Zweites Lab zur Vorlesung „Formale Methoden im Software Entwurf“

TECHNISCHE UNIVERSITÄT DARMSTADT

Verifikation mit LTL und erste Schritte in FOL

Formale Anforderungen an Ihre Abgabe

Abgabefrist

Abgabedeadline des Labs: Mo, 8.7.2024, 8:00

Abgabeort: https://moodle.informatik.tu-darmstadt.de/mod/assign/view.php?id=68298

Besprechung der Lösung des Labs: Mo, 8.7.2024, 8:55

Sie können Ihre Abgabe bis zur Abgabefrist beliebig häufig aktualisieren. Die zuletzt fristgerecht hochgeladene Version wird bewertet. Es reicht, wenn eine Person Ihrer Labgruppe die Lösung hochlädt.

Bitte prüfen Sie, dass Ihre Abgabe aktuell und vollständig ist. Verspätete Abgaben können nicht akzeptiert werden, da die Lösung des Labs direkt nach der Abgabefrist besprochen wird.

Abgabeformat

Die Lösung ist als ein einziges zip-Archiv mit dem Namen: `Lab2_GruppeXYZ.zip` abzugeben, wobei Sie bitte `XYZ` mit Ihrer Labgruppennummer ersetzen.

Das Archiv muss die Unterverzeichnisse `aufgabe1`, `aufgabe2`, `aufgabe3`, `aufgabe4` und `aufgabe5` mit den Lösungsdateien der zugehörigen Aufgaben enthalten. Die Dateien müssen alle Informationen für die jeweilige Lösung beinhalten.

Sie dürfen die Lösungen der Aufgaben 3, 4 und 5a) zusammen in einer einzigen PDF-Datei abgeben. Die Dateien für die Aufgaben 1, 2 und 5b) sind trotzdem notwendig und dürfen nicht in diesem PDF sein.

Nicht-Einhaltung der Ordnerstruktur oder das Fehlen von Dateien für Aufgaben, kann zu Punktabzügen führen.

Die abzugebenden PROMELA-Modelle dürfen keine Syntaxfehler beinhalten. KeY-Dateien (inklusive `.proof` Dateien) müssen mit der KeY-Version dieser Vorlesung ladbar sein.

Weitere Bearbeitungshinweise

- Sie können maximal **60 Punkte** erreichen.
- Sehen Sie sich die Ergebnisse von Spin im Detail an und akzeptieren Sie nicht unkritisch die gegebene Antwort.
- LTL Formeln können im Modell wie folgt definiert und benannt werden: ltl NAME {LTL_FORMEL} mit NAME und LTL_FORMEL als Platzhalter für die entsprechenden Elemente.
- Prüfen Sie bei der Verifikation, dass Spin den gesamten Zustandsraum durchsucht hat und nicht vorher mit einem Fehler abgebrochen hat. In diesem Fall ändern Sie bitte die Flags/Einstellungen in SPIN.

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Informationen finden Sie unter: https://www.informatik.tu-darmstadt.de/studium_fb20/im_studium/studienbuero/plagiarismus/index.de.jsp

Aufgabe 1: Keksautomat II (11 Punkte)

Im letzten Lab gab es eine Aufgabe zur Modellierung eines Keksautomaten. Nun ist das Modell vorgegeben in `aufgabe1/petrinet.pml`. Ihre Aufgabe besteht darin, weitere Eigenschaften zu verifizieren. Ändern Sie dabei das Modell nicht: Die Variable **short** places[PLACES_COUNT] sowie die Geistervariable **byte** transition reichen für die Formalisierung der Eigenschaften aus.

1a) (3 Punkte)

Verifizieren Sie folgende Eigenschaft mit LTL und geben Sie das Ergebnis als `aufgabe1/petrinet_a.pml` ab:

> Wenn niemals Geld eingeworfen wird, dann kann auch kein Keks gegessen werden.

Ihre Abgabedatei muss die verifizierte LTL Formel enthalten.

1b) (3 Punkte)

Verifizieren Sie folgende Eigenschaft mit LTL und geben Sie das Ergebnis als `aufgabe1/petrinet_b.pml` ab:

> Wenn der Automat niemals Geld zurückgibt, dann wird irgendwann nur noch „Skip“ ausgeführt.

Ihre Abgabedatei muss die verifizierte LTL Formel enthalten.

1c) (5 Punkte)

Verifizieren Sie folgende Eigenschaft mit LTL und geben Sie das Ergebnis als `aufgabe1/petrinet_c.pml` ab:

> Wenn der Automat das eingeworfene Geld stets sofort zurückgibt, dann bleibt die interne Kasse immer leer.

Ihre Abgabedatei muss die verifizierte LTL Formel enthalten.

Aufgabe 2: Fairness unter zwei Servern (15 Punkte)

Gegeben ist das Promela Modell in `aufgabe2/services.pml` eines Clients, der Berechnungen mithilfe von Webservern ausführen soll. Im Modell gibt es einige Serverprozesse und einen Clientprozess. Der Client sendet Anfragen an den Service, woraufhin ein Server mit einem Ergebnis antwortet.

Was die Prozesse intern berechnen wurde für die Modellierung (wie oft üblich) wegabstrahiert, damit nur das Wesentliche für die Verifikation bleibt. Für die LTL-Verifikationsaufgaben müssen Sie möglicherweise Geistervariablen und/oder Labels im Modell hinzufügen, ansonsten sollte das Modell gleich bleiben (Ausnahme: Aufgabe 2c).

Eine vollständige Lösung muss folgende Dateien im Verzeichnis `aufgabe2/` beinhalten:

- `solution.txt`
- `services_a.pml`
- `services_b.pml`
- `services_c.pml`

Optional je nach Verifikationsergebnis:

- `services_a.pml.trail`
- `services_b.pml.trail`
- `services_c.pml.trail`

Hinweis: Wenn Sie in der LTL-Formel auf ein Label in einem bestimmten Prozess zugreifen möchten, benutzen Sie dafür proc[id]@label, wobei id für die Prozess _pid steht. Wenn Sie nicht wissen, welcher Prozess welche _pid besitzt, können Sie diese mithilfe eines **printf**-Befehls im Simulationsmodus herausfinden. Alternativ können Sie natürlich auch ganz auf Labels verzichten und Geistervariablen benutzen.

2a) Starvation (5 Punkte)

1. Benutzen Sie die vorgegebene Datei `aufgabe2/services.pml`.
2. Formulieren Sie folgende Eigenschaft mit LTL:

   Wenn immer der Client eine Anfrage sendet, dann erhält er auch irgendwann eine Antwort.

   Speichern Sie das Modell mit dem LTL-Claim unter: `aufgabe2/services_a.pml`
3. Falls die Verifikation nicht erfolgreich war, geben Sie den gefundenen Trail `aufgabe2/services_a.pml.trail` ab. Werfen Sie einen Blick auf den Trail und beantworten Sie (in `aufgabe2/solution.txt`) mit einer kurzen Begründung, ob der Scheduler im gefundenen Trail Weak-Fair und/oder Strong-Fair ist.
4. Falls die Verifikation erfolgreich war, nennen Sie die Fairness-Annahme bei der Verifikation.

2b) Fairness I (5 Punkte)

1. Starten Sie mit der vorgegebenen Datei `aufgabe2/services.pml`.
2. Formulieren Sie folgende Eigenschaft mit LTL:

   Wenn der Client unbegrenzt viele Anfragen sendet (also unendlich oft), dann wird jeder Server für die Berechnungen genutzt, d.h. jeder einzelne Server beantwortet mindestens eine von den Anfragen.

   Speichern Sie das Modell mit dem LTL-Claim unter: `aufgabe2/services_b.pml`
3. Falls die Verifikation nicht erfolgreich war, geben Sie den gefundenen Trail `aufgabe2/services_b.pml.trail` ab. Werfen Sie einen Blick auf den Trail und beantworten Sie mit einer kurzen Begründung, ob der Scheduler im gefundenen Trail Weak-Fair und/oder Strong-Fair ist.
4. Falls die Verifikation erfolgreich war, nennen Sie die Fairness-Annahme bei der Verifikation.

2c) Fairness II (5 Punkte)

1. Starten Sie mit der vorgegebene Datei `aufgabe2/services.pml`.
2. Formulieren Sie folgende Eigenschaft mit LTL:

   Der Server mit `_pid == 1` beantwortet unendlich viele Anfragen.
3. Ändern Sie das Modell so, dass die Verifikation des LTL-Claims unter einem Scheduler mit Weak Fairness erfolgreich ist. Dabei soll ihre Lösung weiterhin nur Nachrichtenchannels benutzen und keine gemeinsam genutzten Variablen (abgesehen von seiteneffektfreien Geistervariablen für die Verifikation). Geben Sie die geänderte Datei unter `aufgabe2/services_c.pml` ab.
4. Ist die Verifikation auch ohne Fairness-Annahme erfolgreich? Falls ja, erläutern Sie es kurz. Falls nein, geben Sie den gefundenen Trail unter `aufgabe2/services_c.pml.trail` ab.

Aufgabe 3: Büchi-Automaten (7 Punkte)

Entwerfen Sie für die LTL-Formel

$$(\neg \Diamond a) \mathcal{U} b$$

über der Signatur (den aussagenlogischen Variablen) $\mathcal{P} = \{a, b\}$ einen Büchi-Automaten, der genau die erfüllenden Läufe (Runs) der Formel akzeptiert. Verwenden Sie für den Büchi-Automaten das Alphabet $\Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$.

Ihren Büchi-Automaten geben Sie als Foto oder Grafik als Datei unter `aufgabe3/buechi.{jpg,png,svg,pdf}` in einem der Formate PNG, JPEG, SVG oder PDF ab.

Aufgabe 4: LTL (13 Punkte)

Forschende stellen ihre Ergebnisse oft auf wissenschaftlichen Konferenzen vor. Dafür schreiben sie Artikel (häufig Paper genannt) und reichen sie bei Konferenzen ein (submit). Anschließend wird das Paper von anderen Forschenden begutachtet (review) und nach wissenschaftlichen Kriterien bewertet und zur Annahme oder Ablehnung empfohlen.

Eine vollständige Lösung muss folgende Dateien im Verzeichnis `aufgabe4/` beinhalten:

- `ltl-a.{jpg,png,svg,pdf}`
- `ltl-b.{jpg,png,svg,pdf}`

4a) Formalisieren (5 Punkte)

Ein Reviewsystem soll die folgende Eigenschaft sicher stellen:

> Es gilt immer: Das Paper ist nicht reviewed bis es submitted ist. Und es gilt immer: Wenn das Paper submitted ist, wird es irgendwann reviewed.

Geben Sie zunächst eine geeignete Signatur $\mathcal{P}$ von aussagenlogischen Variablen an, um obige Eigenschaft zu formalisieren. Wählen Sie die aussagenlogischen Variablennamen sprechend, d.h. der Name deutet auf die beabsichtigte Bedeutung der aussagenlogischen Variablen hin.

Formalisieren Sie dann die obige Eigenschaft in linearer temporaler Logik (LTL) unter Verwendung Ihrer Signatur $\mathcal{P}$.

Geben Sie Ihre Lösung als Foto oder Grafik als Datei unter `aufgabe4/ltl-a.{jpg,png,svg,pdf}` in einem der Formate PNG, JPEG, SVG oder PDF ab.

4b) Büchi-Automat (8 Punkte)

Angenommen, wir haben ein Transitionssystem $\mathcal{T}$, das die Eigenschaft aus Teilaufgabe 4a) einhalten soll. Diese Eigenschaft soll mittels LTL-Modellprüfung (Model Checking) verifiziert werden.

Entwerfen Sie dafür, basierend auf der in Teilaufgabe 4a) formalisierten Eigenschaft, den Büchi-Automaten $\mathcal{B}$, der beim Modellprüfungsverfahren zusammen mit dem zum Transitionssystem gehörigen Büchiautomaten $\mathcal{B}_\mathcal{T}$ verwendet wird, um den Schnittautomaten zu erstellen.

Achtung: Es ist weder verlangt den Büchiautomaten des Transitionssystems $\mathcal{B}_\mathcal{T}$ noch den Schnittautomaten anzugeben.

Geben Sie Ihre Lösung als Foto oder Grafik als Datei unter `aufgabe4/ltl-b.{jpg,png,svg,pdf}` in einem der Formate PNG, JPEG, SVG oder PDF ab.

Aufgabe 5: Logik erster Stufe (Prädikatenlogik, First-Order Logic) (14 Punkte)

5a) Modellierung (6 Punkte)

In dieser Aufgabe sollen Sie Axiome aus der Mengenlehre in Logik erster Stufe formalisieren.

Die Signatur Σ ist über den Typen $\mathsf{TSym} = \{\mathsf{Set}, \mathsf{El}\}$ deklariert. Der Typ El steht für Elemente und der Typ Set umfasst alle Mengen von Elementen des Typs El.

Die Menge der zur Signatur Σ gehörigen Prädikatensymbole $\mathsf{PSym}_\Sigma = \{\subseteq (\mathsf{Set}, \mathsf{Set}), \in (\mathsf{El}, \mathsf{Set})\}$ deklariert genau zwei Prädikate. Dabei soll $\subseteq (N, M)$ ausdrücken, dass N eine Teilmenge von M ist. Eine Formel der Art $\in (e, M)$ drückt aus, dass e ein Element der Menge M ist. Sie dürfen in diesem Aufgabenteil beide Prädikatssymbole auch in Infix-Notation verwenden, also $N \subseteq M$ für $\subseteq (N, M)$ und $e \in M$ für $\in (e, M)$.

Für alle Elemente e vom Typ El und alle Mengen M, N vom Typ Set gilt:

1. Wenn N eine Teilmenge von M ist und e in N ist, dann ist e auch in M.
2. Wenn N keine Teilmenge von M ist, gibt es mindestens ein e in N, das nicht in M ist.
3. Genau dann wenn N und M gleich sind, ist N eine Teilmenge von M und M eine Teilmenge von N.

Formalisieren Sie die obigen mengentheoretischen Eigenschaften in Logik erster Stufe unter Verwendung der Signatur Σ. Sie dürfen die Signatur Σ um zusätzliche Variablen- oder Funktionssymbole erweitern, sofern es für die Formalisierung notwendig ist. In dem Falle deklarieren Sie bitte Ihre zusätzlichen Symbole explizit.

Geben Sie Ihre Lösung als Foto oder Grafik als Datei unter `aufgabe5/fol-a.{jpg,png,svg,pdf}` in einem der Formate PNG, JPEG, SVG oder PDF ab.

5b) KeY (8 Punkte)

Beweisen Sie unter Annahme der Axiome in 5a) die Eigenschaft

> Für alle Elemente e vom Typ El und alle Mengen M, N vom Typ Set gilt: Wenn e in M ist aber nicht in N, dann sind M und N nicht gleich.

in KeY.

Vervollständigen Sie dafür die Datei `aufgabe5/fol.key`, inklusive der nötigen Sorten und Prädikate. Für den Beweis dürfen Sie nur die Regeln aus der Vorlesung verwenden und die Automatik nicht benutzen.

Speichern Sie Ihren Beweis und geben Sie den gespeicherten Beweis unter `aufgabe5/fol.proof` ab.

Hinweis: Beachten Sie in ihrer Formalisierung, dass in KeY bestimmte Namen für Prädikats- und Funktionssymbole schon vergeben sind, wie z.B. die Namen „subset“ oder „sub“ und nicht verwendet werden können. Bei Namenskollisionen, wählen Sie deshalb bitte andere noch nicht vergebene Namen für Ihre eigenen Prädikats- und/oder Funktionssymbole.