HUMAN CAPITAL
NATIONAL COHESION STRATEGY

Wrocław University of Technology

EUROPEAN
UNION
EUROPEAN
SOCIAL FUND

THE DEVELOPMENT OF THE POTENTIAL AND ACADEMIC PROGRAMMES OF WROCŁAW UNIVERSITY OF TECHNOLOGY

Paweł Głuchowski

Modelowanie i Analiza Systemów Informatycznych

Logika Temporalna w Analizie Systemu

(wersja poprawiona, 2013)

Materiały są tłumaczeniem instrukcji laboratoryjnych p.t.

Paweł Głuchowski

Information Systems Analysis

Temporal Logic in System Analysis

Project co-financed by European Union within European Social Fund

Instrukcja do 1. laboratorium z Logiki Temporalnej w Analizie Systemu

Temat: logika LTL i opis własności systemu.

Podczas tego laboratorium przećwiczysz:

- rysowanie reprezentacji danych formuł LTL na liniowych strukturach czasowych,
- pisanie formuł LTL reprezentujących dane liniowe struktury czasowe,
- specyfikowanie własności danego systemu w LTL.

Przewodnik do ćwiczeń 1 i 2:

Rysunek 1 (poniżej) przedstawia liniową strukturę czasu, która składa się z dużych punktów, i łuków łączących je w linie. Punkty opisane są symbolami, np. s_i, i kolorami. Jest też wielokropek między dwoma łukami.

Każdy punkt reprezentuje stan (moment) w strukturze czasu. Nad każdym punktem jest symbol stanu i jego numer (np. stan nr 7 to s_7). Jeśli punkt jest czarny, to jego stan oznaczony jest wartością *prawda* (co znaczy, że wszystko z wyjątkiem *fałszu* jest możliwe). Inny kolor reprezentuje formułę, która w tym stanie jest prawdziwa. Ta funkcja etykietująca stany wytłumaczona jest pod strukturą czasową. Zwykle zamiast używania kolorów pisze się odpowiadające im formuły przy stanach.

Każdy łuk reprezentuje przejście między stanami. Jeśli łuk łączy dwa stany, to stany te przedstawiają kolejność dwóch momentów, tak że między nimi nie ma żadnych momentów. Jeśli między dwoma łukami jest wielokropek, to można by na jego miejscu umieścić pewną liczbę innych stanów, które wszystkie byłyby identyczne ze stanem, z którego wychodzi lewy łuk.

Rysunek 1.

Rysunek 1 przedstawia formułę: $p \wedge Fq$, która jest prawdziwa w stanie s_0. Niech $i \geq 1$ (dla indeksowania stanów). Jeśli i=1, to oczywiście stany s_1 i s_i są tym samym stanem i oznaczone są przez *prawda*$\wedge$q $\equiv$ q. Jeśli i=2, to istnieje tylko jeden stan s_1, oznaczony przez *prawdę*, pomiędzy stanami s_0 i s_i. W pozostałych przypadkach istnieje pewna większa liczba stanów między s_1 i s_i, wszystkich oznaczonych tak jak s_1, to jest przez *prawdę*. Zawsze ważne jest, aby podać relację między stanami połączonymi łukami z wielokropkiem między nimi.

Zadania

Zadanie 1.

Narysuj liniową strukturę czasową reprezentującą podane formuły LTL, gdzie każda formuła jest prawdziwa w stanie s_0. Umieść symbole formuł *p* i *q* przy tych stanach, w których formuły te są prawdziwe. Stany bez symboli tych formuł będą rozumiane jako stany oznaczone przez *„prawda”* (nie pisz tam *prawda*).

a) Narysuj liniową strukturę czasową dla następującej formuły LTL:

$X(p \wedge G(p \Rightarrow Xq \wedge XXXp))$

b) Narysuj liniową strukturę czasową dla następującej formuły LTL:

$\neg p \wedge pUq \wedge FGq$

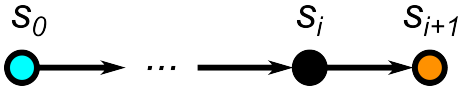
c) Narysuj liniową strukturę czasową dla następującej formuły LTL:

$pU(Xq) \wedge p$

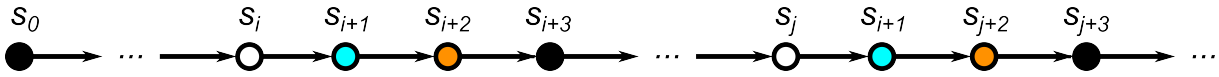
Zadanie 2.

Napisz formuły LTL z symbolami: p, q i temporalnymi lub logicznymi symbolami, przedstawiające podane liniowe struktury czasowe dla następującej funkcji etykietującej: n (niebieski), p (pomarańczowy), b (biały), *prawda* (czarny). Nie pisz *„prawda”* w formułach.

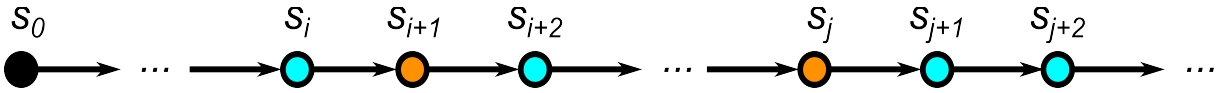
a) Napisz formułę LTL przedstawiającą następującą strukturę czasową dla i>0:



b) Napisz formułę LTL przedstawiającą następującą strukturę czasową dla i>0 i dla j≥i+3:



c) Napisz formułę LTL przedstawiającą następującą strukturę czasową dla i≥0 i dla j>i+2:



Zadanie 3.

Opisz jak najwięcej własności wybranego protokołu komunikacyjnego.

Wybierz ten z następujących protokołów komunikacyjnych, który najlepiej znasz: FTP, SSH, Telnet i TLS i wykonaj następujące czynności:

- Spróbuj opisać jak najwięcej aspektów używania tego protokołu w języku naturalnym i w formułach LTL, gdzie własność, zdarzenie itp. symbolizowane będzie literą. Upewnij się wpierw, że znasz te protokoły.
- Spróbuj znaleźć najważniejsze własności, które muszą być spełnione przez użytkowników protokołu, aby zapewnić poprawne użycie protokołu. Użytkownikami są raczej programy niż ludzie.
- Jeśli to możliwe, narysuj dla każdej formuły liniową strukturę czasową.

To, co będziesz tu robił, będzie wykorzystane w części przyszłych zajęć, gdy będziesz wykorzystywał podobną logikę (CTL) aby modelować użycie protokołu i automatycznie weryfikować jego własności.

Instrukcja do 2. i 3. laboratorium z Logiki Temporalnej w Analizie Systemu

Temat: Modelowanie systemu jako automaty i specyfikowanie jego własności w LTL.

Podczas tych dwóch laboratoriów przećwiczysz:

- używanie narzędzia do modelowania, specyfikowania i weryfikowania – UPPAAL,
- modelowanie podanych jednoprocesowych systemów jako deterministyczne skończenie stanowe automaty,
- specyfikowanie własności tych modeli w LTL.

Przewodnik do ćwiczeń:

Na początku potrzebujesz „UPPAAL 4.0 Small Tutorial”, który możesz pobrać z oficjalnej strony UPPAAL pod www.uppaal.com.

(bezpośredni adres: www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf).

Aby zapoznać się z aplikacją UPPAAL, wykonaj instrukcje z rozdziałów 3.1 i 3.2 z tego tutorialu. Zajmie to trochę czasu, ale nauczy cię, jak używać tego narzędzia. To nie problem, jeśli nie zrozumiesz wszystkiego, co robią tamtejsze automaty. Po następnych laboratoriach już będziesz.

Jeśli będziesz miał jakieś trudności w modelowaniu, symulowaniu lub weryfikowaniu swoich systemów w UPPAAL-u, zajrzyj do pozostałych rozdziałów tego tutorialu (lub innych opublikowanych tamże).

Zadania

Zadanie 1.

Zamodeluj podane systemy jako deterministyczne automaty skończenie stanowe.

Aby zbudować automat, określ stany, w których ten system może być i przejścia, które są możliwe między tymi stanami. Pamiętaj: przejścia muszą być wybierane deterministycznie, więc opisz przejścia właściwymi warunkami i zmianami wartości zmiennych.

Po zbudowaniu automatów przesymuluj ich działanie, aby sprawdzić, czy działają właściwie.

a) Zbuduj model cyfrowego zamka, który otwiera się po wybraniu sekwencji: 1,7,8,3 i pozostaje w stanie otwarcia. Można wprowadzać wszystkie 10 cyfr od 0 do 9 nieskończenie długo, aż do wprowadzenia poprawnej sekwencji. Poprawną sekwencję można więc wprowadzić w dowolnym czasie, czyli po wprowadzeniu dowolnego ciągu cyfr niezawierającego wymaganej sekwencji. Upewnij się, że sekwencje kończące się na: ...11783, ...171783, ...1781783 otworzą zamek.

Wskazówka: Użyj stanów *committed*, które opuszczane są w tym samym momencie, w którym stają się aktywne.

b) Zbuduj model maszyny do biletów, która przyjmuje monety, oblicza na bieżąco ich sumaryczną wartość i wydaje bilet. Jest tylko jeden rodzaj biletu, więc użytkownik inicjuje działanie maszyny przez wrzucenie pierwszej monety. Monety można wrzucać dopóty, dopóki suma ich wartości jest mniejsza niż cena biletu. Maszyna nie zwraca reszty. Po wydrukowaniu biletu maszyna powraca do początkowego stanu.

Zauważ, że jest wiele różnych monet. Użytkownik może użyć przynajmniej jednej monety lub tyle ich chce.

Zadanie 2.

Określ własności modeli z zadania 1 w logice LTL.

Spróbuj znaleźć jak najwięcej własności swoich modeli. Rozważ zwłaszcza następujące rodzaje własności:

- blokada *(deadlock)* systemu nigdy nie jest możliwa, z wyjątkiem końcowego stanu (jeśli jest);
- jeśli pewne stany muszą zostać osiągnięte dla poprawnego wykonania procesu, to zostaną w końcu osiągnięte;
- jeśli pewne stany nie mogą zostać osiągnięte dla poprawnego wykonania procesu, to nie zostaną nigdy osiągnięte;
- wszystkie poprawne akcje systemu mogą zostać wykonane, czyli wszystkie poprawne stany mogą zostać osiągnięte; itd.

Zapisz te własności jako formuły LTL niezawierające zagnieżdżania operatorów temporalnych (UPPAAL nadal ma pewne ograniczenia) i zweryfikuj te własności za pomocą weryfikatora UPPAAL. Jeśli weryfikacja nie zakończy się sukcesem, popraw swój model (automat) i/lub temporalne formuły.

Wskazówka: Zamiast operatora temporalnego G napisz '*A[]*', zamiast F napisz '*E<>*', zamiast $\Rightarrow$ napisz '*imply*', zamiast $\wedge$ napisz '*and*' i zamiast $\vee$ napisz '*or*'.

Wskazówka: Aby sprawdzić możliwość blokady, napisz '*A[] not deadlock*'.

Temat: Modelowanie systemu jako automaty i specyfikowanie jego własności w CTL.

Podczas tych dwóch laboratoriów przećwiczysz:

- modelowanie podanego współbieżnego systemu jako niedeterministyczne skończenie stanowe automaty,
- specyfikowanie własności tego modelu w CTL.
- modelowanie i specyfikowanie systemu, który używa protokołu komunikacyjnego (z pierwszego laboratorium z logiki temporalnej w analizie systemu).

Przewodnik do zadań (z jednym ćwiczeniem wewnątrz!):

Celem każdego z zadań jest: zbudowanie poprawnego modelu systemu, używając więcej niż jednego automatu, określenie jego własności (np. bezpieczeństwo, osiągalność) w logice CTL i zweryfikowanie ich.

Jeśli będziesz miał jakieś trudności w modelowaniu, symulowaniu lub weryfikowaniu swoich systemów w UPPAAL-u, zajrzyj do tutorialów ze strony www.uppaal.com.

Poniżej znajduje się prosty przykład systemu pewnych programów komunikujących się ze sobą (należy go wykonać). Model wykonany jest w UPPAAL. Pewne własności modelu podane są w CTL, co pozwala je zweryfikować.

Przykład.

Zbudujmy prosty system typu *chat*, składający się z dwóch rodzajów programu: aplikacji użytkownika U i serwera S. Są trzy instancje aplikacji użytkownika: U(1), U(2) i U(3) i jedna instancja aplikacji serwera: S. Zamodelujemy tylko logowanie i wylogowanie, zakładając spełnienie następującej własności (poza tym, że system musi działać): każdy użytkownika może się zalogować lub wylogować.

Utworzymy dwa szablony *(templates):* U dla aplikacji użytkownika i S dla serwera. Napisz w deklaracjach systemu (system declarations): *system U, S;*

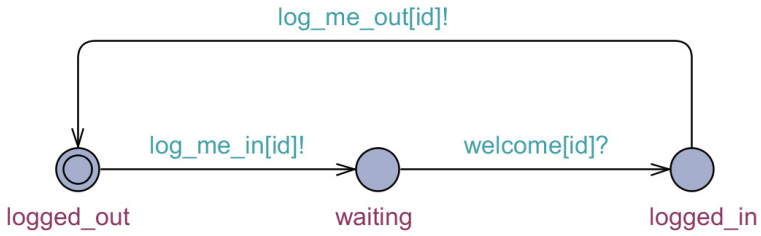
Zaczynamy od utworzenia szablonu dla użytkownika. Nazwij go 'U' i wpisz parametr w polu obok nazwy: *const N id*

Teraz chcemy mieć trzy instancje aplikacji użytkownika numerowane od 0 do 2, więc napisz w deklaracjach projektu *(declarations)*: *typedef int[0,2] N;*

Następnie zbuduj automat przedstawiony na rysunku 1, gdzie nazwy stanów są fioletowe, stan *logged_out* jest początkowy (*initial*), komunikaty wysyłane (!) i odbierane (?) przez kanały są jasnoniebieskie, a *id* jest numerem użytkownika. Aby używać kanały, napisz w deklaracjach projektu *(declarations):* *chan log_me_in[N], log_me_out[N];*

broadcast chan welcome[N], bye[N];

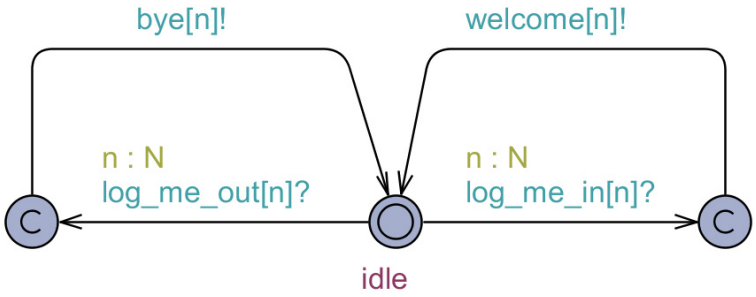
chan to kanał normalnej komunikacji jeden do jednego. *broadcast chan* – jeden do wielu.



Rysunek 1. *Szablon użytkownika 'U'*

Teraz możemy zbudować automat dla serwera. Utwórz nowy szablon, nazwij go 'S' i zbuduj automat przedstawiony na rysunku 2. Serwer mógłby mieć tylko jeden stan (*idle*), ale w celu wykonania więcej niż jednej operacji tego samego typu (np. obsługa dwóch komunikatów) potrzebujemy zwykle jednego przejścia dla każdej z nich. Możliwe to jest dzięki użyciu stanów oznaczonych jako 'c' *(committed).*

W wysyłaniu i odbieraniu komunikatów wybór instancji 'U' wykonywany jest przez własność przejścia *select* (brązowy kolor). Na przykład, jeśli serwer otrzymuje komunikat *log_me_in* od U(1), to zachowuje na krótko numer użytkownika 'n' (tutaj, n:=1), co pozwala serwerowi wysłać komunikat *welcome[n]!* witający zalogowanego użytkownika U(n) czyli U(1).



Rysunek 2. *Serwer 'S'*

Zadania

Zadanie 1.

Ten system nie w pełni obsługuje komunikaty *welcome* i *bye*. Dokończ go i popraw, jeśli trzeba.

Dla naszego systemu możemy podać i zweryfikować przynajmniej te własności:

- system nie może się zablokować: *A[] not deadlock*
- każdy użytkownik może się zalogować: *E<> forall (i : N) U(i).logged_in*

Zauważ, że zagnieżdżanie operatorów temporalnych w UPPAAL nie jest możliwe, co ogranicza nasze możliwości weryfikacji własności systemu.

Zadanie 2.

Zbuduj model systemu dwóch lub więcej programów komunikujących się ze sobą wg wybranego (w 3. zadaniu 1. laboratorium) protokołu komunikacyjnego.

Działanie systemu, jego własności w języku naturalnym i jako formuły LTL jest już przygotowane. Zbuduj teraz ten system jako automaty w UPPAAL i zapisz i zweryfikuj jego własności jako formuły CTL. Zweryfikuj tak wiele własności, jak to możliwe, by potwierdzić, że system działa zgodnie z protokołem. Jeśli danej formuły nie można zweryfikować w UPPAAL, zrób to „ręcznie".

Podsumowanie – należy:

- zbudować niedeterministyczne automaty, z wykorzystaniem zegarów, kanałów itp. i przetestować je, by upewnić się, że działają zgodnie z protokołem komunikacyjnym;
- sformułować jak najwięcej własności, w tym bezpieczeństwa i osiągalności, dla systemu w logice CTL;
- automatycznie (jeśli to niemożliwe, to ręcznie, przez symulację) zweryfikować te własności, by potwierdzić spełnienie wymagań protokołu.

Jeśli weryfikacja nie jest pomyślna, popraw automaty lub formuły.

Wskazówka: Aby zapisać formułę $AG(p \Rightarrow AFq)$ napisz p --> q