Integrability in Nonstandard Modeling of Hybrid Systems

Kengo Kido, The University of Tokyo. <k-kido@is.s.u-tokyo.ac.jp>

Nonstandard Modeling of Hybrid Systems

WHILE$^{\mathtt{dt}}$ is a modeling language for hybrid systems introduced in [Suenaga & Hasuo, ICALP'11]. It is an extension of a usual imperative language, with a constant dt that represents an *infinitesimal*—a positive value that is smaller than any positive real.

In WHILE$^{\mathtt{dt}}$, we model continuous flow in hybrid systems as if it were infinitely many infinitesimal jumps, dispensing with explicit use of differential equations. An example is shown above, where t is understood to grow from 0 to 1 in a continuous manner. The usual Hoare-style program logic is just as valid in this extension, leading to an automatic precondition generator in [Hasuo & Suenaga, CAV'12].

$$t := 0;\ \mathtt{while}\ (t \le 1)\ \ t := t + \mathtt{dt} \qquad (\dagger)$$

Nonstandard Analysis and Semantics of WHILE$^{\mathtt{dt}}$

The formal semantics of WHILE$^{\mathtt{dt}}$ is given using Robinson's *nonstandard analysis (NSA)*, a framework that supports use of infinitesimals in a mathematically rigorous manner. There *hyperreals*—including (standard) reals, infinitesimals, and infinites as their multiplicative inverses—are (equivalence classes of) infinite sequences of real numbers. For example, the hyperreal $\omega^{-1} = \big[(\mathbf{1}, \frac{\mathbf{1}}{\mathbf{2}}, \frac{\mathbf{1}}{\mathbf{3}}, \dots)\big]$ is infinitesimal and is used as the denotation $[\![\mathtt{dt}]\!]$ of the constant dt.

The semantics of the program ($\dagger$) is then defined as follows. We consider the *i-th section* of the execution of the program, for each $i \in \mathbb{N}$:

$$\boxed{t := 0;\ \mathtt{while}\ (t \le 1)\ t := t + \mathbf{1}}\ \boxed{t := 0;\ \mathtt{while}\ (t \le 1)\ t := t + \tfrac{\mathbf{1}}{\mathbf{2}}} \cdots \boxed{t := 0;\ \mathtt{while}\ (t \le 1)\ t := t + \tfrac{\mathbf{1}}{\mathbf{i+1}}} \cdots \quad (*)$$

Each section is dt-free and its semantics is obvious. The values of t are then bundled up and we obtain $\big[(1+1, 1+\frac{1}{2}, 1+\frac{1}{3}, \dots)\big]$. This is a hyperreal that is infinitely close to 1.

The Integrability Problem

In the current definition the denotation $[\![\mathtt{dt}]\!]$ is fixed to be a specific infinitesimal ω^{-1}. This choice, however, is arbitrary: we expect the behavior of a WHILE$^{\mathtt{dt}}$ program to be independent from the choice of $[\![\mathtt{dt}]\!]$—at least if the program is modeling a "realistic" physical system. That is, we ask if a program satisfies the following.

Definition (Integrability). A WHILE$^{\mathtt{dt}}$ program P is *integrable* if, for any positive infinitesimals ∂_1 and ∂_2 and any memory state σ, we have $[\![P]\!]^{\partial_1}\sigma \simeq [\![P]\!]^{\partial_2}\sigma$. Here $[\![P]\!]^{\partial}$ is the (state transformer) semantics of P when $[\![\mathtt{dt}]\!] = \partial$; and $\simeq$ denotes that all the stored values are infinitely close.

The name comes from the notion of *Riemann integrability* in analysis, where any progressive sequence of partitions is required to lead to the same Riemann sum.

Unfortunately, not every WHILE$^{\mathtt{dt}}$ program is integrable. An artificial example is on the right: if $[\![\mathtt{dt}]\!] = \omega^{-1}$ the program terminates (since every section does—see $(*)$); however if $[\![\mathtt{dt}]\!] = \big[(\frac{\mathbf{1}}{\pi}, \frac{\mathbf{1}}{\mathbf{2}\pi}, \frac{\mathbf{1}}{\mathbf{3}\pi}, \dots)\big]$ it does not. Worse, the following naive modeling of a (seemingly benign) billiard ball bouncing between two walls turns out to be nonintegrable.

$$t := 0;\ \mathtt{while}\ (t \ne 1)\ \ t := t + \mathtt{dt}$$

```
x := 0.5;  v := 1;  t := 0;
while (t ≤ 10) do {
    if (x < 0 ∨ x > 1)
        then v := −0.8 ∗ v;
    x := x + v ∗ dt;    t := t + dt   }
```

Indeed, while the program behaves in the way we intend under $[\![\mathtt{dt}]\!] = [(1, \frac{1}{3}, \frac{1}{5}, \dots)]$, under $[\![\mathtt{dt}]\!] = [(\frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \dots)]$ the ball gets caught in the wall after the first bounce and stays there since x does not get to ≤ 1.

For this specific example, we can make it integrable by the following two simple modifications: 1) change the guard $x < 0 \lor x > 1$ of the if branch into $(x < 0 \land v \le 0) \lor (x > 1 \land v \ge 0)$; or 2) add $x := 1$ to the then clause.

Towards a Proof Method for Integrability

Hence we aim at a generic methodology for establishing integrability of WHILE$^{\mathtt{dt}}$ programs. The project is in a very early stage and we will very much appreciate your suggestions.

One possible direction we are looking at is the relationship to the *non-interference* property that concerns security of programs. It states that there is no "information leak" from high-security variables to low-security ones.

Definition (Non-interference). Let $V = V_h \oplus V_l$ be a partition of variables into *high-security* and *low-security* ones. A program P satisfies *non-interference* if, for any states σ_1, σ_2 such that $\sigma_1|_{V_l} = \sigma_2|_{V_l}$, we have $([\![P]\!]\sigma_1)|_{V_l} = ([\![P]\!]\sigma_2)|_{V_l}$.

Non-interference is similar to integrability if we see dt as a high-security variable. Therefore we suspect that proof methods for non-interference like [Terauchi & Aiken, SAS'05] and [Sabelfeld & Sands, CSF'00] can be applied.