IV113 Validace a verifikace

Ověřování modelu pro Lineární Temporální Logiku

Jiří Barnat

Motivace

Ověřování kvality

- Testování je neúplné, nedává garanci správnosti.
- Deduktivní verifikace je drahá.

Typický kontext výskytu chyb

- Neočekávané či krajní vstupy.
- Interakce komponent.
- Paralelizmus (nelze testovat).

Model Checking – Ověřování modelu

- Automatizovaný proces verifikace pro ...
- ... paralelní a distribuované systémy.

Sekce

Verifikace paralelních a reaktivních programů

Paralelní programy

Paralelní kompozice

- Komponenty souběžně přispívají k transformaci výchozího stavu na cílový.
- Významová funkce pro paralelně běžící programy vznikne libovolným proložením atomických akcí jednotlivých komponent. (Interleaving.)

Nekompozicionalita významových funkcí

- Významovou funkci paralelního programu nelze získat jako složení významových funkcí jednotlivých komponent.
- Výsledek může být závislý na proložení akcí.

Příklad nekompozicionality

Příklad

- Systém: (y=x; y++; x=y) $\|$ (y=x; y++; x=y)
- Vstup-výstupní proměnná x
- Významová funkce obou procesů je λx->x+1.
- Složení významových funkcí: $(\lambda$x->x+1$)\cdot(\lambda$x->x+1$)$.
- $(\lambda$x->x+1$)\cdot(\lambda$x->x+1$)\ 0 = 2$

2 konkrétní běhy

- Stav $= (x, y_1, y_2)$
- (0,-,-) $\xrightarrow{y_1=x}$ (0,0,-) $\xrightarrow{y_2=x}$ (0,0,0) $\xrightarrow{y_1++}$ $\xrightarrow{x=y_1}$ (1,1,0) $\xrightarrow{y_2++}$ $\xrightarrow{x=y_2}$ (**1**,1,1)
- (0,-,-) $\xrightarrow{y_1=x}$ (0,0,-) $\xrightarrow{y_1++}$ $\xrightarrow{x=y_1}$ (1,1,-) $\xrightarrow{y_2=x}$ (1,1,1) $\xrightarrow{y_2++}$ $\xrightarrow{x=y_2}$ (**2**,1,2)

Vlastnosti paralelních programů

Pozorování

- Konkrétní časování událostí souvisejících s interakcí programů je forma vstupu.
- Paralelní programy jsou svým způsobem reaktivní systémy, neboť nejsou dopředu známa všechna vstupní data.

Důsledek

- U paralelních a reaktivních programů často požadujeme (specifikujeme) chování, která se nedají vyjádřit s použitím vstupních a výstupních podmínek.

Vlastností paralelních/reaktivních programů

Příklady specifikace

- Události A a B nastanou dříve, než událost C.
- Uživateli program není dovoleno vložit novou hodnotu vstupu, dokud program přechozí hodnotu nezpracuje.
- Není pravda, že procedura X bude souběžně vykonávána procesem P i Q (vzájemné vyloučení).
- Každá akce A vyvolá sekvenci reakcí B,C a D.

Formalizace slovního popisu

- Lze formalizovat pomocí temporálních logik.
- Amir Pnueli, 1977

Dokazování pomocí formulí temporálních logik

Pozorování

- Pro modální logiky je možné vystavět podobné důkazové systémy, jako pro Hoarovu logiku.
- Tyto důkazové techniky vykazují stejné/podobné nevýhody jako verifikace paralelních programů s využitím vstupních a výstupních podmínek.

Model checking

- Jiná metoda ověření platnosti formule temporální logiky.

Ověřování modelu (Model Checking)

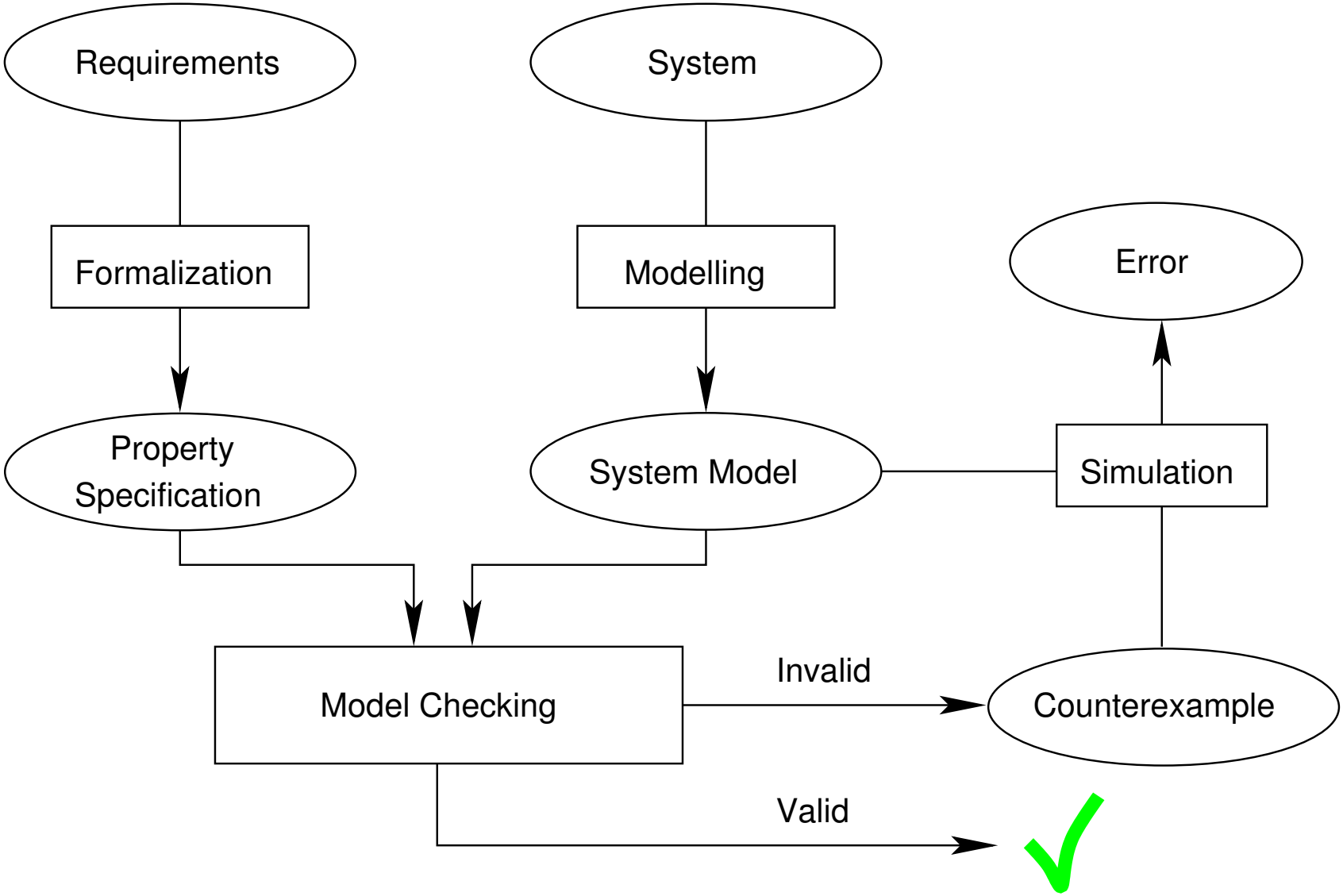
Ověřování modelu – přehled

- Vytvoříme formální model $\mathcal{M}$ verifikovaného systému.
- Specifikaci vyjádříme formulí φ zvolené temporální logiky.
- Rozhodneme, zda $\mathcal{M} \models \varphi$. Tj., zda $\mathcal{M}$ je modelem formule φ. (Odtud název ověřování modelu.)

Volitelné

- Postranním produktem rozhodnutí může být (případně větvící se) posloupnost stavů, dokládající rozhodnutí. Tato posloupnost se běžně označuje slovem protipříklad (často produkována pouze za účelem ukázání neplatnosti φ).
- Proces rozhodnutí lze pro konečné (a některé nekonečné) modely systému automatizovat.

Ověřování modelu – schéma



Automatické nástroje pro ověřování modelu

Model-checkery

- Softwarové nástroje, které pro model systému a temporální vlastnost provedou rozhodnutí o splnění dané vlastnosti modelem.
- SPIN, UppAll, SMV, Prism, DIVINE ...

Modelovací jazyky

- Procesy popsány jako rozšířené konečné automaty.
- Rozšíření umožňuje podmínit provedení přechodu/akce platností boolovského výrazu, případně synchronizací s akcí jiného souběžně běžícího procesu.

Modelování a formalizace verifikovaného systému

Formalizace stavového prostoru – Atomické propozice

Připomenutí

- Na systém lze nahlížet jako na množinu stavů, které se mění vlivem vykonávání akcí programu.
- Stav = valuace modelovaných proměnných.

Atomické propozice

- Základní tvrzení vyjadřujících se o kvalitách jednotlivých stavů. (Např. $max(x, y) \geq 3, \ldots$)
- Platnost atomických propozic musí být algoritmicky rozhodnutelná na základě daného stavu, tj. s využitím valuace modelovaných proměnných.
- Množina základních o stavu pozorovatelných jevů je ovlivněna mírou abstrakce použitou při modelování systému.

Formalizace stavového prostoru – Kripkeho struktury

Kripkeho struktury

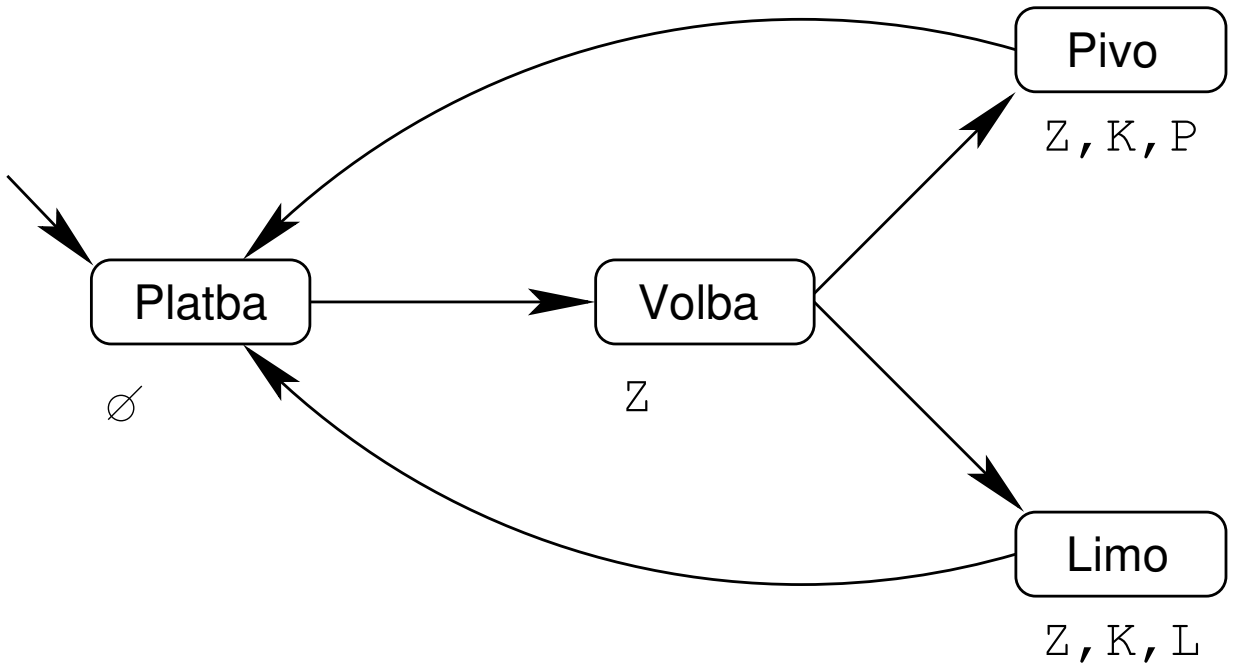
- Nechť je dána množina atomických propozic AP.
- Kripkeho struktura je čtveřice (S, T, I, s_0), kde
  - S je (konečná) množina stavů,
  - $T \subseteq S \times S$ je přechodová relace,
  - $I : S \to 2^{AP}$ je interpretace AP.
  - $s_0 \in S$ je počáteční stav.

Kripkeho přechodové systémy

- Je-li dána množina Act akcí proveditelných programem, je možné Kripkeho struktury rozšířit o označení přechodů.
- Kripkeho přechodový systém je pětice $(S, T, I, s_0, \mathcal{L})$, kde
  - (S, T, I, s_0) je Kripkeho struktura,
  - $\mathcal{L} : T \to Act$ je značkovací funkce.

Kripkeho struktura – příklad

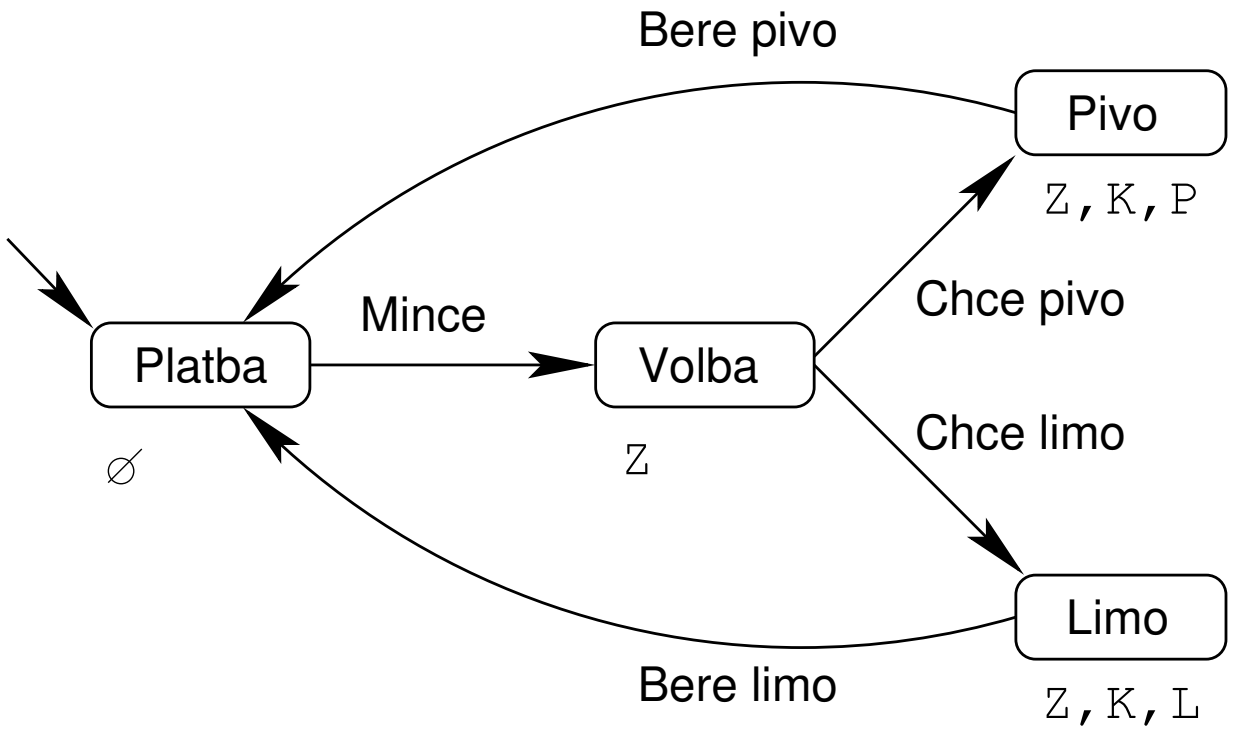
Kripkeho struktura



AP=$\{$Z – zaplaceno, K – kelímek, L – limo, P – pivo$\}$

Kripkeho struktura – příklad

Kripkeho přechodový systém



AP={Z – zaplaceno, K – kelímek, L – limo, P – pivo}

Formalizace stavového prostoru – běh systému

Běh

- Maximální (tj. nerozšiřitelný) sled v grafu indukovaného Kripkeho strukturou, počínající v iniciálním stavu Kripkeho struktury.
- Nechť $M = (S, T, I, s_0)$ je Kripkeho struktura.
- Běh je posloupnost stavů $\pi = s_0, s_1, s_2, \ldots$ taková, že $\forall i \in \mathbb{N}_0.(s_i, s_{i+1}) \in T$.

Konečné běhy – úmluva

- Maximální sled může být konečný: $\pi = s_0, s_1, s_2, \ldots, s_k$, pokud $\nexists s_{k+1} \in S.(s_k, s_{k+1}) \in T$.
- Z technických důvodu lze na konečné maximální sledy nahlížet jako na nekonečné běhy, které vzniknou opakováním posledního stavu daného maximálního sledu.
- Maximální sled $s_0, \ldots, s_k$ se chápe jako běh $s_0, \ldots, s_k, s_k, s_k, \ldots$.

Implicitní a explicitní popis systému

Pozorování

- Kripkeho struktura, jež udává konkrétní chování systému, se často nepopisuje výčtem stavů a hran (explicitní forma), ale pouze programem (implicitní forma).
- Implicitní zápis může být exponenciálně úspornější.

Generování stavového prostoru

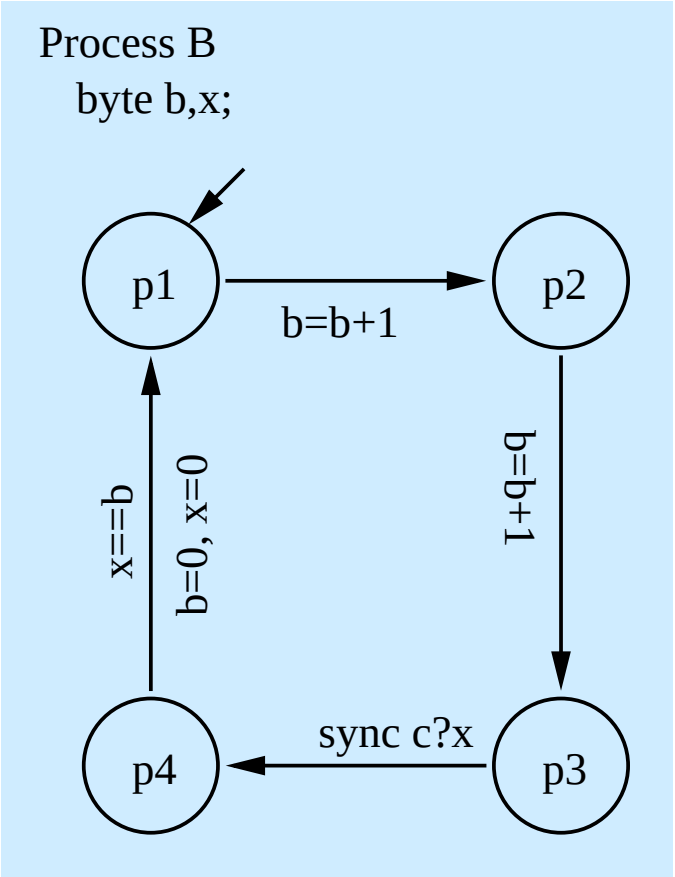
- Výpočet explicitní reprezentace z implicitního popisu.
- Interpretace implicitního zápisu sleduje přesně formálně definovaná pravidla.

Praxe

- Programovací jazyky nemají přesnou sémantiku.
- Využívají se umělé modelovací jazyky.

Příklad modelovacího jazyka – DVE

- Konečný automat
  - Stavy (Lokace)
  - Iniciální stav
  - Přechody
  - (Akceptující stavy)
- Přechody rozšířené o
  - Stráž
  - Synchronizaci s předáváním hodnot
  - Efekt (přiřazení)
- Lokální proměnné
  - `integer`, `byte`
  - `channel`



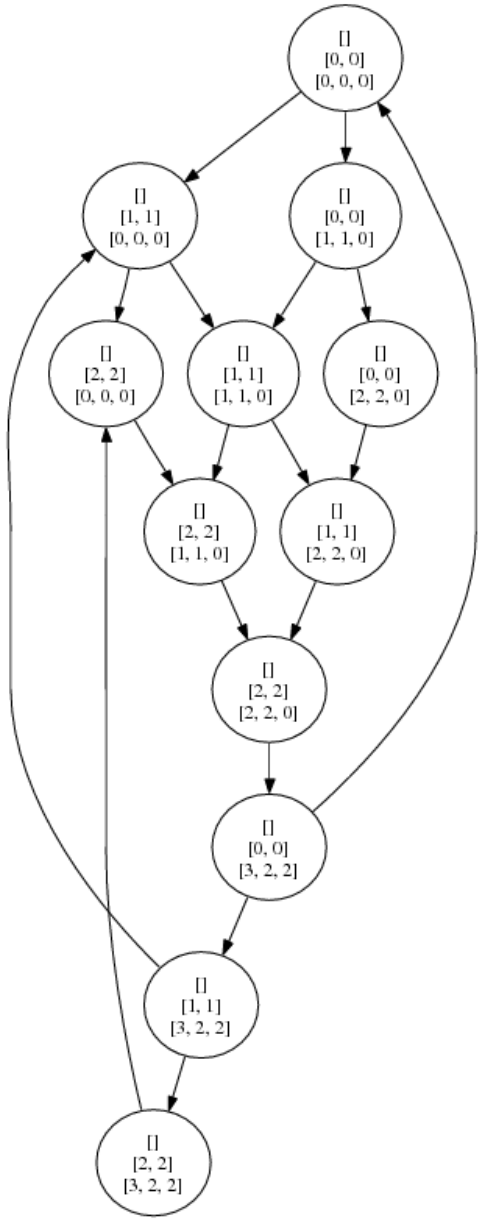
Příklad modelu v jazyce DVE

```
channel {byte} c[0];

process A {
byte a;
state q1,q2,q3;
init q1;
trans
q1→q2 { effect a=a+1; },
q2→q3 { effect a=a+1; },
q3→q1 { sync c!a; effect a=0; };
}

process B {
byte b,x;
state p1,p2,p3,p4;
init p1;
trans
p1→p2 { effect b=b+1; },
p2→p3 { effect b=b+1; },
p3→p4 { sync c?x; },
p4→p1 { guard x==b; effect b=0, x=0; };
}

system async;
```



Sémantika ukázána simulací

State: []; A:[q1, a:0]; B:[p1, b:0, x:0]
0 ⟨0.0⟩: q1 → q2 { effect a = a+1; }
1 ⟨1.0⟩: p1 → p2 { effect b = b+1; }
Command:1

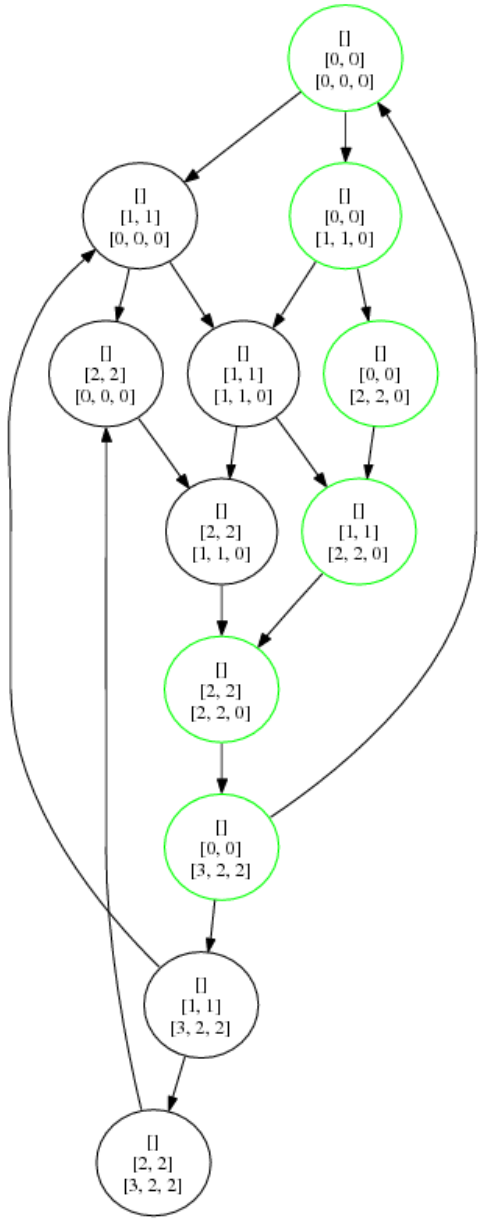
State: []; A:[q1, a:0]; B:[p2, b:1, x:0]
0 ⟨0.0⟩: q1 → q2 { effect a = a+1; }
1 ⟨1.1⟩: p2 → p3 { effect b = b+1; }
Command:1

State: []; A:[q1, a:0]; B:[p3, b:2, x:0]
0 ⟨0.0⟩: q1 → q2 { effect a = a+1; }
Command:0

State: []; A:[q2, a:1]; B:[p3, b:2, x:0]
0 ⟨0.1⟩: q2 → q3 { effect a = a+1; }
Command:0

State: []; A:[q3, a:2]; B:[p3, b:2, x:0]
0 ⟨0.2&1.2⟩: q3 → q1 { sync c!a; effect a = 0; }
p3 → p4 { sync c?x; }
Command:0

State: []; A:[q1, a:0]; B:[p4, b:2, x:2]



Formalizace vlastností

Specifikace jako jazyky nekonečných slov

Problém

- Jak se formálně vyjadřovat o kvalitách běhu?
- Jak mechanicky rozhodovat, že běh má danou kvalitu?

Řešení

- Konečný automat jako mechanický pozorovatel běhu.
- Běh je nekonečný!
- Konečné automaty pro jazyky nekonečných slov (ω-regulární jazyky)
- Büchi akceptační podmínka – automat akceptuje slovo pokud nekonečněkrát projde koncovým stavem.

Automaty nad nekonečnými slovy

Büchi automaty

- Büchi automat je pětice $A = (\Sigma, S, s, \delta, F)$, kde
  - Σ je konečná abeceda znaků,
  - S je konečná množina stavů,
  - $s \in S$ je iniciální stav,
  - $\delta : S \times \Sigma \to 2^S$ je přechodová relace, a
  - $F \subseteq S$ je množina koncových stavů.

Jazyk akceptovaný Büchi automatem A

- Běh ρ automatu A nad nekonečným slovem $w = a_1 a_2 \ldots$ je sekvence stavů $\rho = s_0, s_1, \ldots$ taková, že $s_0 \equiv s$ a $\forall i : s_i \in \delta(s_{i-1}, a_i)$.
- $inf(\rho)$ – množinu stavů, které se v ρ vyskytly nekonečně krát.
- Běh ρ je akceptující, pokud $inf(\rho) \cap F \neq \emptyset$.
- Jazyk akceptovaný automatem A je množina všech slov, pro které existuje akceptující běh. Označujeme $L(A)$.

Přehlednost zápisu/nákresu

Pozorování

- AP={X,Y,Z},
- Hrana označená $\{X\}$ značí, že platí X a neplatí Y a Z.
- Pokud je třeba vyjádřit že platí X, neplatí Z a na platnosti Y nezáleží, je třeba vést hrany pod $\{X\}$ a $\{X, Y\}$.

Množiny AP jako boolovské formule

- Jednotlivé hrany mezi dvěma stejnými vrcholy pod různými kombinacemi atomických propozic lze sloučit do jedné hrany ohodnocené boolovskou formulí.

Příklad

- Hrany {X}, {Y}, {X,Y}, {X,Z}, {Y,Z} a {X,Y,Z} lze nahradit jednou hranou ohodnocenou formulí $X \vee Y$.
- Pokud vůbec nezáleží na platnosti při provedení hrany, je možné ji označit štítkem $\mathit{true} \equiv X \vee \neg X$.

Vyjádřete Büchi automatem

Systém

- Prodejní automat
- $\Sigma = 2^{\{Z,K,L,P\}}$,
- $jeZ = \{A \in \Sigma \mid Z \in A\}$, $jeK = \{A \in \Sigma \mid K \in A\}$, ...

Vlastnosti

- Prodejní automat vydá alespoň jeden kelímek s nápojem.
- Prodejní automat vydá alespoň jeden kelímek s limonádou.

- Prodejní automat vydá nekonečně mnoho kelímků.
- Prodejní automat vydá nekonečně mnoho kelímků s pivem.

- Prodejní automat nevydá kelímek s nápojem, bez zaplacení.
- Pokaždé, když zaplatím, dostanu kelímek s nápojem.

Lineární temporální logika

LTL neformálně

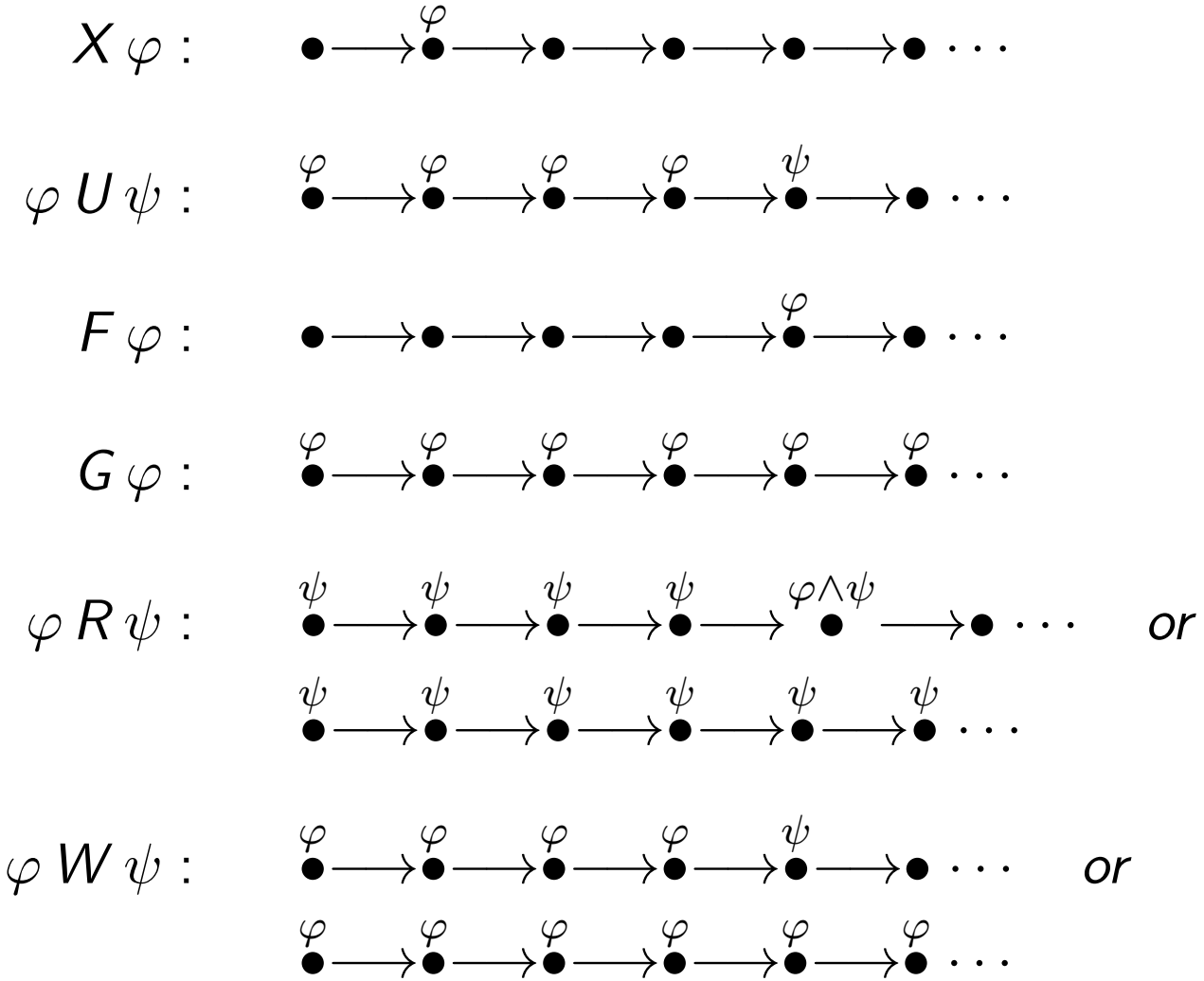
Formule φ

- Vyhodnocuje se nad jedním během systému.
- Vyjadřuje se o platnosti atomických propozic ve stavech daného běhu.

Temporální operátory LTL

- $F\,\varphi$ — (Future) Někde v běhu platí φ.
- $G\,\varphi$ — (Globally) V daném běhu vždy platí φ.
- $\varphi\,U\,\psi$ — (Until) Někde platí ψ a do té doby platí φ.
- $X\,\varphi$ — (Next) V příštím stavu platí φ.
- $\varphi\,W\,\psi$ — (Weak Until) Jako Until ale ψ nemusí nastat.
- $\varphi\,R\,\psi$ — (Release) ψ platí dokud neplatí $\varphi \wedge \psi$.

Grafické znázornění sémantiky temporálních operátorů



Syntax LTL

Nechť AP je množina atomických propozic. Pak

- Je-li $p \in AP$, pak p je formule.
- Je-li φ formule, pak $\neg\varphi$ je formule.
- Jsou-li φ a ψ formule, pak $\varphi \vee \psi$ je formule.
- Je-li φ formule, pak $X\,\varphi$ je formule.
- Jsou-li φ a ψ formule, pak $\varphi\,U\,\psi$ je formule.

Alternativní zápis

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\,\varphi \mid \varphi\,U\,\varphi$$

Syntaktické zkratky

Výroková logika

- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$
- $\varphi \Leftrightarrow \psi \equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$

Temporální operátory

- $F\,\varphi \equiv \mathit{true}\,U\,\varphi$
- $G\,\varphi \equiv \neg F\,\neg\varphi$
- $\varphi\,R\,\psi \equiv \neg(\neg\varphi\,U\,\neg\psi)$
- $\varphi\,W\,\psi \equiv \varphi\,U\,\psi \vee G\,\varphi$

Alternativní syntax

- $F\varphi \equiv \diamond\varphi$
- $G\varphi \equiv \Box\varphi$
- $X\varphi \equiv \circ\varphi$

Modely formulí LTL

Model LTL formule

- Je dána množina atomických propozic AP.
- Modelem LTL formule je běh π Kripkeho struktury.

Značení

- Nechť $\pi = s_0, s_1, s_2, \ldots$.
- Suffix běhu π počínaje stavem s_k budeme značit jako $\pi^k = s_k, s_{k+1}, s_{k+2}, \ldots$.
- K-tý stav běhu π, budeme značit jako $\pi(k) = s_k$.

Sémantika LTL

Předpoklady

- Je dána množina atomických propozic AP.
- Je dán běh π Kripkeho struktury $M = (S, T, I, s_0)$.
- φ, ψ jsou syntakticky správné LTL formule.
- $p \in AP$ je atomická propozice.

Sémantika

$$
\begin{array}{rcl}
\pi \models p & \text{iff} & p \in I(\pi(0)) \\
\pi \models \neg\varphi & \text{iff} & \pi \not\models \varphi \\
\pi \models \varphi \vee \psi & \text{iff} & \pi \models \varphi \text{ or } \pi \models \psi \\
\pi \models X\,\varphi & \text{iff} & \pi^1 \models \varphi \\
\pi \models \varphi\, U\, \psi & \text{iff} & \exists k.0 \leq k, \pi^k \models \psi \text{ and} \\
 & & \forall i.0 \leq i < k, \pi^i \models \varphi
\end{array}
$$

Sémantika temporálních operátorů

$$\pi \models F\,\varphi \quad \text{iff} \quad \exists k.k \geq 0, \pi^k \models \varphi$$

$$\pi \models G\,\varphi \quad \text{iff} \quad \forall k.k \geq 0, \pi^k \models \varphi$$

$$\begin{aligned}\pi \models \varphi\,R\,\psi \quad \text{iff} \quad & (\exists k.0 \leq k, \pi^k \models \varphi \wedge \psi \text{ and} \\ & \forall i.0 \leq i < k, \pi^i \models \psi) \\ & \text{or } (\forall k.k \geq 0, \pi^k \models \psi)\end{aligned}$$

$$\begin{aligned}\pi \models \varphi\,W\,\psi \quad \text{iff} \quad & (\exists k.0 \leq k, \pi^k \models \psi \text{ and} \\ & \forall i.0 \leq i < k, \pi^i \models \varphi) \\ & \text{or } (\forall k.k \geq 0, \pi^k \models \varphi)\end{aligned}$$

LTL Model checking

Verifikace s použitím LTL

- Na systém nahlížíme jako na množinu možných běhů.
- Systém splňuje vlastnost specifikovanou LTL formulí pokud (a jen tehdy) všechny běhy systému začínající v iniciálním stavu systému splňují danou formuli.
- Kterýkoliv běh systému, který začíná v iniciálním stavu a nesplňuje danou formuli může být dokladem toho, že systém nesplňuje specifikovanou vlastnost.

Tvrzení

- Pokud konečně stavový systém nesplňuje vlastnost specifikovanou formulí LTL, tak to lze dokladovat během π, který lze vyjádřit jako $\pi = \pi_1 \cdot (\pi_2)^\omega$, kde

  $\pi_1 = s_0, s_1, \ldots, s_k$

  $\pi_2 = s_{k+1}, s_{k+2}, \ldots, s_{k+n}$, kde $s_k \equiv s_{k+n}$.

- Běhy s uvedenou vlastností nazýváme *lasa* (lasso shape).

LTL Model Checking s využitím teorie formálních jazyků a automatů

Spojitost s jazyky nekonečných slov

Pozorování 1

- Systém je množina (nekonečných) běhů.
- Na systém lze nahlížet jako na jazyk nekonečných slov.

Pozorování 2

- Dva různé běhy (různé posloupnosti stavů) jsou z hlediska platnosti dané formule ekvivalentní, pokud se shodují v interpretaci atomických proměnných.
- Jestliže $\pi = s_0, s_1, \ldots,$ pak $I(\pi) \stackrel{def}{\Longleftrightarrow} I(s_0), I(s_1), I(s_2), \ldots$.

Pozorování 3

- Každý běh danou formuli buď splňuje, anebo nesplňuje.
- Každá LTL formule vymezuje množinu splňujících běhů.

Redukce problému na jazykovou inkluzi

Problém

- Je dána množina AP, Kripkeho struktura $M = (S, T, I, s_0)$ a specifikace LTL formulí φ.
- Splňuje systém M specifikaci φ? $(M \models \varphi)$

Jazyky nekonečných slov

- Nechť $\Sigma = 2^{AP}$.
- Jazyk L_{sys} všech běhů systému M:

$$L_{sys} = \{I(\pi) \mid \pi \text{ je běh v } M\}.$$

- Jazyk L_φ všech běhů splňující formuli φ:

$$L_\varphi = \{I(\pi) \mid \pi \models \varphi\}.$$

Pozorování

- Systém M splňuje specifikaci φ právě když $L_{sys} \subseteq L_\varphi$.

L_{sys} a L_φ vyjádřeny Büchi automatem

Tvrzení 1

- Pro každou LTL formuli φ existuje (a lze efektivně sestrojit) Büchi automat A_φ takový, že $L_\varphi = L(A_\varphi)$.
- [Vardi, Wolper 1986]

Tvrzení 2

- Pro každou Kripkeho strukturu $M = (S, T, I, s_0)$ lze sestrojit Büchi automat A_{sys} takový, že $L_{sys} = L(A_{sys})$.
- Konstrukce A_{sys}
  - Nechť AP je množina atomických propozic.
  - Pak $A_{sys} = (S, 2^{AP}, s_0, \delta, S)$, kde
  - $q \in \delta(p, a)$ právě když $(p, q) \in T \wedge I(p) = a$.

Synchronní kompozice Büchi automatů – zjednodušená

Tvrzení

- Nechť $A = (S_A, \Sigma, s_A, \delta_A, F_A)$ a $B = (S_B, \Sigma, s_B, \delta_B, F_B)$ jsou Büchi automaty nad shodnou abecedou Σ. Pak existuje (lze efektivně sestrojit) Büchi automat $A \times B$ takový, že $L(A \times B) = L(A) \cap L(B)$.

Pozorování

- Büchi automat A_{sys} je zkonstruován tak, že $F_A = S_A$, tj. má všechny stavy akceptující.

Konstrukce $A \times B$ pro případ, že $F_A = S_A$

- $A \times B = (S_A \times S_B, \Sigma, (s_A, s_B), \delta_{A\times B}, S_A \times F_B)$
- $(p', q') \in \delta_{A\times B}((p, q), a)$ pro všechna
  $p' \in \delta_A(p, a)$
  $q' \in \delta_B(q, a)$
- V plné obecnosti je konstrukce $A \times B$ složitější.

Redukce na problém prázdnosti Büchi automatu

Tvrzení

- Pro každou LTL formuli φ platí: $co{-}L(A_\varphi) = L(A_{\neg\varphi})$.

Redukce $M \models \varphi$ na problém prázdnosti $L(A_{sys} \times A_{\neg\varphi})$

- $M \models \varphi \iff L_{sys} \subseteq L_\varphi$
- $M \models \varphi \iff L(A_{sys}) \subseteq L(A_\varphi)$
- $M \models \varphi \iff L(A_{sys}) \cap co{-}L(A_\varphi) = \emptyset$
- $M \models \varphi \iff L(A_{sys}) \cap L(A_{\neg\varphi}) = \emptyset$
- $M \models \varphi \iff L(A_{sys} \times A_{\neg\varphi}) = \emptyset$

Redukce problému na detekci akceptujícího cyklu

Tvrzení

- Büchi automat $A = (S, \Sigma, s_0, \delta, F)$ akceptuje neprázdný jazyk právě když existují stav $s \in F$ a slova $w_1, w_2 \in \Sigma^*$ taková, že $s \in \hat{\delta}(s_0, w_1)$ a $s \in \hat{\delta}(s, w_2)$.
- Tj. graf Büchi automatu obsahuje dosažitelný akceptující cyklus (cyklus přes nějaký akceptující stav).

Rozhodovací procedura pro problém $M \models \varphi$?

- Zkonstruuje se produktový automat $(A_{sys} \times A_{\neg\varphi})$.
- Graf produktového automatu se prověří na přítomnost akceptujících cyklů.
- Obsahuje-li graf akceptující cyklus, pak $M \not\models \varphi$.
- Neobsahuje-li graf akceptující cyklus, pak $M \models \varphi$.