DD2452 Formal Methods
Lab 1: Deductive Verification of an ABS Controller

Jonas Haglund

September 20, 2018

1 Introduction

In this lab, you will use the WP plugin of Frama-C to formally verify a C program controlling an anti-lock braking system (ABS) of a car. ABS is a safety system used in cars, motorcycles, trucks and airplanes to prevent the wheels from locking under braking, which would otherwise make the vehicle difficult to control. The purpose of ABS is to optimize braking distance and retain steerability and stability of the vehicle.

Section 5 explains your tasks. Sections 2 and 3 explain the algorithm the C program implements and are not necessary to read. Section 4 gives a brief overview of the structure of the C program (which contains explaining comments) and might be good to read.

2 Organization and Workings of an ABS system

The braking system in a car is organized roughly as a chain as follows:

1. The braking pedal is connected to the master cylinder.
2. The master cylinder is connected to the hydraulic modulator.
3. The hydraulic modulator is connected to the brake circuits (there are two brake circuits for safety reasons if one would fail: one for the front wheels and one for the rear wheels).
4. The brake circuits are connected to the brake calipers.

The master cylinder converts the mechanical force on the braking pedal to a proportional hydraulic force, by forcing brake fluid into the hydraulic modulator.

The hydraulic modulator increases, holds or decreases the brake pressure by controlling the amount of brake fluid in the brake circuits by opening and closing the inlet and outlet valves. When the inlet valve is open/closed, brake fluid is passed/not passed into the brake circuit. When the outlet valve is open/closed, brake fuild is pumped/not pumped out of the brake circuit into the master cylinder reservior. Hence, the hydraulic modulator controls the brake pressure as follows:

- Increasing brake pressure: The inlet vavle is open and the outlet vavle is closed. This results in the driver controlling the brake pressure.
- Holding brake pressure: Both valves are closed, resulting in constant brake pressure, irrespectively of how hard the driver pushes the braking pedal.
- Decreasing brake pressure: The inlet vavle is closed and the outlet vavle is open, resulting in the braking pedal having no influence on the braking of the car.

The brake fluid in the brake circuits causes a hydraulic force to press the brake pads in the calipers against the brake disk.

The ABS mechanism is implemented by the electronic control unit (ECU). In order for the ECU to control the brake pressure, the ECU is connected to the hydraulic modulator and continously sends a control signal to the hydraulic modulator of what the state of the valves shall be. To compute the control signal the ECU makes use of the following three sensors:

- Braking pedal sensor: Records the mechanical force on (or the displacement of) the braking pedal.
- Wheel angular velocity sensors: Records the rotational velocity of the wheels.
- Vehicle acceleration sensor: Records the change in velocity of the vehicle.

The ECU minimizes the braking distance by means of a concept called wheel slip ratio, defined as follows. A wheel is freely rolling when no braking nor driving torque is applied to the wheel. When a wheel is freely rolling its angular velocity is $\omega_0 = v/R$ where v is the velocity of the car and R is the radius of the wheels. When a braking torque is applied to a wheel, the wheel starts to slip and its angular velocity ω_t decreases: $\omega_t < \omega_0$. The wheel slip ratio $(\omega_0 - \omega_t)/\omega_0$ gives an indication of the amount of wheel slip. If the wheel is locked, $\omega_t = 0$, then the wheel slip ratio is at maximum one. If the wheel is slipping just a bit, $\omega_t \approx \omega_0$ and the wheel slip ratio is close to zero. The relationship between wheel slip ratio and the optimal braking distance depends on the velocity of the vehicle and surface conditions (wet or dry). Usually the optimal wheel slip ratio S_{ref} ranges between 0.10 and 0.60.

3 An Intelligent ABS Algorithm

The goal of the ABS controller in the ECU is, under braking, to keep the actual wheel slip $S = (\omega_0 - \omega_t)/\omega_0$ as close to S_{ref} as possible (where $S_{ref} \in [0.10, 0.60]$ depending on surface conditions and velocity). For simplification the ABS algorithm controls the braking of only one wheel. S_{ref} is defined to be 0.15: $S_{ref} = 0.15$.

Input to the algorithm is:

- Braking pedal pushing indicator: Instead of the braking pedal sensor mentioned above that records mechanical force on or displacement of the braking pedal, a "braking pedal pushing indicator" is used which returns 1 if the braking pedal is pushed and 0 otherwise.

- Wheel angular velocity sensor: Records the rotational speed ω_t of the wheel in radians/s.
- Vehicle acceleration sensor: Records the acceleration av of the car in m/s^2.

Output of the algorithm is a control signal u_c to the hydraulic modulator. The hydraulic modulator reacts to the control signal u_c as follows:

- $u_c > 0.15$: The hydraulic modulator increases the brake pressure by opening the inlet valve and closing the outlet valve.
- $u_c < -0.15$: The hydraulic modulator decreases the brake pressure by closing the inlet valve and opening the outlet valve.
- Otherwise: The states of the inlet and outlet valves are unchanged.

It can be assumed that the ABS algorithm is invoked once every 20 ms (a timer can be configured to raise one interrupt every 20 ms, and at each interrupt the ABS algorithm is invoked). The ABS algorithm considered in this lab is a combination of PID (Proportional, Integral, Derivative) and intelligent control.

PID is a common control method based on calculating the error of previous control signals (in this context, the difference between the actual wheel slip ratio S and the optimal wheel slip ratio S_{ref}). The proportional, integral and derivative components are used, respectively, to proportionally correct the control signal, correct accumulated errors over time, and correct the current error.

The intelligent control part is based on fuzzy logic, a successful control method suitable for handling non-linear behaviors, braking in this context. Fuzzy logic works basically as follows. Given inputs $i = (i_1, ..., i_n)$, a set of rules is applied on i. Each rule R_m has the form: **if** $P_m(i)$ **then** $\mu_m(i)$. The meaning of such a rule is: if $P_m(i)$ is true, then compute the membership function μ_m applied on i: $\mu_m(i)$. $0 \leq \mu_m(i) \leq 1$ gives an indication of the degree to which i is in a certain set. $\mu_m(i) = 0$ means that i is not in the set, $\mu_m(i) = 1$ means that i is definitively in the set, and $0 < \mu_m(i) < 0$ means that i is in the set to a certain degree. In this context $i = (e, e')$, where $e = S - S_{ref}$ is the error and e' is rate of change of the error, and one of the sets considered is whether e and e' are zero. Those $\mu_m(i)$ that have been computed, depending on whether the corresponding $P_m(i)$ are true, are then used to compute the final output (the control signal u_c in this context).

The PID part of the algorithm computes $e = S - S_{ref}$ and $e' = (S - S_{previous})/\Delta t$, where S is the most recent computation of the actual wheel slip, $S_{previous}$ is the wheel slip computed by the previous invocation of the algorithm, and $\Delta t = 0.020$ s. Recall $\omega_0 = v/R$ (where v is the speed

of the vehicle and R is the radius of the wheels) and $S = (\omega_0 - \omega_t)/\omega_0$. These two equations give $S = (v/R - \omega_t)/(v/R)$. Multiplying the right side by $R/R = 1$ gives $S = (v - \omega_t R)/v$. The velocity of the vehicle can be computed by integrating over the acceleration a over time and adding the velocity $v_0 = \omega_t R$ of the vehicle when the braking started (at which time ω_t is the angular velocity of a freely rolling wheel): $v = \int a \; dt + v_0$.

The fuzzy logic part of the algorithm computes the final control signal u_c to the hydraulic modulator by means of e, e', and the following set of rules:

$$\mu_{NB}(x) = \begin{cases} 1 & \text{if } x \leq -1 \\ -2x - 1 & \text{if } -1 < x < -0.5 \\ 0 & \text{if } -0.5 \leq x \end{cases}$$

$$\mu_{NM}(x) = \begin{cases} 0 & \text{if } x \leq -1 \\ 2x + 2 & \text{if } -1 < x \leq -0.5 \\ -4x - 1 & \text{if } -0.5 < x < -0.25 \\ 0 & \text{if } -0.25 \leq x \end{cases}$$

$$\mu_{NS}(x) = \begin{cases} 0 & \text{if } x \leq -0.5 \\ 4x + 2 & \text{if } -0.5 < x \leq -0.25 \\ -4x & \text{if } -0.25 < x < 0 \\ 0 & \text{if } 0 \leq x \end{cases}$$

$$\mu_{ZE}(x) = \begin{cases} 0 & \text{if } x \leq -0.25 \\ 4x + 1 & \text{if } -0.25 < x \leq 0 \\ -4x + 1 & \text{if } 0 < x < 0.25 \\ 0 & \text{if } 0.25 \leq x \end{cases}$$

$$\mu_{PS}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 4x & \text{if } 0 < x \leq 0.25 \\ -4x + 2 & \text{if } 0.25 < x < 0.5 \\ 0 & \text{if } 0.5 \leq x \end{cases}$$

$$\mu_{PM}(x) = \begin{cases} 0 & \text{if } x \leq 0.25 \\ 4x - 1 & \text{if } 0.25 < x \leq 0.5 \\ -2x + 2 & \text{if } 0.5 < x < 1 \\ 0 & \text{if } 1 \leq x \end{cases}$$

$$\mu_{PB}(x) = \begin{cases} 0 & \text{if } x \leq 0.5 \\ 2x - 1 & \text{if } 0.5 < x < 1 \\ 1 & \text{if } 1 \leq x \end{cases}$$

where x is either e or e'. The membership functions have a triangular shape and state the degree to which e and e' are considered to be very negative (negative-big, NB), moderately negative (negative-medium, NM), slightly negative (negative-smallm, NS), zero (ZE), slightly positive (positive-small, PS), moderately positive (positive-medium, PM), and very positive (positive-big, PB).

The membership functions give a weight of what value the control signal u_c should have for the sets NB, NM, NS, ZE, PS, PM and PB. The control signal u_c to the hydraulic modulator is then calculated by means of the product-sum inference method:

$$u_c = \frac{\sum_{i,j} \mu_i(e) \cdot \mu_j(e') \cdot u_{i,j}}{\sum_{i,j} \mu_i(e) \cdot \mu_j(e')},$$

where i and j ranges over the set {NB, NM, NS, ZE, PS, PM, PB}, and $u_{i,j}$ is the control signal that is appropriate when for i and j. $u_{i,j}$ is derived as follows (recall $S = (v - \omega_t R)/v$, $e = S - S_{ref}$, and $e' = (S - S_{previous})/\Delta t$). Consider the desired behavior of the brakes for the possible ranges that e and e' can be in:

- NB:
  - e is very negative: This means that the wheel slip is far below optimal and that the wheel velocity is far too high. The wheel slip is increased by greatly increasing braking.
  - e' is very negative: This means that the wheel slip is decreasing fast, either towards or away from the optimum S_{ref}.
- NM:
  - e is moderately negative: The wheel slip is significantly below optimal, and the wheel velocity is too high. The wheel slip is increased by increasing the braking.
  - e' is moderately negative: The wheel slip is decreasing at a moderate rate, either towards or away from the optimum.
- NS:
  - e is slightly negative: The wheel slip is slightly below optimal, meaning that the wheel velocity is a bit to high. The wheel slip is increased by lightly increasing the braking.
  - e' is slightly negative: The wheel slip is decreasing slowly towards or away from the optimum.
- ZE:

  - e is zero: The wheel slip is optimal. No change in braking is needed.
  - e' is zero: The wheel slip is constant.

- PS:
  - e is slightly positive: The wheel slip is slightly above optimal, meaning that the wheel velocity is a bit too low. The wheel slip is decreased by lightly decreasing the braking.
  - e' is slightly positive: The wheel slip is increasing slowly towards or away from optimum.

- PM:
  - e is moderately positive: The wheel slip is significantly above the optimum with the wheel velocity being too low. The wheel slip is decreased by decreasing the braking.
  - e' is moderately positive: The wheel slip is increasing at a moderate rate towards or away from optimum.

- PB:
  - e is very positive: The wheel slip is far above optimum with wheel velocity being far too low. The wheel slip is greatly reduced by no braking.
  - e' is very positive: The wheel slip is increasing fast towards or away from optimum.

The values of the control signal u_c are in the interval [-1, 1] and can be considered to have the following meanings when interpreted by the hydraulic modulator:

- $u_c = 1$: Increase brake pressure as much as possible.
- $u_c = 2/3$: Increase brake pressure moderately.
- $u_c = 1/3$: Increase brake pressure by a small amount.
- $u_c = 0$: No change.
- $u_c = -1/3$: Decrease brake pressure by a small amount.
- $u_c = -2/3$: Decrease brake pressure moderately.
- $u_c = -1$: Decrease brake pressure as much as possible.

With the desired behavior of the brakes depending on the values of e and e' and the interpretation of u_c, $u_{i,j}$ is defined as follows:

$u_{e,e'}$		e'						
		NB	NM	NS	ZE	PS	PM	PB
e	NB	1	1	1	1	2/3	1/3	0
	NM	1	1	1	2/3	2/3	0	-1/3
	NS	1	2/3	2/3	1/3	0	-1/3	-2/3
	ZE	1	2/3	1/3	0	-1/3	-2/3	-1
	PS	2/3	1/3	0	-1/3	-2/3	-2/3	-1
	PM	1/3	0	-2/3	-2/3	-1	-1	-1
	PB	0	-1/3	-2/3	-1	-1	-1	-1

For instance, when $e \in$ NM and $e' \in$ PS, then the appropriate control signal is 2/3. That is, the brake pressure should increase by a moderate amount. Also, for simplification, the verification does not need to deal with arithmetic overflows.

4 Structure of the C Program to Verify

You are given a C program, called tabs.c, implementing the algorithm explained in sections 2 and 3. The most relevant variables and functions of tabs.c are:

- signal_to_hydraulic_modulator: Dummy variable representing the address of the register that causes the ECU to send the written value to the hydraulic modulator.
- wt_sensor: Dummy variable representing the address of the register used to read the angular wheel velocity sensor.
- bp_sensor: Dummy variable representing the address of the register used to read whether the brake pedal is pushed.
- at_sensor: Dummy variable representing the address of the register used to read the acceleration sensor.
- R: Global constant storing the radius of the wheels.
- delta_t: Stores the number of milliseconds between interrupts and invocation of the ABS software.
- acceleration_sum: Global variable accumulating the acceleration samples. Used to compute the velocity of the vehicle by integration.
- velocity_before_braking: Global variable storing the velocity of the vehicle just before braking.
- md(index, x): Function implementing the membership functions used to compute the degree to which x is in the set represented by index

(index = 0 = NB, index = 1 = NM, index = 2 = NS, index = 3 = ZE, index = 4 = PS, index = 5 = PM, index = 6 = PB).

- compute_velocity_of_vehicle(): Function computing the velocity of the vehicle.
- compute_wheel_slip(v, wt): Function computing the wheel slip given the current velocity v and angular wheel velocity wt.
- compute_control_signal(): Function computing the control signal that shall be sent to the hydraulic modulator.
- hydraulic_modulator_driver(): The function the timer interrupt routine calls to update the brake pressure.

5 Tasks

Verifying programs with floating point computations with Frama-C requires interactive theorem proving, which we will not do. Therefore all units are multiplied by 1000 and only integer computations are performed.

Your task is to specify formally and verify in Frama-C with the WP plugin the following properties:

- If the brake pedal is not pushed, then the brakes are not applied ($u_c = -1000 = -1 \cdot 1000$, meaning that the outlet valve is open). That is, -1000 is written to the variable signal_to_hydraulic_modulator:

$$\text{signal\_to\_hydraulic\_modulator} = -1000.$$

- If the brake pedal is not pushed, then the current velocity is correctly stored in the variable velocity_before_braking:

$$\text{velocity\_before\_braking} = \text{wt\_sensor} \cdot \text{R}.$$

- If the brake pedal is not pushed, then the current acceleration is correctly stored in the variable acceleration_sum:

$$\text{acceleration\_sum} = \text{at\_sensor}.$$

- If the brake pedal is not pushed, then S_previous is assigned zero:

$$\text{S\_previous} = 0.$$

- The velocity is computed correctly (the first term is divided by 1000 since both factors of that term have already been multiplied by 1000):

$$\text{compute\_velocity\_of\_vehicle}() =$$
$$\text{acceleration\_sum} \cdot \text{delta\_t}/1000 + \text{velocity\_before\_braking}.$$

- The wheel slip is computed correctly:

$$\text{compute\_wheel\_slip(v, wt)} = (\text{v - wt} \cdot \text{R}/1000)/\text{v}.$$

- If the following three conditions hold:
  - the brake pedal is pushed,
  - the wheel slip is below optimal by 250 ($S - S_{ref} = e \leq -250$), and
  - the wheel slip is decreasing by 250 units per time interval ($(S - S_{previous})/\Delta t = e' \leq -250$),

  then brake pressure is increased ($u_c > 150 = 1000 \cdot 0.15$).

  That is, a value greater than 150 is assigned to signal_to_hydraulic_modulator:

$$\text{signal\_to\_hydraulic\_modulator} > 150.$$

In some function contracts the values of the array u must be stated (assumed in a requires clause).

Your solution shall be tabs.c annotated with ACSL annotations such that Frama-C verifies the properties listed above (ACSL is an annotation language of which Frama-C implements a subset). The C code in tabs.c must not be modified. In addition, write a succinct report containing at least the following:

1. Your thoughts about the difficulty of the lab.
2. Annotation overhead (number of lines and number of goals to be proved).
3. How did you familiarize yourself with the code? Testing? Reading?
4. Your logical verification approach (your reasoning before you annotated the code).
5. How did you annotate the code? Did you rewrite the code into a simpler equivalent form and annotating that form, and then working your way back to the original code by making small stepwise modifications of the simpler code and its associated annotations?
6. Verification time (in seconds).
7. Is there any code in tabs.c that you would rewrite in order to ease the verification?
8. Will you write code differently in the future? Why?

9. What was most difficult?

10. Are there any properties that you have not verified that you think are relevant to verify?

11. Do you see any flaws in this verification approach where only source code is analyzed?

12. Do you see any practical limitations of what programs/properties that can be verified with Frama-C and the WP plugin?

13. Do you see any shortcomings in the WP-plugin with respect to the expressiveness of annotations?

14. Were there any annotations that you expected to be proved/not to be proved but was not proved/proved in Frama-C? In such a case, what was the assertion?

15. Have you experienced any strange behavior of (or bugs in) Frama-C? For instance, were you forced to add a statement annotation that was included in a requires clause?

16. The annotated code in an appendix.

These are the requirements for the grade E. For a D, the report shall be well-written and well-structured. For a C, the requirements for E and D must be fulfilled, and in addition the properties verified for tabs.c must be verified for tabs_loop.c. The difference between tabs_loop.c and tabs.c is that in the function compute_control_signal, some for loops in tabs_loop.c replace explicit additions in tabs.c.