Weryfikacja modelowa

Protokoły komunikacyjne
2006/2007

Paweł Kaczan
pk209469@students.mimuw.edu.pl

Plan

- Wstęp
- Trzy kroki do weryfikacji modelowej
- Problemy
- Podsumowanie
- Dziedziny zastosowań

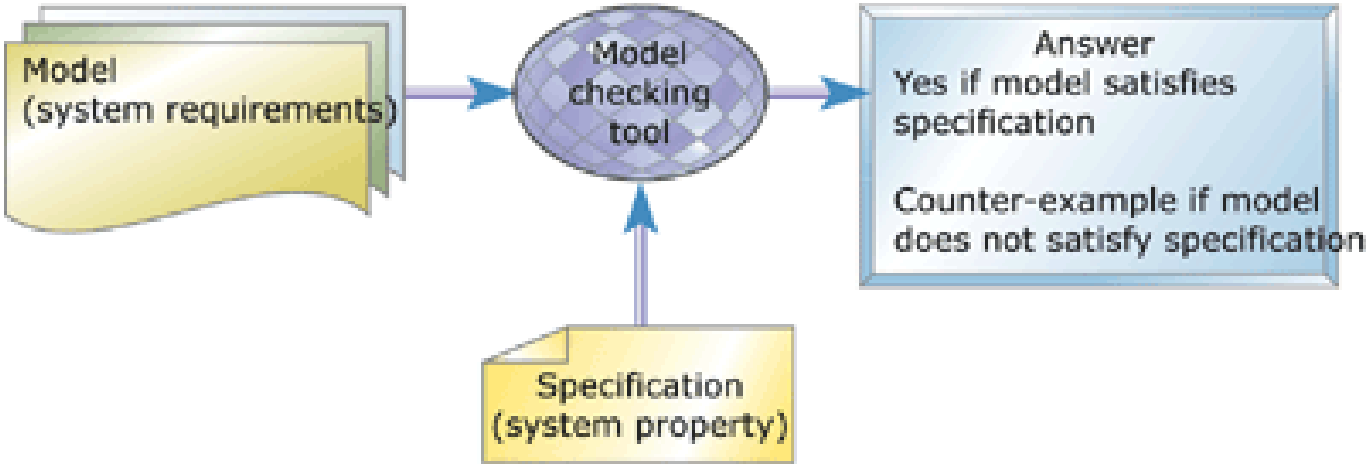
Problemy z wymaganiami

- Czy opis wymagań odpowiada intencjom klienta?
- Czy wymagania są sformułowane w sposób jasny i klarowny?
- Czy sformułowane są w sposób modularny, zapewniający podzielność?
- Czy wymagania mogą stanowić bazę do formułowania testów akceptacyjnych?
- Czy pozwalają na swobodę w technicznej realizacji?

Weryfikacja modelowa vs inne techniki

- Opis słowny, tabele, slajdy, itp.
- UML
- Prototypy
- Weryfikacja modelowa

Etapy weryfikacji modelowej



1. Modelowanie
2. Specyfikacja wymagań
3. Weryfikacja

Krok pierwszy - modelowanie

- Czym jest modelowanie? – Formalny opis systemu. Skończenie stanowe automaty.
- Ograniczenia czasowe, pamięciowe – abstrakt systemu.
- Struktura Kripkego do modelowania systemu.

Struktura Kripkego

- Typ niedeterministycznego skończonego automatu
- Czwórka M = (S, S_0, R, L), gdzie
  - S – skończony zbiór stanów
  - S_0 – zbiór stanów początkowych
  - R – relacja przejścia
  - $L:S \rightarrow 2^{AP}$ – funkcja etykietująca wierzchołki zbiorami atomowych formuł, które są prawdziwe w danym stanie

$$M \models \phi \text{ wtw gdy } \forall s \in S_{\text{pocz}} \; s \models \phi$$

Przykład

S = (S1, S2, S3, S4)

S_0 = { S1 }

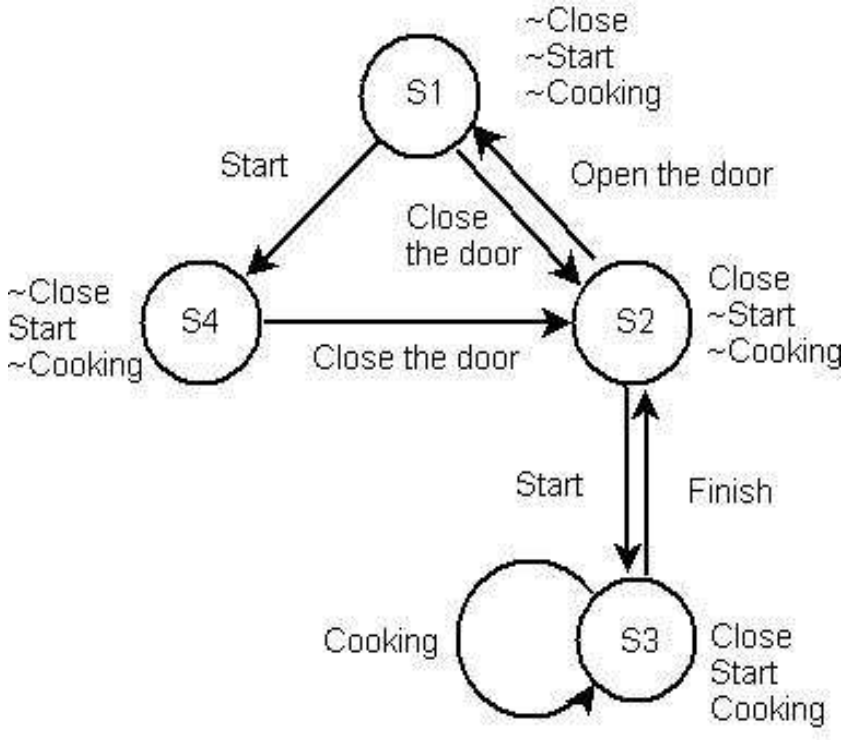
R = ({S1,S2}, {S2,S1},{S1,S4},
{S4,S2}, {S2,S3}, {S3,S2},
{S3, S3}

L(S1) = {¬close,¬start,¬cooking}

L(S2) = {close,¬start,¬cooking}

L(S3) = {close,start,cooking}

L(S4) = {¬close,start,¬cooking}



Krok drugi - specyfikacja

- Logika klasyczna?
- **Logika temporalna** to logika umożliwiająca rozważanie zależności czasowych bez wprowadzania czasu explicite.

Operatory logiki temporalnej

- **Gφ** oznacza, że od danego punktu już zawsze będzie miało miejsce φ. *(globally)*
- **Fφ** oznacza, że kiedyś w przyszłości będzie miało miejsce φ. *(finally)*

Przykład:

G(pociąg jedzie → ¬szlaban podniesiony)

- zawsze jeśli pociąg jedzie, to szlaban jest opuszczony

GF(szlaban podniesiony)

- *nigdy nie będzie tak, że szlaban nie będzie już cały czas opuszczony*

G i F pozwalają na umiejscawianie zjawisk w czasie, ale nie dają nam wielu możliwości przedstawiania zależności czasowych pomiędzy zjawiskami.

Operatory logiki temporalnej

- **φ*U*ψ** – kiedyś nastąpi ψ. Do czasu jego pierwszego wystąpienia zawsze będzie φ. *(until)*
- **φ*R*ψ** – ψ będzie zachodziło tak długo, aż nie zajdzie φ. *(release)*

W logikach z dyskretnym czasem można zdefiniować operator

- **Xφ**, oznaczający że φ nastąpi w następnej chwili. XXφ oznacza więc że nastąpi za 2 chwile, XXXXXφ za 5 chwil itd. *(next)*

Logiki temporalne

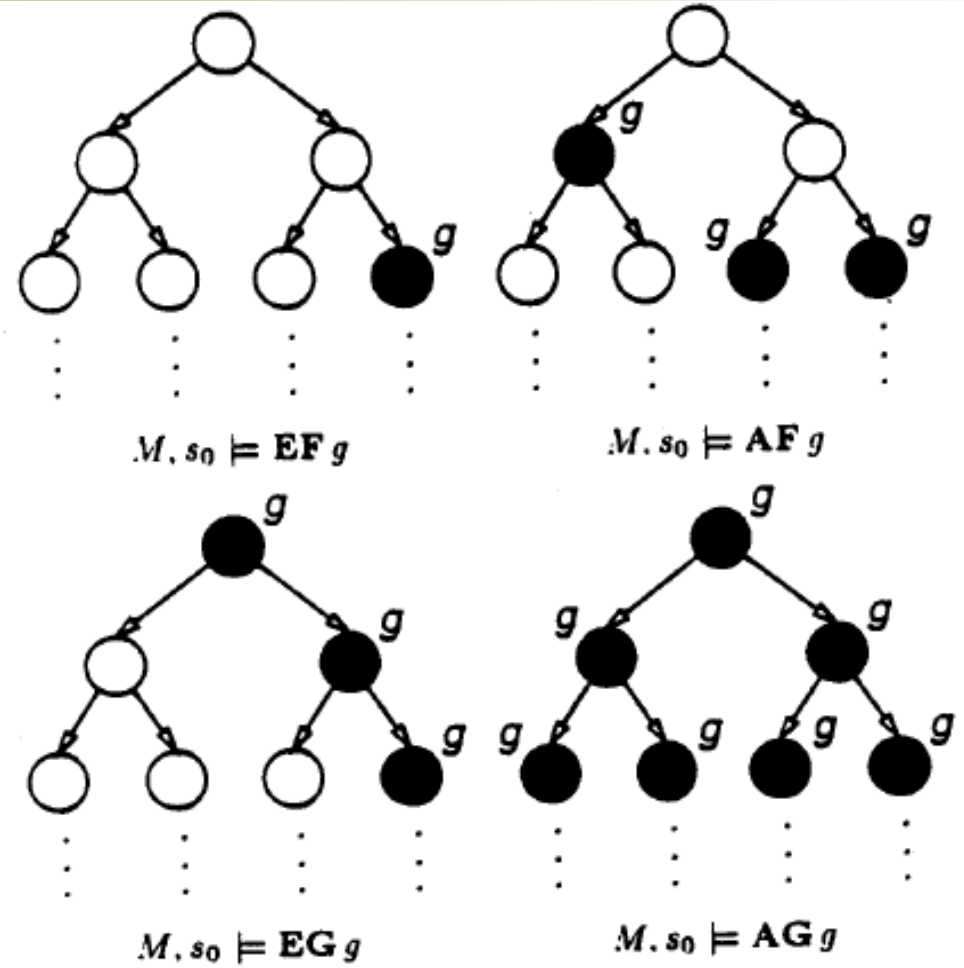
- zakładają liniową strukturę czasu
  - Logika LTL (Linear Temporal Logic)

  wykorzystuje operatory G, F, X, U, R.
- zakładają rozgałęzioną strukturę czasu
  - Logika CTL*(Computation Tree Logic)

  wykorzystuje operatory LTL oraz
    - ***A*φ** – dla każdej ścieżki zachodzi φ
    - ***E*φ** – istnieje taka ścieżka obliczeń, że φ
  - Logika CTL

  ograniczenie logiki *CTL** - operatory mogą występować tylko parami – operator ścieżkowy, operator stanowy

Rozgałęziony czas w CTL



Binarne diagramy decyzyjne

- Kanoniczna forma reprezentacji formuł.
- Struktura acyklicznego grafu skierowanego.
- Ustalony porządek na zmiennych.
- Rozmiar zależny od porządku na zmiennych.
- Zapewniają efektywne wykonywanie operacji.

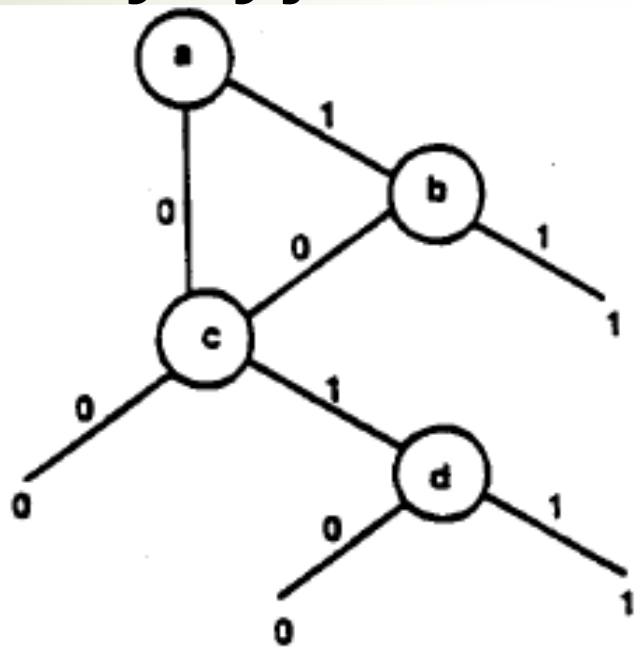


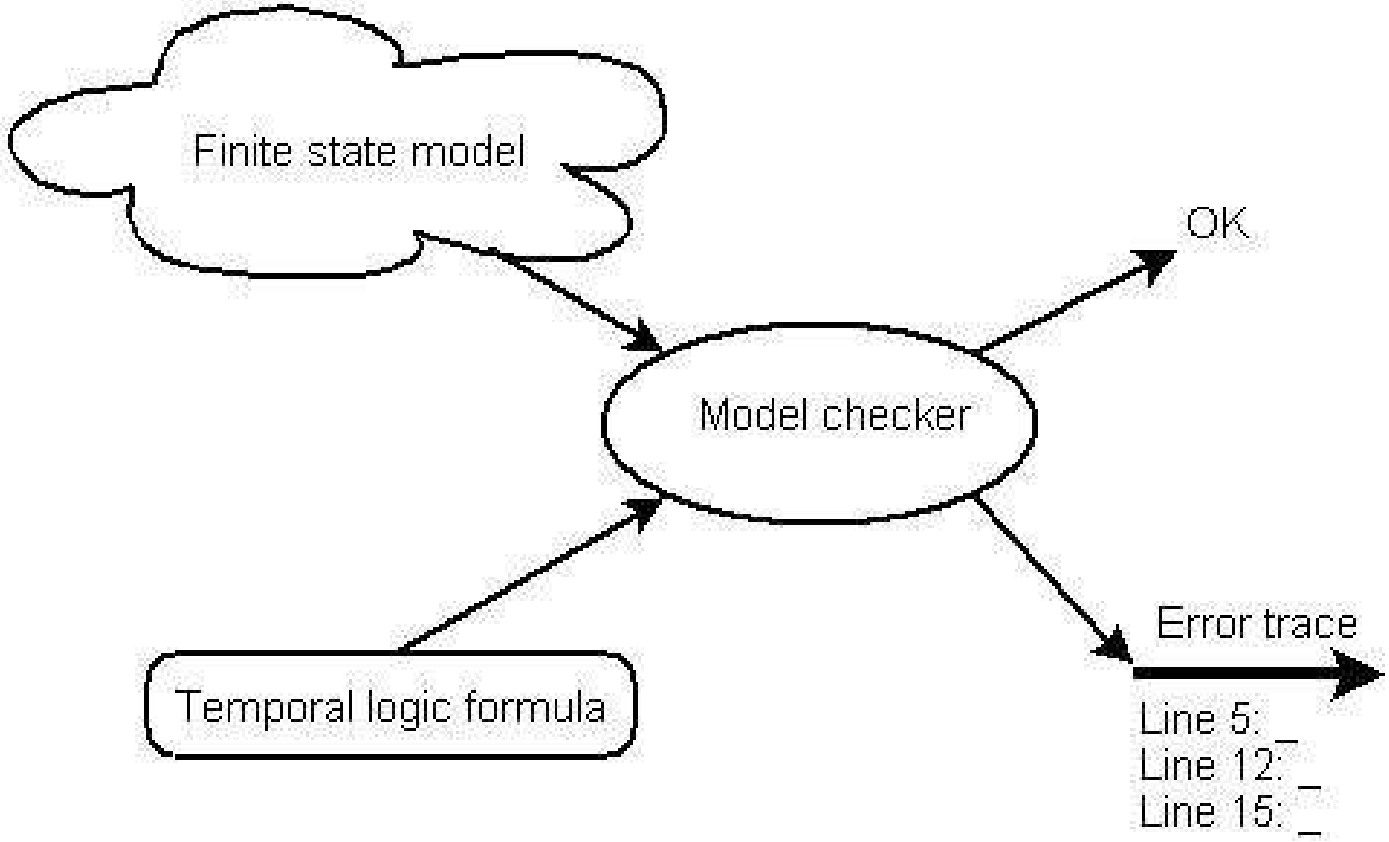
Diagram dla ((a i b) lub (c i d)) przy założeniu, że a < b < c < d.
Formuła prawdziwa dla wartościowania:
{a = 1, b = 0, c = 1, d = 1}

Przykład

Specyfikacja w logice CTL:

- AG (start → AF cooking)
- AG ((close i start) → AF cooking)

Krok trzeci - weryfikacja



Przykład

AG (start → AF cooking)

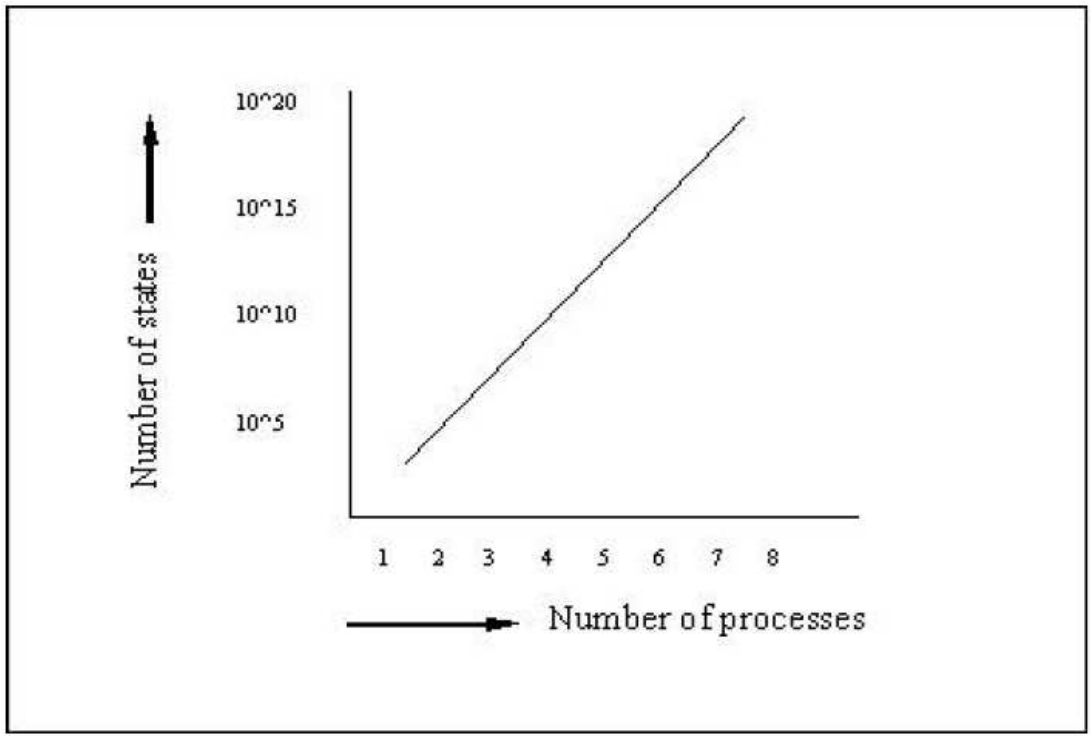
1. Zamień na ¬EF (start i EG ¬ cooking))
2. Od składowych do całej formuły:
   - S (start) = {S3, S4}
   - S (¬cooking) = {S1, S2, S4}
   - S (EG ¬ cooking) = {S1, S2, S4}
   - S (start i EG ¬ cooking) = {S4}
   - S (EF (start i EG ¬ cooking)) = {S1, S2, S3, S4}
   - S (¬ (EF (start i EG ¬ cooking))) = {}
3. Analiza wyniku

Przykład

AG ((close i start) → AF cooking)

1. Zmień na ¬ EF(close i start i EG ¬ cooking)
2. Od formuł składowych:
   - S (close)= {S2, S3}
   - S (start)= {S3, S4}
   - S (¬ cooking) = {S1, S2, S4}
   - S (EG ¬ cooking) = {S1, S2, S4}
   - S (close i start i EG ¬ cooking) = {}
   - S (EF (close i start i EG ¬ cooking) = {}
   - S (¬ (EF (close i start v EG ¬ cooking)) = {S1, S2, S3, S4}
3. Analiza wyniku

Weryfikacja modelowa w praktyce



- Problem **wykładniczej** w stosunku do liczby procesów eksplozji liczby stanów!

Eksplozja stanów

Metody radzenia sobie z tym problemem oparte są o

- Teorię automatów
  - On-the-fly model checking
  - Partial-order reduction
- Symbolicznej strukturze
- Innych metodach
  - Równoważność
  - Symulacja, bisymulacja
  - Indukcja
  - Symetria
  - ...

Symboliczna weryfikacja modelowa

- Wykorzystanie OBDD do reprezentacji grafu przejść między stanami.
- Stan systemu reprezentowany jako wektor v, gdzie jeden element odpowiada stanowi jednego elementu systemu.
- Relacja przejścia R(v, v’), gdzie v – aktualny stan, v’ – następny stan.
- Formuła R(v, v’) konwertowana jest do postaci OBDD.

Symboliczna weryfikacja modelowa

- Algorytm weryfikacji symbolicznej implementowany przy pomocy procedury *Check,* która bierze formułę CTL wraz z argumentami, a zwraca OBDD, które reprezentuje te stany, w których zachodzi dana formuła.
- 10^5 – 10^20 – 10^120 stanów
- SMV

Rozwiązanie jako punkt stały

- Każda formuła f interpretowana jako zbiór stanów, w których zachodzi f.
  - P(S) – zbiór podzbiorów S (krata)
  - Najmniejszy element – zbiór pusty (false), największy – zbiór wszystkich stanów (true)
  - Funkcjonał $\tau(Y)$ – formuła z dokładnie jednym nieustaloną literą Y
  - $\tau(Y)$ definiuje funkcje $P(S) \rightarrow P(S)$, która dla skończonego zbioru S posiada punk stały

$$\tau(P) = P$$

- Posiadając charakterystykę punktu stałego w CTL jesteśmy w stanie wykorzystać ogólny algorytm w celu obliczenia tego punktu.

Trendy

- Relatively straightforward extensions of current systems
- Require more theoretical work
- Use combination of the abstraction and compositional reasoning techniques
- Probabilistic verification
- The ability to reason automatically about entire families of finite-state systems
- Investigation Model Checking techniques combined with theorem proving

Weryfikacja modelowa - zalety

- W pełni automatyczna:
  - tworzymy model
  - formułujemy wymagania
  - weryfikujemy
- Kontrprzykład, gdy odpowiedź negatywna

Weryfikacja modelowa - wady

- Złożoność obliczeniowa, eksplozja stanów
- Nie weryfikacja, tylko tropienie błędów
  - weryfikujemy model a nie sam system
  - sprawdzamy tylko wymaganie f, innych nie
  - błędy w narzędziach
- Wykonanie abstrakcji wymaga pracy eksperta

Dziedziny zastosowań

- sprzęt (SMV, NuSMV, Murphi)
- protokoły, oprogramowanie systemowe, sterowniki (Spin)
- oprogramowanie (BLAST, PathFinder, CBMC)
- systemy zależne od czasu (Uppaal, Kronos)
- systemy stochastyczne (PRISM)

Bibliografia

- http://www.cs.cmu.edu/~emc/papers.htm
- http://www.kenmcmil.com
- http://www.embedded.com/showArticle.jhtml?articleID=17603352
- http://www.mimuw.edu.pl/~sl/teaching/PMW
- http://courses.projects.cis.ksu.edu/index.php?option=com_content&task=section&id=1&Itemid=2