CS734 Assignment 3: Temporal Logic & Model-Checking

Due: 1330 Thursday April 25, 2002

1. Zeno's Paradox and Real-Time Systems

Give a brief summary of Zeno's paradox and describe what is meant by Zeno behaviour in a real-time systems model using an example.

2. Expressiveness of Temporal Logics

In terms of expressiveness, Linear Temporal Logic (LTL) and Computational Tree Logic (CTL) are incomparable (i.e., each can express properties that the other logic cannot. In this question we investigate some of the differences of these two logics.

a) Find an example of a formula ϕ of LTL and a Kripke structure $\mathbf{M}$ such that $\mathbf{M} \not\models \phi$ and $\mathbf{M} \not\models \neg\phi$.
HINT: This is possible because of the implicit universal quantification over paths in the definition of $\mathbf{M} \models \phi$.

b) Find a formula of CTL ψ that does not have an equivalent LTL formula.

c) Consider the LTL formula:

$$\mathrm{GF}(\eta = \mathrm{tick}) \rightarrow \mathrm{GF}(\eta = \alpha)$$

and the CTL formula:

$$AGAF(\eta = \mathrm{tick}) \rightarrow AGAF(\eta = \alpha)$$

Use one or more example Kripke structures to illustrate the different meanings of these formulas.

d) The inability of CTL to verify properties of "fair paths" has led to "Fair CTL" or CTL with fairness constraints. Specifiy your example(s) from part (c) in PVS and then check the results of the standard CTL operators from `ctlops` prelude theory with the use of Fair CTL operators from the `fairctlops` prelude theory to show how these operators pick out fair paths.

3. Model Checking Application

Consider the dining philosophers problem with 3 philosophers. Model this problem in PVS and create a decentralized control strategy that avoids deadlock and livelock.

State and prove by model-checking theorems that state that:

a) *In your system there exists at least one fair path where each philosopher eats infinitely often.*

b) *In every state of you system, a path is a always possible to a state where a given philosopher is eating.*

c) *Every philosopher gets to eat infinitely often assuming that no philosopher will eat continuously forever.*
To do this last part you will need to impose a fairness constraint on each philosopher to the effect that the philosopher will not eat forever (i.e., each of the philosophers will be Thinking infinitely often).
HINT: The best way to do this is probably to use the `Fairctlops` operators (see the bottom of the PVS prelude file) to consider only fair paths.

BONUS: Repeat the above for the 5 philosopher problem.