Jak uprościć model protokołu?

Jarosław Bąbel

26 maja 2004

Spis treści

1 Wstęp

Bezpośrednie zapisanie modelu protokołu z reguły jest proste. Kod naturalnego modelu zapisany np. w Promeli jest łatwy do czytania i analizy. Niestety w tej formie rzadko pozwala na weryfikację przy użyciu narzędzi komputerowych (w tym wypadku przy użyciu Spina), ponieważ wymaga zbyt dużo pamięci. Jedynym sensownym wyjściem jest dokładna analiza i uproszczenie modelu. Wady tego rozwiązania to pracochłonność i mało czytelny wynik (szczególnie dla kogoś innego niż autor).

W dalszej części chciałbym przedstawić kilka pomysłów, dzięki którym można drastycznie zmniejszyć przestrzeń stanów potrzebną do analizy modelu protokołu. Zbudujemy dwa modele systemu naturalny i abstrakcyjny.

2 Krótki opis protokołu Five Packet Handshake Protocol

Jest to część TCP. Jej zadaniem jest przesłanie wiadomości między dwoma węzłami przez medium, które może gubić, przestawiać i powielać przesyłane pakiety. Zakładamy, że nadawca otrzymuje z otoczenia ciąg wiadomości i przesyła je w kolejności do odbiorcy. Zadaniem odbiorcy jest przekazanie komunikatów do własnego środowiska.

Na przesłanie wiadomości składa się wymiana pięciu pakietów:

1. Nadawca wysyła *needuid(v)*, gdzie *v* to unikalny identyfikator. Ten pakiet to prośba do odbiorcy o nowe *id* do przesłania właściwej wiadomości.
2. Odbiorca wysyła *accept(u,v)*, gdzie *u* to nowe *id*.
3. Nadawca wysyła wiadomość *m* w komunikacie *send(m,u)*.
4. Odbiorca potwierdza odebranie wiadomości wysyłając *ack(u)*.
5. Nadawca kończy wymianę pakietów nadając *cleanup(u)*, po odebraniu tego komunikatu odbiorca nie może używać wartości *u*.

3 Model naturalny

Z grubsza przepisujemy specyfikację protokołu na model w Promeli. Dla każdego rodzaju komunikatu definiujemy osobny kanał. Etykiety w procesie nadawcy i odbiorcy pokazują kolejne fazy protokołu, są to tzw. stany kontrolne.

4 Model abstrakcyjny

Podstawowy pomysł: przechowujemy tylko tyle informacji ile potrzeba do przewidzenia możliwego dalszego zachowania protokołu. Na stan protokołu składają się: wartości zmiennych, zawartość kanałów komunikacyjnych, stan kontrolny.

5 Jak to zrobić?

Trzeba zauważyć, że operacje wykonywane na zmiennych protokołu (patrząc jednocześnie na nadawcę i odbiorcę) to porównanie oraz przypisanie nowej (dotąd nie używanej) wartości.

Zmienne i kanały zastępujemy zbiorem zmiennych typu boolean opisujących stan protokołu. Stany kontrolne (przepływ sterowania między fazami protokołu) modelu abstrakcyjnego są takie same jak modelu naturalnego.

6 Funkcja wiążąca modele

Związek między modelami odzwierciedla funkcja ukonkretniająca γ, która przekształca zbiory stanów modelu abstrakcyjnego w zbiory stanów modelu naturalnego.

Każdą zmienną boolowską modelu abstrakcyjnego możemy traktować jak zbiór abstrakcyjnych stanów protokołu (czyli takich, w których odpowiednio b lub $\neg b$ jest prawdziwe). Te zbiory odpowiadają zbiorom stanów protokołu naturalnego $\gamma(b)$ i $\gamma(\neg b)$, które nie muszą być rozłączne (w ten sposób można zmniejszyć rozmiar modelu zlepiając stany, przykład będzie później). Uzupełniamy definicję funkcji γ dla dowolnych zbiorów stanów:

$$\gamma(\phi \wedge \phi') \equiv \gamma(\phi) \wedge \gamma(\phi')$$

$$\gamma(\phi \vee \phi') \equiv \gamma(\phi) \vee \gamma(\phi')$$

7 Jak dobrać zmienne boolowskie?

Dla każdej pary zmiennych protokołu ustalamy zmienną boolowską, która mówi o równości tych zmiennych. Dla x i x' przyjmujemy zmienną x_eq_x', która przyjmuje wartość *true* wtedy i tylko wtedy gdy $x = x'$. Funkcję γ definiujemy tak:

$$\begin{aligned} \gamma(x\_eq\_x') &\equiv x = x' \\ \gamma(\neg x\_eq\_x') &\equiv x \neq x' \end{aligned}$$

Z kanałami jest trochę trudniej. Potencjalny wpływ odebrania komunikatu na przebieg protokołu można ustalić sprawdzając, czy jego parametry są równe zmiennym protokołu albo parametrom innych komunikatów być może w innych kanałach. Każda wartość boolowska dla kanału mówi o tym, czy kanał *może* przechowywać komunikat pewnej postaci (np. kanał x zawiera komunikat (a, b) taki, że $a = u$ i $b \neq w$, gdzie u i w to zmienne protokołu). Określenie *może zawierać* odzwierciedla fakt, że medium gubi komunikaty. Jedyne czego możemy być pewni, to nieobecność pewnego konkretnego komunikatu w kanale (jeżeli nigdy go nie wysyłaliśmy).

8 Definicja funkcji γ dla kanałów

Przykład na to, że $\gamma(b)$ i $\gamma(\neg b)$ nie muszą być rozłączne:

$$\gamma(c\_mc\_\psi) \equiv true \tag{1}$$

$$\gamma(\neg c\_mc\_\psi) \equiv \neg\exists\langle u_1, \ldots, u_n\rangle \in c.\psi \tag{2}$$

Jeżeli (1) zdefiniowalibyśmy jako negację (2), to dla każdego stanu, w którym zachodzi c_mc_ψ musi też istnieć stan, w którym zachodzi $\neg c\_mc\_\psi$, bo komunikat może być w kanale, ale może go też nie być.

Przy takiej definicji zlepiamy niektóre ze stanów: w każdym stanie wiemy, że kanał *może* coś zawierać, ale tylko w niektórych nie istnieje odpowiedni komunikat (wiemy, że go nie wysyłaliśmy). W ten sposób nie tracimy informacji, ale zmniejszamy model.

9 Na co trzeba uważać?

Dla zmiennych utrzymujących sekwencje identyfikatorów jest trochę inaczej. Ich istotną cechą jest to, że przyjmują kolejne wartości tzn. abstrakcja operacji porównania i przypisania *nowej* wartości nie wystarcza.

Problem rozwiążemy unikając modelowania tych zmiennych przez typ integer. Przyjmujemy, że każda nadawana wiadomość jest nowa. Odbiór komunikatu wiąże się ze sprawdzeniem, czy odebrany komunikat jest tym, który nadawca próbuje wysłać. Do modelowania duplikatów używamy flagi, która zaznacza, czy ten komunikat był wysłany wcześniej, flaga jest resetowana, gdy nadawca ma zamiar wysłać nowy komunikat. W ten sposób otrzymujemy symulację zachowania sekwencji identyfikatorów.

10 Co robimy ze zmiennymi abstrakcyjnymi?

Usuwamy zmienne, które nie mają znaczenia dla działania protokołu (np. nigdzie nie są porównywane albo nigdy nie sprawdzamy, czy odebrany komunikat jest identyczny z wartością jakiejś zmiennej).

Mamy już zbiór zmiennych o znaczeniu odpowiadającym stanom modelu naturalnego. Teraz konstruujemy model abstrakcyjny, który symuluje (co to oznacza patrz dalej) zachowanie modelu naturalnego. Instrukcje dozorowane przerabiamy na odpowiedniki abstrakcyjne. Musimy też odpowiednio modyfikować wartości zmiennych boolowskich. Przykład:

`Ackchan?u -> (u == SmsgUID) -> Cleanupchan!u;`

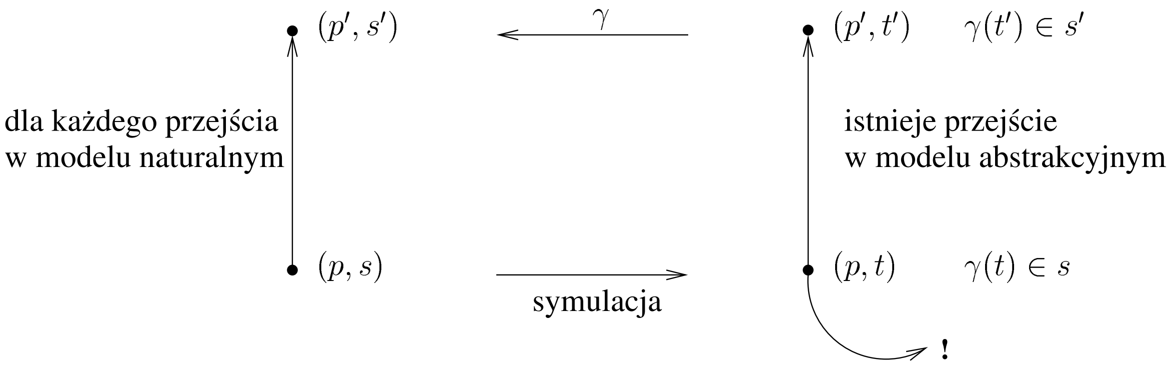
przerabiamy na:

```
Ackchan_mc_SmsgUID ->
   Cleanupchan_mc_RmsgUID =
           (Cleanupchan_mc_RmsgUID || RmsgUID_eq_SmsgUID);
   Cleanupchan_mc_RlastUID =
           (Cleanupchan_mc_RlastUID || RlastUID_eq_SmsgUID);
```

W praktyce można wprowadzać wiele bardzo abstrakcyjnych optymalizacji. Kod, który powstaje w ten sposób jest zupełnie nieczytelny (trzeba objąć całość, żeby zrozumieć poszczególne kroki).

11 Weryfikacja wymagań

Model abstrakcyjny symuluje model naturalny tzn. jeżeli model naturalny może wykonać atomowy krok z punktu p do p' zmieniając jednocześnie stan zmiennych i kanałów z s na s' to dla każdego stanu t zmiennych abstrakcyjnych takiego, że $\gamma(t) = s$ powinien istnieć stan t' taki, że $\gamma(t') = s'$ i taki, że model abstrakcyjny może wykonać krok z punktu p do p' zmieniając stan zmiennych z t na t'.



W LTL-u (czyli w Spinie) wszystkie formuły zaczynają się od „dla każdej ścieżki...”. Jeżeli jakaś własność bezpieczeństwa zachodzi w modelu abstrakcyjnym, to musi też zachodzić w modelu naturalnym. W modelu abstrakcyjnym mogą być przejścia nie „używane” w symulacji. Ważne, żeby nie szkodziły własnościom bezpieczeństwa.

12 Podsumowanie

Opisane podejście do modelowania jest bardzo skuteczne i bardzo trudne. Model naturalny nie pozwalał na analizę komputerową. Dla modelu abstrakcyjnego Spin pokazuje 575 osiągalnych stanów. Przy użyciu Spina i modelu abstrakcyjnego pokazano, że protokół dostarcza wiadomości bez duplikacji. Jeżeli wykluczy się możliwość zawieszenia i restartu nadawcy lub odbiorcy, to liczba osiągalnych stanów maleje do 46 i można pokazać, że protokół dostarcza komunikaty bez gubienia.