CCS s předáváním hodnot

Hlavní myšlenka

„Handshake“ synchronizace je rozšířena o možnost výměny celočíselných hodnot.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

CCS s předáváním hodnot

Hlavní myšlenka

„Handshake“ synchronizace je rozšířena o možnost výměny celočíselných hodnot.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

CCS s předáváním hodnot

Hlavní myšlenka

„Handshake“ synchronizace je rozšířena o možnost výměny celočíselných hodnot.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

Parametrizované procesní konstanty

Např.: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

CCS s předáváním hodnot

Hlavní myšlenka

„Handshake“ synchronizace je rozšířena o možnost výměny celočíselných hodnot.

$$\overline{pay(6)}.Nil \mid pay(x).\overline{save(x/2)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid \overline{save(3)}.Nil \mid Bank(100)$$

$$\downarrow \tau$$

$$Nil \mid Nil \mid Bank(103)$$

Parametrizované procesní konstanty

Např.: $Bank(total) \stackrel{\text{def}}{=} save(x).Bank(total + x)$.

Překlad CCS s předáváním hodnot na standardní CCS

CCS s předáváním hodnot

$$C \stackrel{\text{def}}{=} in(x).C'(x)$$

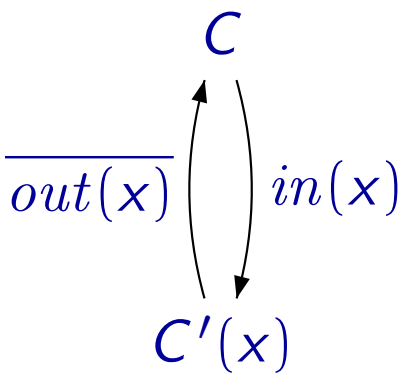
$$C'(x) \stackrel{\text{def}}{=} \overline{out(x)}.C$$

$\longrightarrow$

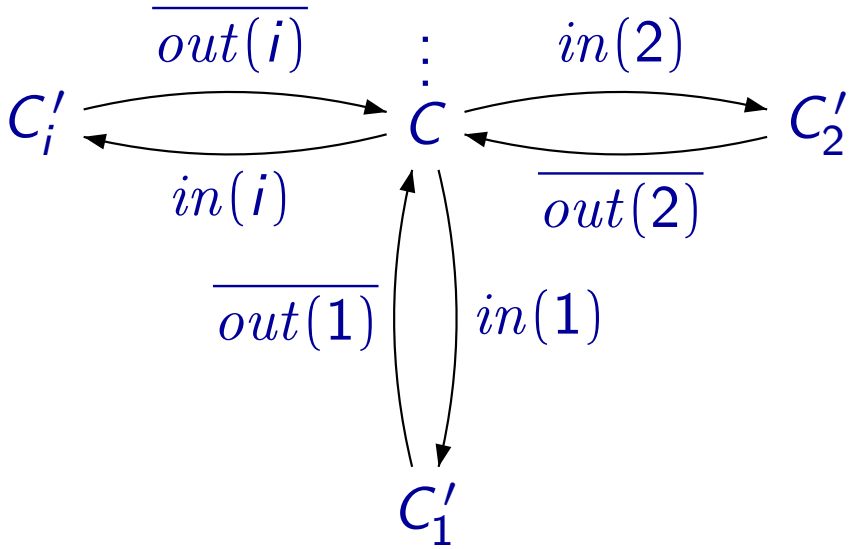
Standardní CCS

$$C \stackrel{\text{def}}{=} \sum_{i \in \mathbb{N}} in(i).C'_i$$

$$C'_i \stackrel{\text{def}}{=} \overline{out(i)}.C$$



$\longrightarrow$



Symbolický CCS

Nekonečný CCS

CCS je turingovsky úplný

Fakt

CCS může simulovat výpočet libovolného Turingova stroje.

Poznámka

- Vyjadřovací síla CCS je stejná jako vyjadřovací síla libovolného programovacího jazyka.
- Jeho účel je spíš popis chování reaktivních systémů než vykonávání konkrétních výpočtů.

Ekvivalence chování

Implementace

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (CM \mid CS) \setminus \{coin, coffee\}$$

Ekvivalence chování

Implementace

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (CM \mid CS) \setminus \{coin, coffee\}$$

Specifikace

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Ekvivalence chování

Implementace

$$CM \stackrel{\text{def}}{=} coin.\overline{coffee}.CM$$

$$CS \stackrel{\text{def}}{=} \overline{pub}.\overline{coin}.coffee.CS$$

$$Uni \stackrel{\text{def}}{=} (CM \mid CS) \setminus \{coin, coffee\}$$

Specifikace

$$Spec \stackrel{\text{def}}{=} \overline{pub}.Spec$$

Otázka

Chovají se procesy Uni a $Spec$ „stejně“? (Je jejich chování ekvivalentní?)

$$Uni \equiv Spec$$

Cíle

Co by měla rozumná ekvivalence chování splňovat?

- abstrahovat od stavů (brát do úvahy jen chování – akce)
- abstrahovat od nedeterminismu
- abstrahovat od vnitřního chování

Cíle

Co by měla rozumná ekvivalence chování splňovat?

- abstrahovat od stavů (brát do úvahy jen chování – akce)
- abstrahovat od nedeterminismu
- abstrahovat od vnitřního chování

Co ještě by měla ekvivalence chování splňovat?

- reflexivita $P \equiv P$ pro každý proces P
- tranzitivita Z $Spec_0 \equiv Spec_1 \equiv Spec_2 \equiv \cdots \equiv Impl$ plyne
$$Spec_0 \equiv Impl$$
- symetrie $P \equiv Q$ právě tehdy, když $Q \equiv P$

Cíle

Co by měla rozumná ekvivalence chování splňovat?

- abstrahovat od stavů (brát do úvahy jen chování – akce)
- abstrahovat od nedeterminismu
- abstrahovat od vnitřního chování

Co ještě by měla ekvivalence chování splňovat?

- reflexivita $P \equiv P$ pro každý proces P
- tranzitivita Z $Spec_0 \equiv Spec_1 \equiv Spec_2 \equiv \cdots \equiv Impl$ plyne
$$Spec_0 \equiv Impl$$
- symetrie $P \equiv Q$ právě tehdy, když $Q \equiv P$

Kongruence

- Jestliže $C_1 \equiv C_2$, pak $P(C_1) \equiv P(C_2)$
- Změna komponenty C_1 v programu P za ekvivalentní komponentu C_2 by neměla změnit chování celého programu.

Jazyková ekvivalence

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS.

Množina slov pro $s \in Proc$

$$Traces(s) = \{w \in Act^* \mid \exists s' \in Proc : \ s \xrightarrow{w} s'\}$$

Jazyková ekvivalence

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS.

Množina slov pro $s \in Proc$

$$Traces(s) = \{w \in Act^* \mid \exists s' \in Proc : \ s \xrightarrow{w} s'\}$$

Nechť $s \in Proc$ a $t \in Proc$.

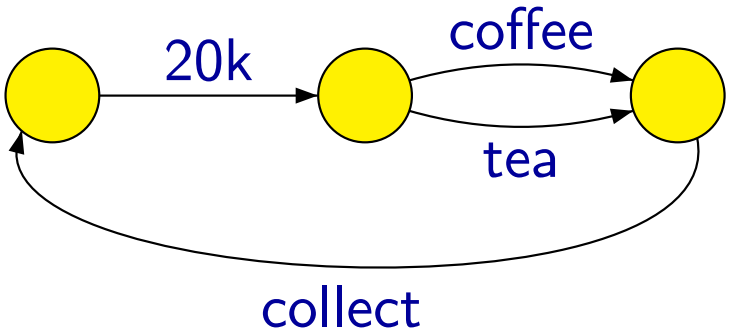
Jazyková ekvivalence (Trace Equivalence)

Řekneme, že s a t jsou jazykově ekvivalentní (trace equivalent) ($s \equiv_t t$) právě tehdy, když

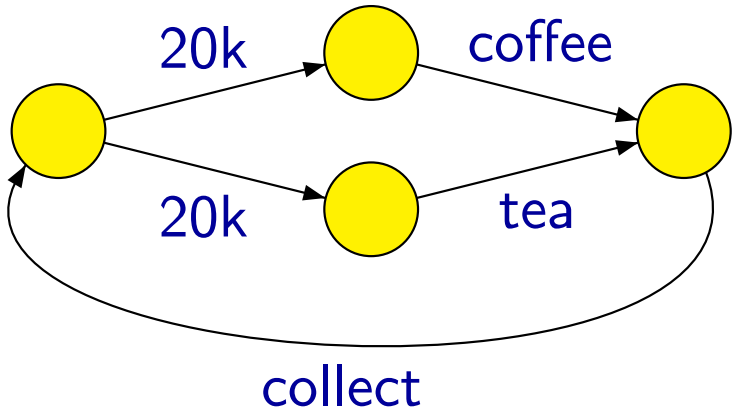
$$Traces(s) = Traces(t)$$

Je to „dobrá“ ekvivalence chování?

Black-box experimenty



- Vhodím minci
- Mám na výběr mezi stiskem tlačítek „coffee“ a „tea“



- Vhodím minci
- Někdy jde stisknout jen „coffee“ a v ostatních případech zase jen „tea“

Hlavní myšlenka

Dva procesy považujeme za ekvivalentní z hlediska chování právě tehdy, když je vnější pozorovatel nedokáže odlišit.

Silná bisimilarita

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS.

Silná bisimulace (Strong Bisimulation)

Binární relace $R \subseteq Proc \times Proc$ je silná bisimulace (strong bisimulation) právě tehdy, když kdykoliv $(s, t) \in R$ potom pro každé $a \in Act$:

- jestliže $s \xrightarrow{a} s'$, pak $t \xrightarrow{a} t'$ pro nějaké t' takové, že $(s', t') \in R$
- jestliže $t \xrightarrow{a} t'$, pak $s \xrightarrow{a} s'$ pro nějaké s' takové, že $(s', t') \in R$.

Silná bisimilarita

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS.

Silná bisimulace (Strong Bisimulation)

Binární relace $R \subseteq Proc \times Proc$ je silná bisimulace (strong bisimulation) právě tehdy, když kdykoliv $(s, t) \in R$ potom pro každé $a \in Act$:

- jestliže $s \xrightarrow{a} s'$, pak $t \xrightarrow{a} t'$ pro nějaké t' takové, že $(s', t') \in R$
- jestliže $t \xrightarrow{a} t'$, pak $s \xrightarrow{a} s'$ pro nějaké s' takové, že $(s', t') \in R$.

Silná bisimilarita (Strong Bisimilarity)

Dva procesy $p_1, p_2 \in Proc$ jsou silně bisimulačně ekvivalentní (strongly bisimilar, $p_1 \sim p_2$) právě tehdy, když existuje silná bisimulace R tž. $(p_1, p_2) \in R$.

$$\sim \ = \ \cup\{R \mid R \text{ je silná bisimulace}\}$$

Relaci $\sim$ nazýváme silná bisimulační ekvivalence nebo silná bisimilarita.

Základní vlastnosti silné bisimilarity

Věta

$\sim$ je ekvivalence (reflexivní, symetrická a tranzitivní)

Základní vlastnosti silné bisimilarity

Věta

$\sim$ je ekvivalence (reflexivní, symetrická a tranzitivní)

Věta

$\sim$ je největší bisimulace

Základní vlastnosti silné bisimilarity

Věta

$\sim$ je ekvivalence (reflexivní, symetrická a tranzitivní)

Věta

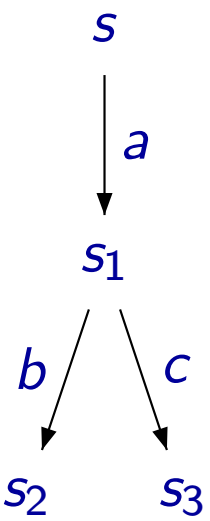
$\sim$ je největší bisimulace

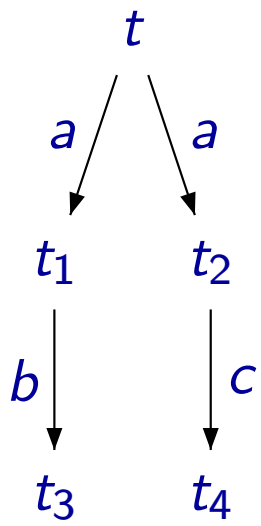
Věta

$s \sim t$ právě tehdy, když pro každé $a \in Act$:

- jestliže $s \xrightarrow{a} s'$, pak $t \xrightarrow{a} t'$ pro nějaké t' takové, že $s' \sim t'$
- jestliže $t \xrightarrow{a} t'$, pak $s \xrightarrow{a} s'$ pro nějaké s' takové, že $s' \sim t'$.

Jak ukázat neekvivalenci?

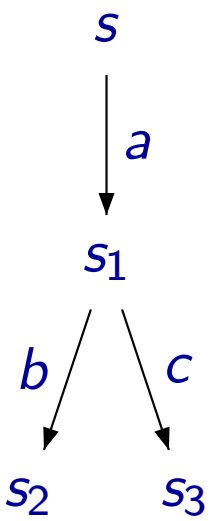


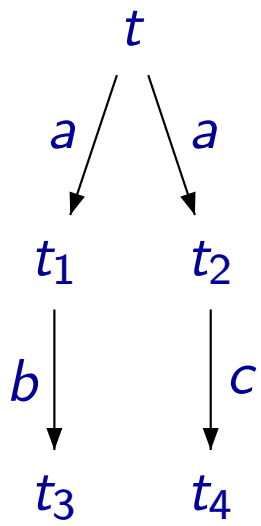


K důkazu $s \not\sim t$:

- Vyjmenovat všechny binární relace a o každé ukázat, že nesplňuje současně to, že by obsahovala (s, t) a byla silnou bisimulací. (Náročné: $2^{|Proc|^2}$ relací.)

Jak ukázat neekvivalenci?

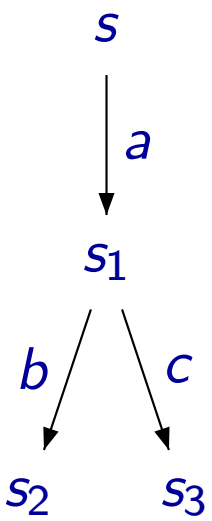


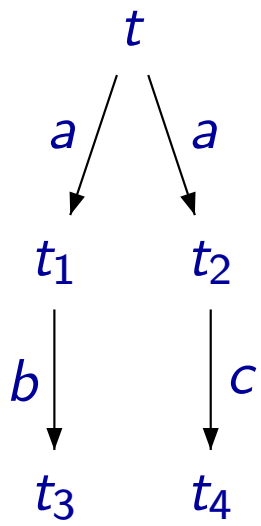


K důkazu $s \not\sim t$:

- Vyjmenovat všechny binární relace a o každé ukázat, že nesplňuje současně to, že by obsahovala (s, t) a byla silnou bisimulací. (Náročné: $2^{|Proc|^2}$ relací.)
- Provést nějaká pozorování, která umožní vyřadit mnoho kandidátů na bisimulaci najednou.

Jak ukázat neekvivalenci?





K důkazu $s \not\sim t$:

- Vyjmenovat všechny binární relace a o každé ukázat, že nesplňuje současně to, že by obsahovala (s, t) a byla silnou bisimulací. (Náročné: $2^{|Proc|^2}$ relací.)
- Provést nějaká pozorování, která umožní vyřadit mnoho kandidátů na bisimulaci najednou.
- Využít herní charakterizaci silné bisimilarity.

Silně bisimulační hra

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS a $s, t \in Proc$.

Definujeme hru dvou hráčů „útočníka“ (attacker) a „obránce“ (defender) začínající z s a t.

- Hra se hraje na kola (rounds) a konfigurace hry jsou dvojice stavů z $Proc \times Proc$.
- V každém kole je právě jedna konfigurace aktuální. Na začátku hry je aktuální konfigurací dvojice (s, t).

Silně bisimulační hra

Nechť $(Proc, Act, \{\xrightarrow{a} | \ a \in Act\})$ je LTS a $s, t \in Proc$.

Definujeme hru dvou hráčů „útočníka“ (attacker) a „obránce“ (defender) začínající z s a t.

- Hra se hraje na kola (rounds) a konfigurace hry jsou dvojice stavů z $Proc \times Proc$.
- V každém kole je právě jedna konfigurace aktuální. Na začátku hry je aktuální konfigurací dvojice (s, t).

Intuice

Obránce chce ukázat, že s a t jsou silně bisimulačně ekvivalentní, zatímco útočník se snaží dokázat opak.

Pravidla bisimulační hry

Pravidla hry

V každém kole hráčí změní aktuální konfiguraci následovně::

1. útočník si vybere jeden z procesů aktuální konfigurace a provede vybraný tah podle relace $\xrightarrow{a}$ pro nějaké $a \in Act$
2. obránce musí odpovědět tahem podle $\xrightarrow{a}$ (se stejnou akcí a) v druhém procesu dvojice z aktuální konfigurace

Těmito dvěma tahy dosažená dvojice procesů se stává novou aktuální konfigurací. Hra pokračuje dalším kolem podle stejných pravidel.

Pravidla bisimulační hry

Pravidla hry

V každém kole hráčí změní aktuální konfiguraci následovně::

1. útočník si vybere jeden z procesů aktuální konfigurace a provede vybraný tah podle relace $\xrightarrow{a}$ pro nějaké $a \in Act$
2. obránce musí odpovědět tahem podle $\xrightarrow{a}$ (se stejnou akcí a) v druhém procesu dvojice z aktuální konfigurace

Těmito dvěma tahy dosažená dvojice procesů se stává novou aktuální konfigurací. Hra pokračuje dalším kolem podle stejných pravidel.

Výsledek hry

- Hráč, který nemá k dispozici žádný tah, prohrává. Jeho soupeř vyhrál.
- Pokud je hra nekonečná, vyhrává obránce.

Herní charakterizace silné bisimilarity

Věta

- Stavy s a t jsou silně bisimulačně ekvivalentní právě tehdy, když obránce má univerzální vítěznou strategii (universal winning strategy) v bisimulační hře z počáteční konfigurace (s, t).
- Stavy s a t nejsou silně bisimulačně ekvivalentní právě tehdy, když útočník má univerzální vítěznou strategii v bisimulační hře z počáteční konfigurace (s, t).

Herní charakterizace silné bisimilarity

Věta

- Stavy s a t jsou silně bisimulačně ekvivalentní právě tehdy, když obránce má univerzální vítěznou strategii (universal winning strategy) v bisimulační hře z počáteční konfigurace (s, t).
- Stavy s a t nejsou silně bisimulačně ekvivalentní právě tehdy, když útočník má univerzální vítěznou strategii v bisimulační hře z počáteční konfigurace (s, t).

Poznámka

Bisimulační hru je možné použít jak pro důkaz bisimilarity dvou procesů tak pro důkaz toho, že procesy bisimulačně ekvivalentní nejsou. Vhodná je především pro negativní případ.

Silná bisimilarita je kongruence pro CCS operace

Věta

Nechť P a Q jsou CCS procesy takové, že $P \sim Q$. Potom

- $\alpha.P \sim \alpha.Q$ pro každou akci $\alpha \in Act$
- $P + R \sim Q + R$ a $R + P \sim R + Q$ pro každý CCS proces R
- $P \mid R \sim Q \mid R$ a $R \mid P \sim R \mid Q$ pro každý CCS proces R
- $P[f] \sim Q[f]$ pro každou funkci přejmenování f
- $P \setminus L \sim Q \setminus L$ pro každou množinu návěští L.

Další vlastnosti silné bisimilarity

Následující vlastnosti platí pro každý CCS proces P, Q a R

- $P + Q \sim Q + P$
- $P \mid Q \sim Q \mid P$
- $P + Nil \sim P$
- $P \mid Nil \sim P$
- $(P + Q) + R \sim P + (Q + R)$
- $(P \mid Q) \mid R \sim P \mid (Q \mid R)$