verified for the complete model.

Also since the properties verified depicts the correctness of the individual model, the overall model becomes automatically correct as all the properties also holds for the combined model.

Chapter 4

Conclusion

4.1 Summary

The model of partograph application has been developed using UPPAAL. This model helped in simulation of the application which helped in understanding the working of the application. Also the verifier of the UPPAAL helped in the formal verification of various key properties which are necessary for proving correctness of the application.

During the development of models, we found that the notion of timed automata and their graphical representation served extremely well to understand the application during the simulation and verification. In addition, the graphical simulation features of UPPAAL lead to fast detection of (obvious) errors in the early models.

We also verified 9 properties for Cervical Dilation reading and 2 more properties for other readings like pulse, fetal heart rate etc. These properties have helped in proving that the application will always work correctly irrespective of the various readings that are entered by the user during the usage of the application.

So, to conclude, we can say that the application has been formally verified for the inconsistencies and no bugs were identified.