**15-424/15-624 Recitation 9**
**dTL semantics**

1. **dTL: intuition for those pesky semantics!**

   In dTL, you have to keep track of the entire execution of a program, not just the end states like in $\mathsf{d}\mathcal{L}$. Thus, every time a piece of program is executed, a new function $\mu$ is appended to a trace.

   This function is, for ODEs, the solution to the IVP (initial value problem, ODE + starting state). For example, if the previous state ended at $x = 1$, and $x' = 1$ is executed, then *one* of the potential $\mu$'s domain's is $(0, 5)$, $\mu(0)(x) = 1$, and $\mu(5)(x) = 6$.

   For assignments or tests, it's like the solution to the IVP, but it takes 0 time, and the starting state is whatever was effected by the transformation. Let's continue the previous example, where $\mu(5)(x) = 5$. Now, $x := x + 1$ is executed. Then you will append a new $\mu_1$, which is only defined for 0, and $\mu(0)(x) = 7$.

   To simplify things, the notation $\widehat{\mu}$ transforms a regular state $\mu$ into a trace component, i.e. $\mu(x) = \widehat{\mu}(0)(x)$, for all variables. This ensures traces are only made up of functions.

   Now consider the following program

   $$\alpha \equiv a := -b; z' = v, v' = a; ?v \geq 0; a := A; z' = v, v' = a$$

   This program corresponds to the scenario where a robot first decides to brake, then accelerates. Since braking can't make robots go backwards, there's a test in between the ODEs that ensures this holds.

   There are two potential sets of traces here. One set in which the test failed, and one set in which it didn't. Let's start with the one where it failed.

   - *Failed test:* The traces will be of the form $(\widehat{\mu_0}, \widehat{\mu_1}, \varphi_1, \widehat{\Lambda})$. Assignment creates two states, the origin and end states, resp. $\widehat{\mu_0}$ and $\widehat{\mu_1}$. These are defined similarly to $\mathsf{d}\mathcal{L}$, where $\mu_1 = \mu_0[x \mapsto [\![-b]\!]_{\mu_0}]$. Then, $\varphi$ is simply the solution to the IVP for some duration $t$, so $\varphi_1(0) = \mu_1$. Because the test failed, we know that $\varphi(t)(x) < 0$. Finally, the system transitions to the error state because of the failed test - but the trace is not eliminated, like it is in $\mathsf{d}\mathcal{L}$!
   - *Passed test:* The traces will be of the form $(\widehat{\mu_0}, \widehat{\mu_1}, \varphi_1, \widehat{\mu_2}, \widehat{\mu_3}, \varphi_2)$. The half of the trace is the same. Here, however, $\varphi_1(t) = \widehat{\mu_2}$, where $t$ is however long the first ODE evolved for. The assignment is as before, so $\mu_3 = \mu_2[x \mapsto [\![-b]\!]_{\mu_2}]$. Finally, the $\varphi_2$ is the solution to the IVP starting at $\widehat{\mu_3}$. It can evolve for however long!

   These two sets of traces define all the possible executions of the problem. Let's analyse all temporal properties that can can be derived from this program for the question $v < 0$. Assume $-b$ is negative and $A$ is positive.

- $\langle\alpha\rangle\Diamond v < 0$: valid. If $v$ starts negative, we can just pick that state to prove $v < 0$. Even if $v$ starts out positive, there's a trace that can make it negative before failing the test. Since traces are not pruned, even when they fail tests, we can take that, and the first state at which $v$ became negative during that trace, to satisfy $v < 0$.
- $\langle\alpha\rangle\Box v < 0$: satisfiable. It's falsifiable whenever $v$ starts positive. No matter how the program executes, that initial state will ensure that no trace will have negative $v$ throughout. It's satisfiable if we choose a trace where $v$ starts negative. It will stay negative no matter how long it evolves, and even though it won't pass the test, we can still use that trace as a witness for the satisfiability of $\Box v < 0$.
- $[\alpha]\Diamond v < 0$: unsatisfiable. We must consider all possible traces of the program, in particular, the one where $v$ starts positive but does not evolve for long enough for $v$ to become negative.
- $[\alpha]\Box v < 0$: unsatisfiable. The same reasoning as above can be used, since this property is even stronger than the $\Diamond$ one.

Hopefully this illustrates how difficulty in proving these progresses as we change from $\Diamond$ to $\Box$ modalities in the hybrid program sense as well as the temporal sense. Notice how getting to pick a trace is still stronger than getting to pick whether the property needs to hold throughout or just at some point!

2. **dTL semantics and proof rules**

   The fact that these two sets of traces result from interpreting the program helps us better understand the sequence proof rule.

$$\frac{[\alpha]\Box\phi \wedge [\alpha][\beta]\Box\phi}{[\alpha;\beta]\Box\phi}$$

   The RHS of the conjunction is obvious - we know it from $\mathsf{d}\mathcal{L}$, but why do we need that first one? Partly, we need to check that property for the first set of traces, the ones where the test failed. Traces in dTL are like weeds! No matter how much you try to prune them, *they always come back*.

   More accurately, we need it because without a temporal modality, $[\alpha]\phi$ only looks at the end of the traces. In other words, what $[\alpha][\beta]\Box\phi$ checks is whether $\phi$ holds throughout traces of $\beta$, where you start at an end state of $\alpha$. It doesn't actually check that $\phi$ holds throughout $\alpha$, but that's what we want to happen.

3. We also checked a question that everyone got wrong in the exam, including your TAs! But unfortunately, no solutions can be given out on the interwebs, so have a smiley ☺.