AUFGABE 7: ABSTRAKTE INTERPRETATION & VERIFIKATION

In dieser Aufgabe werden Sie zunächst einen Sensorstub implementieren, der mit einem vorgegebenen Ringpuffer zur Zwischenspeicherung der Messergebnisse interagiert. Im Anschluss sollen die Abwesenheit von Laufzeitfehlern in beiden Komponenten mittels Astrée und FRAMA-C nachgewiesen werden. Ferner wird ebenfalls mittels Astrée die numerische Stabilität eines digitalen α-β-Filter bewiesen.

Grundlegende Übungen

Aufgabe 1 *Einführende Frage*

Was ist der Unterschied zwischen den Astrée-Konzepten der *Known Facts* und der *Assertions*? Wann sollte welches dieser beiden Konzepte zum Einsatz kommen?
Antwort:

Sensor

Aufgabe 2 *Sensor-Stub*

Implementieren Sie einen Stub für einen Sensor. Dieser soll bei jedem Aufruf einen simulierten Messwert vom Datentyp `float` liefern. Sorgen Sie dabei dafür, dass für Astrée klar ist, dass es von den auf den Übungsfolien angegebenen einschränkenden Annahmen bezüglich des konkreten Messwerts ausgehen darf.

Filter (I): Initialisierung

Aufgabe 3 *Filterstabilität*

In der Datei `src/ab_filter.c` findet sich eine Implementierung des in der Übung vorgestellten α-β-Filter für den Datentyp float. Machen Sie sich zunächst mit der Implementierung vertraut.

Welche Nachbedingung muss für die Filterinitialisierung gelten, damit das Filtern stabil ist?

Antwort:

Aus den Erfahrungen mit dem I4Copter ist bekannt, dass sich die Filterparameter in den folgenden Bereichen bewegen:

Abtastintervall $T \in (0 \dots 1]$
Prozessvarianz $\sigma_w^2 \in [0.5 \dots 30.0]$
Rauschvarianz $\sigma_v^2 \in \left[10^{-3} \dots 10^{-1}\right]$
Wert $y\,[\kappa] \in [-10 \dots 10]$

Vergewissern Sie sich durch händische Intervallrechnung, inwiefern sich aus diesen Parametern durch die in der Übung beschriebene und hier umgesetzte Filterinitialisierung stabile Werte für α und β ergeben.
Antwort:

Welcher Wertebereich ergibt sich daraus für den Parameter λ?
Antwort:

Annotieren Sie die obigen Wertebereiche für die Filterparameter sowie die Nachbedingungen für stabile Filterparameter α und β in der Filterinitialisierung für Astrée.

Weisen sie die Gültigkeit dieser Wertebereiche und damit die Filterstabilität mittels Astrée nach. Wo müssen Sie die Analyse durch zusätzliches Wissen unterstützen? Geben Sie sich an den jeweiligen Stellen den Analysezustand relevanter Variablen aus.
Antwort:

☞ __ASTREE_known_*
☞ __ASTREE_log_vars

Ringpuffer

Moderne Mikrocontroller stellen häufig eine DMA-Einheit zur Verfügung, die Peripheriedaten *asynchron* zum Prozessor beschaffen kann. Deswegen sollen in dieser Aufgabe die von der Sensorhardware gelieferten Daten in Schüben verarbeitet werden. Hierfür stellen wir Ihnen eine Implementierung eines Ringpuffers bereit.

☞ `bndbuf.c`, `bndbuf.h`

Aufgabe 4 *Laufzeitfehler*

In den Dateien `include/bndbuf.h` und `src/bndbuf.c` finden Sie eine Implementierung eines Ringpuffers. Machen Sie sich zunächst mit der vorgegebenen Schnittstelle des Ringpuffers und der Ansteuerung in der Funktion `main` in der Datei `src/main.c` vertraut. Weisen Sie nun die Abwesenheit von Laufzeitfehlern wie etwa Speicherzugriffsfehler in der Ringpuffer-Implementierung mittels Astrée nach. Wieso scheitert Astree initial, sinnvolle Grenzen für relevante Variablen festzulegen? Wie können sie Astree dabei unterstützen, diese Werte sinnvoll nachzuverfolgen? *Für welchen Kontext gelten Astrées Korrektheitsgarantieen?* Diese Teilaufgabe gilt als erfolgreich bearbeitet, wenn Astrée im Modul `bndbuf.c` keine Fehler mehr meldet.

Aufgabe 5 *Korrektheit*

Versuchen Sie nun mittels Astrée auch die funktionale Korrektheit der gegebenen Implementierung zu beweisen. Betrachten Sie dabei zunächst die folgende Eigenschaft:

- Die Füllstandsvariable `count` verlässt nie die Pufferkapazität.

Verwenden Sie zum Nachweis ausschließlich `assert`-Operationen.

Betrachten Sie nun die folgenden beiden Eigenschaften:

- Die Füllstandsvariable `count` entspricht der Differenz (unter Berücksichtigung der Umlaufsemantik an den Puffergrenzen) zwischen Schreib- und Lesezeiger.
- Falls der Puffer nicht vollständig belegt ist, speichert `bnd_put` das Element tatsächlich im Puffer ab.

Hinweis: Diese beide Eigenschaften lassen sich nicht durch Astrée nachweisen. Wieso? Betrachten Sie dazu den Analysezustand an verschiedenen Stellen Ihrer Implementierung. Welche Zusammenhänge müssten Sie ausdrücken, können diese aber nicht durch Annotationen formulieren? Wie sind die relevanten Informationen im Analysezustand modelliert?

☞ `__ASTREE_analysis_log`

☞ `__ASTREE_log_vars`

Antwort:

Aufgabe 6 FRAMA-C

Weisen Sie nun mittels FRAMA-C die oben betrachteten Aspekte der funktionalen Korrektheit der gegebenen Implementierung nach.

Setzen Sie sich hierzu zunächst mittels **source** `framacenv.sh` (im Quellverzeichnis des Aufgabenordners) die passende Umgebung für FRAMA-C und dessen Abhängigkeiten auf. Danach können Sie mittels `make frama-c-gui` die graphische Oberfläche mit den empfohlenen Parametern starten, die Befehlszeilenvariante von FRAMA-C starten Sie mittels `make frama-c`.

Betrachten Sie dabei also die folgenden Eigenschaften:

- Die Füllstandsvariable `count` verlässt nie die Pufferkapazität.
- Die Füllstandsvariable `count` entspricht der Differenz (unter Berücksichtigung der Umlaufsemantik an den Puffergrenzen) zwischen Schreib- und Lesezeiger.
- Falls der Puffer nicht vollständig belegt ist, speichert `bnd_put` das Element tatsächlich im Puffer ab.

Gehen Sie dabei wie folgt vor: Definieren Sie zunächst die durch uns vorgegebenen Prädikate und definieren Sie zunächst die Vor- und Nachbedingungen der Methoden im Modul `src/bndbuf.c`, welche für die Abwesenheit von Laufzeitfehlern erforderlich sind. Erweitern Sie dann die Nachbedingungen sukzessive um die oben geforderten Eigenschaften bzw. die mit ihnen verknüpften Prädikate und axiomatisieren Sie die Funktionen entsprechend.

Wie unterscheidet sich der Verifikationsansatz mittels FRAMA-C von der Verifikation mittels Astrée?

Antwort:

Erweiterte Übung

Filter (II): Filterverwendung

Aufgabe 7 *Filterschritt*

Die Funktion `ab_process` führt einen einzelnen Filterschritt aus und wählt dabei zwischen Varianten für den eingeschwungenen Zustand und die Filtererholung aus. Die Filterverarbeitung wird in der Datei `src/main.c` durch die `#define EXTENDED` -Direktive aktiviert. Entfernen Sie dazu den Kommentar vor der Direktive.

Betrachten Sie zunächst die Implementierung für den eingeschwungenen Zustand.
Was kann in jedem Filterschritt schiefgehen?
Antwort:

Wie können Sie erkennen, dass das Filter in einen unbrauchbaren Zustand übergeht?
Antwort:

Treffen Sie bei der Implementierung Maßnahmen, die das Filter in kontrollierter Art und Weise wieder in einen brauchbaren Zustand überführen!

Betrachten Sie dazu zunächst das Filter für den eingeschwungenen Zustand und fangen Sie nun die Fälle ab, in denen das Filter im laufenden Betrieb in einen unbrauchbaren Zustand übergehen würde. *Finden Sie die relevanten Stellen, bspw. mittels Astrée und schalten Sie in diesem Fall auf das ebenfalls implementierte Erholungsverfahren um.* Auch während der Erholungsphase müssen die Filterparameter α_n und β_n sich im stabilen Bereich befinden. Weisen Sie auch dies mit Hilfe von Astrée nach.

☞ `ab_step`

☞ `ab_step_transient`

Hinweis: Achten Sie darauf, dass Sie die Korrektheit ihrer Implementierung nicht nur für bestimmte Filterparametersätze nachweisen, sondern für den gesamten Definitionsbereich! Diese Teilaufgabe gilt als erfolgreich bearbeitet, wenn alle Vor- und Nachbedingungen annotiert sind und Astrée in den relevanten Funktionen keine Fehler mehr meldet.

Aufgabe 8 *Fortgeschrittene Axiomatisierung von Datenstrukturen*

Im Modul `src/bndbuf.c` finden sich die Funktionen `framac_axiom_new_push_pop` und `framac_axiom_push_eventual_pop`. Diese Funktionen werden durch die #

`define EXTENDED`-Direktive aktiviert. Entfernen Sie dazu den Kommentar vor der Direktive.
Welche Eigenschaft wird durch den erfolgreichen Beweis für die Ringpufferimplementierung jeweils nachgewiesen? Welche Aussage ist stärker und damit aussagekräftiger?
Antwort:

Beweisen Sie nun die Gültigkeit der beiden als Funktionen formulierten Aussagen (d.h. die relevanten `assert`- und `ensures`-Klauseln), indem Sie die beiden Funktionen sowie die Beschreibung der aufgerufenen Ringpuffermethoden erweitern und verfeinern. ☞ behaviors

Aufgabe 9 *Ausblick*

Die beiden Funktionen `framac_axiom_new_push_pop` und `framac_axiom_push_eventual_pop` betrachten bestimmte Operationsfolgen auf dem Ringpuffer, welche jeweils bestimmte Aussagen zulassen. Die zugrundeliegende Aussage ist die schwächere Form einer allgemeinen Eigenschaft des Ringpuffers. Wie könnte die Funktion erweitert werden, um diese Aussage allgemeingültig zu beweisen?
Hinweis: Ein Beweis dieser stärkeren Eigenschaft ist **nicht** erforderlich.
Antwort:

Hinweise

- Bearbeitung: Gruppenarbeit
- Abgabefrist: 24.01.2025
- Fragen bitte an `i4ezs@lists.cs.fau.de`