15-424/15-624 Recitation 2
Hybrid Programs: Syntax, Semantics, and an Example

1. Quiz

   Explain the difference between syntax and semantics:

   Syntax defines a notation (e.g. the grammars for terms, formulas, and hybrid programs). Semantics gives the syntax meaning (e.g. valuation of terms, the satisfaction relation $\nu \models F$ for a first-order formula, and the transition semantics ρ for HPs).

   Fill in the missing semantics for the following terms and formulas:

   (a) $[\![x]\!]_\nu = \nu(x)$ where x is a variable

   (b) $[\![c]\!]_\nu = c$ where c is a rational constant

   (c) $[\![\theta_1 + \theta_2]\!]_\nu = [\![\theta_1]\!]_\nu + [\![\theta_2]\!]_\nu$

   (d) $\nu \models (\theta_1 = \theta_2)$ iff $[\![\theta_1]\!]_\nu = [\![\theta_2]\!]_\nu$

   (e) $\nu \models F \wedge G$ iff $\nu \models F$ or $\nu \models G$

2. Syntax & Semantics

   In recitation we reviewed the syntax and semantics of terms, formulas, and hybrid programs. There was some confusion about where you can find these definitions.

   Syntax of Terms.

   `http://symbolaris.com/course/fcps13/02-diffeq.pdf#page=8`

   The grammar for terms can be found in Lecture 2, Section 5. A term θ is defined by the grammar (where θ, ϑ are terms, x a variable, and c a rational constant):

$$\theta, \vartheta ::= x \mid c \mid \theta + \vartheta \mid \theta \cdot \vartheta$$

   Semantics of Terms.

   `http://symbolaris.com/course/fcps13/02-diffeq.pdf#page=9`

   We define the semantics of a term by its value, according to the valuation function defined in Lecture 2, Section 6.

   The *value of term* θ in state ν is denoted $[\![\theta]\!]_\nu$ and defined by induction on the structure of θ:

$$\begin{aligned}
[\![x]\!]_\nu &= \nu(x) && \text{if } x \text{ is a variable}\\
[\![c]\!]_\nu &= c && \text{if } c \text{ is a rational constant}\\
[\![\theta + \vartheta]\!]_\nu &= [\![\theta]\!]_\nu + [\![\vartheta]\!]_\nu\\
[\![\theta \cdot \vartheta]\!]_\nu &= [\![\theta]\!]_\nu \cdot [\![\vartheta]\!]_\nu
\end{aligned}$$

Syntax of Formulas.

`http://symbolaris.com/course/fcps13/02-diffeq.pdf#page=9`

The grammar for formulas can be found in Lecture 2, Section 5.

$$F, G ::= \theta = \vartheta \mid \theta \geq \vartheta \mid \neg F \mid F \wedge G \mid F \vee G \mid F \rightarrow G \mid F \leftrightarrow G \mid \forall x\, F \mid \exists x\, F$$

Semantics of Formulas.

`http://symbolaris.com/course/fcps13/02-diffeq.pdf#page=10`

We define the semantics of a formula by a satisfaction relation $\nu \models F$. In other words, $\nu \models F$ means that F is satisfied in the state ν. If F is satisfied in all states ν, then we call F *valid* and write $\models F$. A full discussion of the semantics of formulas can be found in Lecture 2, Section 6.

The *satisfaction relation* $\nu \models F$ for a first-order formula F of real arithmetic in state ν is defined inductively:

- $\nu \models (\theta_1 = \theta_2)$ iff $[\![\theta_1]\!]_\nu = [\![\theta_2]\!]_\nu$.
- $\nu \models (\theta_1 \geq \theta_2)$ iff $[\![\theta_1]\!]_\nu \geq [\![\theta_2]\!]_\nu$.
- $\nu \models \neg F$ iff $\nu \not\models F$, i.e. if it is not the case that $\nu \models F$.
- $\nu \models F \wedge G$ iff $\nu \models F$ and $\nu \models G$.
- $\nu \models F \vee G$ iff $\nu \models F$ or $\nu \models G$.
- $\nu \models F \rightarrow G$ iff $\nu \not\models F$ or $\nu \models G$.
- $\nu \models F \leftrightarrow G$ iff ($\nu \models F$ and $\nu \models G$) or ($\nu \not\models F$ and $\nu \not\models G$).
- $\nu \models \forall x\, F$ iff $\nu_x^d \models F$ for all $d \in \mathbb{R}$.
- $\nu \models \exists x\, F$ iff $\nu_x^d \models F$ for some $d \in \mathbb{R}$.
- $\nu \models [\alpha]\phi$ iff $\omega \models \phi$ for all ω with $(\nu, \omega) \in \rho(\alpha)$
- $\nu \models \langle\alpha\rangle\phi$ iff $\omega \models \phi$ for some ω with $(\nu, \omega) \in \rho(\alpha)$

If $\nu \models F$, then we say that F is true at ν or that ν is a model of F. A formula F is *valid*, written $\vDash F$, iff $\nu \models F$ for all states ν. A formula F is a *consequence* of a set of formulas Γ, written $\Gamma \vDash F$, iff, for each ν: $\nu \models G$ for all $G \in \Gamma$ implies that $\nu \models F$.

The modal operators $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are introduced in Section 8 for Lecture 4:
`http://symbolaris.com/course/fcps13/04-contracts.pdf`

Syntax of Hybrid Programs

`http://symbolaris.com/course/fcps13/03-choicecontrol.pdf#page=9`

The grammar for hybrid programs can be found in Lecture 3, Section 7.

HPs are defined by the following grammar (α, β are HPs, x a variable, θ a term possibly containing x, and H a formula of first-order logic of real arithmetic):

$$\alpha, \beta \ ::= \ x := \theta \mid ?H \mid x' = \theta \,\&\, H \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^*$$

3. Example

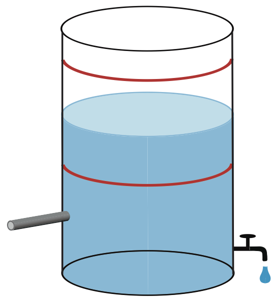

Figure 1: Water Tank Example

Suppose you have a tank of water with an input pipe that you control, and output spigot that a human user controls. You don't want the tank to overflow, so the height of the water must never go above 9 inches. You also don't want the water level to go below the pipe and the spigot, which we say are at height 0.

When the input pipe is on, the height of the water increases at 1 inch per second. When the input pipe is off, the height of the water does not increase.

However, when the spigot is open, water flows out of the tank at 1 inch per second. And when the spigot is closed, water flows out of the tank at 0.1 inches per second (it leaks).

Problem: Design a controller which will keep the water level between 0 and 9 and then write a hybrid program that models the whole system.

Solution: In our controller design, we decided that we only wanted to switch the pipe to `on` if the water level fell below a height of 1 inch, and that we would only switch the pipe to `off` if the water level rose above a height of 8 inches. When the water level was between 1 and 8 inches, we would not change the state of the input pipe. We modeled the human user as a non-deterministic choice without any guards, since the human could open or close the spigot under any circumstances.

$$\begin{aligned}&((?(h \leq 1); s := 1) \ \cup\ ?(1 < h \land h < 8) \ \cup\ (?(h \leq 8); s := 0);\\ &\ (s := s - 1) \ \cup\ (s := s - 0.1);\\ &\ \{h' = s \ \&\ 0 \leq h \leq 9\})^*\end{aligned}$$

At the end of recitation, it was pointed out that this model would require the user to control the spigot in the same instant of time as the controller, and that the assignment of s by the human user depended on the value of s assigned by our controller. These are undesirable properties!

To allow the controller and the human to interact with the spigot at different times, we could use non-deterministic choice ($\cup$) instead of sequential composition (;) to separate them. A full solution to this problem (not presented in recitation) could look like this:

$$\begin{array}{l}((?(h \leq 1); s := 1) \;\cup\; ?(1 < h \land h < 8) \;\cup\; (?(h \leq 8); s := 0) \\ \;\cup (d := -1) \;\cup\; (d := -0.1); \\ \;\{h' = s + d \;\&\; 0 \leq h \leq 9\})^*\end{array}$$