

Advanced Topics in Concurrent Systems
DM869

Mark Jervelund
Mark@jervelund.com
Doommius.com/notes.php



UNIVERSITY OF
SOUTHERN DENMARK

IMADA

February 26, 2019

Contents

notes	1
Introduction to inference systems	1
Conculus of the cumunication system	2

notes

w.r.t With regards to
s.t. such that

Introduction to inference systems

Rules

Theorem 1. $\frac{}{num(Z)}[Zero]$

Theorem 2. $\frac{num(Z)}{num(Sx)}[Succ]$

num(Z)

Num(Z) is derivable iff x encodes a natural number, if any derivation for number x has exactly height n, then x encodes n proof by induction, on the structure of the given derivation for num(x)

Case Zero The derivation starts with rule[Zero] hence X must be 0, the height must be 1

case one Num(Z) is derivable iff x encodes a natural number, if any derivation for number x has exactly height n+1, then x encodes n proof by induction, on the structure of the given derivation for num(x)

Case Zero The derivation starts with rule[Zero] hence X must be n+1, the height must be n+1

succ

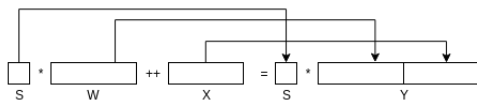
Case Zero

The derivation rule starts with rule[succ] hence X=Sy for some y, we have a derivation for num(Y), Its height, say m.

By induction HP, y encodes m-1 thus Sy, encodes m-1

Add

Theorem 3. $\frac{add(w,x,y)}{add(Sw,X,Sy)}[Succ]$



add(W,X,Y) is derivable iff W+X = Y

Prove by induction on the derivation of add(W,X,Y)

Case +Z There is no inductive step

W = Z, X = x = Y, 0+X=X

Case +S W= Sw, X = x, Y = Sy,

We have a derivation for add(w,x,y) We can apply the inductive hypothesis(ind. HP) w+x=y, W=1+w, Y=1+y, we conclude that 1+w+x=1+y, W+X =Y

sub

Theorem 4. $\frac{num(Z)}{num(Sx)}[Succ]$

sub(w,x,y) def, rules s.t. sub(q,x,y) is derivable iff w-x=y.

It can be proved that there is no proof for this.

if then

Theorem 5. $\frac{if\ x+y=z}{then\ w-x=y} \frac{add(x,y,w)}{sub(w,x,y)}$

Conculus of the cumunication system

C is a channel

C = new channel ("IP eg 10.130.10.42")

C.open(); connect C.send(42);

x: c.recv() P:

Def. a labelled transition sstem is (S, L, \rightarrow)

- S is a set of steates (processes)
- K us a set of lables (Actions)
- $\rightarrow \subseteq S \times L \times S$ is trasition relation

Natation $S \xrightarrow{e} S'$ means (S,e, S') $\in \rightarrow$

$P ::= \emptyset$ //Termination program
 $\overline{c}.P$ //send on channel c and continue as P
 $\overline{c}.P$ // Recieve on channel c and continue as P
 $\frac{}{c.P \xrightarrow{c} P}$ [Send]
 $\frac{}{\overline{c}.P \xrightarrow{\overline{c}} P}$ [Recieve]
 $\frac{P \xrightarrow{c} P' \quad Q \xrightarrow{a} Q'}{P|Q \xrightarrow{c} P'|Q'}$ [Com]

How can i see that two programs are running at the same time.

$P | Q$ // P and Q urn concurrently

$c.P | \overline{c}.Q \rightarrow P | Q$ //If we have two nodes, c, and \overline{c} and c wants to send to \overline{c} and \overline{c} want's to recieve from c, this is synchronys transmission. eg, a communication can't fail.

Spec R spec 1 Secn Rimpl

R should be an equivalence relation between two processes -transitive: reason about chain refinemnt

Reflexivity PRP Symmytry Spec R impl. \rightarrow Impl R spec

Def A cotntect is any term generated $C := [-] \alpha.C | C + P | (P|C)$

$P ::= \alpha.P | O | P + P' | (P|P')$

Def A trace is a sequence $\alpha_1 \dots \alpha_n \in ACT$

(ACT)* is the finite word containing all possible programs

The set of traces of a process P is the set of traces(P) $\{\alpha_1 \dots \alpha_n\}$

$$Traces(User) = \{\epsilon; \overline{P}; \overline{P}.enter; \overline{P}.enter.exit; \dots\} = \{\epsilon\} \cup \{\overline{P}t | \overline{P}t \in traces(U_1)\} \quad (1)$$

Def a Relation R is a visimulation if Whenever two processes PRQ it holds that : We have two conditions if P can proform some transition

$$\text{if } p \xrightarrow{\alpha} P' \text{ then there is } Q \xrightarrow{\alpha} Q' \wedge R' R Q' \quad (2)$$

$$\text{if } Q \xrightarrow{\alpha} Q' \text{ then there is } P \xrightarrow{\alpha} P' \wedge R' R Q' \quad (3)$$

Def P,Q are called Bisimilar (P Q) iff there is a bisimulation r st. PRQ

Desiderata for behaviorial eq.

-Equivalence relation

-Congruence (w.r.t to the syntax for out programming language)

def. A relation R over CCS processes is a bisimulation if whenever $(P, Q) \in R$ it hold that for any lable α :

Theorem 6. 1) If $p \xrightarrow{\alpha} p'$ then there is q' s.t. $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in R$

Theorem 7. 2) symmetric of 1)

P and Q are called bisimilar (P Q) if there a bisimulation R. S.t. $(P, Q) \in R$

Lemma: Bisimilarirt is an equivalence relation

Proof -Reflexivity $\forall P, P \vdash P \xrightarrow{\alpha} p', p \xrightarrow{\alpha} p' - I \subseteq$ -Prove that I is a bisimulation: $(P, Q) \in I \Rightarrow$

$P=Q \Rightarrow p \xrightarrow{\alpha} p'$ then $Q \xrightarrow{\alpha} p' (P', P') \in I$

-Symmetry: $P \sim Q \Rightarrow Q \sim P$. - By def. of bisimilarity there is a bisimulation R such that $(P, Q) \in R$. Claim R^{-1} (The relation where you flip every pair) $= \{(x, y) | (y, x) \in R\}$ is a bisimulation. $(Q, P) \in R^{-1}$ is bisimilar, You can conclude $Q \sim P$.

Given that R is a bisimulation then R^{-1} is a bisimulation for any R . let $(s, t) \in R^{-1}$. 1) For any $S \xrightarrow{\alpha} S'$, let T be any s.t. $(S, T) \in R$ (There is at least one since R is a Bisimulation). (Since S is in relation with T by our definition, by this point by construction.) It follows that $T \xrightarrow{\alpha} T'$, $(S', T') \in R^{-1}$ by the above formula. this proves the first conditioning of the def. of bisimulation.

2.) Likewise: given $T \xrightarrow{\alpha} T'$, there is a transition $S \xrightarrow{\alpha} S'$ s.t. $(T, S) \in R$ hence $(T', S') \in R^{-1}$.

-Transitivity: $P \sim Q, Q \sim S \Rightarrow P \sim S$, **The strategy here is to show that it is similar to what we did before.** -From $P \sim Q$ it follows that $\exists R_1$ s.t. R_1 is a bisimulation and $(P, Q) \in R_1$ -From $Q \sim S$ it follows that $\exists R_2$ s.t. R_2 is a bisimulation and $(Q, S) \in R_2$.

Define $R = R_1 \dot{\cup} R_2 = \{(x, y) | \exists y. (x, y) \in R_1 \wedge (y, z) \in R_2\}$ If R is a bisimulation by construction $(P, S) \in R$ hence $P \sim S$.

$(T_1, T_2) \in R$ **If t1 can perform an action, then t2 can do the same with the same labels.** 1)

let $T_1 \xrightarrow{\alpha} T_2$, by construction of R . T_3 s.t. $(T_1, T_3) \in R_1$, therefore $T_3 \xrightarrow{\alpha} T'_3$ s.t. $(T'_3, T'_2) \in R_1$

$(T_3, T_2) \in R_2$, since R_2 is a bisimulation, $T_2 \xrightarrow{\alpha} T'_2$ s.t. $(T'_3, T'_2) \in R$ **Inset picture here**

$T'_1 R_1 T'_3 R_2 T'_2 \Rightarrow T'_1 R T'_2$

2) $T_2 \xrightarrow{\alpha} T'_2$,

Lemma: \sim is a bisimulation See notes

Lemma: \sim is a congruence $\forall C, P, Q. P \sim Q \Rightarrow C[P] \sim C[Q]$ The result is a result for a weaker/equivariant one.

Proof: by structural induction Let $P \sim Q$, Proceed by structural induction on C , Base: $C = _$, (No holes) It follows by reflexivity of \sim if $C = _$ then $C[P] = P \sim Q = C[Q]$ induction case:

- $C = \alpha. _$ $\alpha.P \sim \alpha.Q$
- $C = S | _$ $S | P \sim S | Q$
- $C = _ | S$ $S | P \sim S | Q$
- $C = S + _$
- $C = _ + S$
- $C = _ \setminus L$

Lemma: If $P \sim Q$ then: 1) $a.P \sim a.Q$ 2) $\forall S. S | P \sim S | Q$ 3) $\forall S. S + P \sim S + Q$

1) $P \sim Q \Rightarrow a.P \sim a.Q$ let R be a Bisimulation s.t. $(P, Q) \in R$. Def. $R' = R \cup \{(a.P, a.Q)\}$ The two conditions

- $a.P \xrightarrow{\alpha} p$ the other process can only reply with $a.Q \wedge a.Q \xrightarrow{\alpha} Q$. $(P, Q) \in R'$,

2) $P \sim Q \Rightarrow \forall S. S | P \sim S | Q$. Let R be a bisimulation $(P, Q) \in R$. -Def. R as the following relation $R = \{(S | P, S | Q) | P \sim Q, P, Q, S \text{ are CCS Processes}\}$ Assume that $S | P \xrightarrow{\alpha} S' | P'$. Derivations for $S | P \xrightarrow{\alpha} S' | P'$

have thererfo: the first rule applied is 1) Lpar, 2) Rpar, 3, com

1) Lpar

$$\frac{S \xrightarrow{\alpha} S' \quad S \xrightarrow{\alpha} S'}{S | P \xrightarrow{\alpha} S' | P' \quad (P = P')} \quad \frac{S \xrightarrow{\alpha} S'}{S | Q \xrightarrow{\alpha} S' | Q} \quad (4)$$

$(S' | P, S' | Q) \in R$ by def of R

2) Rpar

$$\frac{P \xrightarrow{\alpha} P'}{S | P \xrightarrow{\alpha} S | P'} \quad (S = S') \text{ Since } P \sim Q \exists Q' \xrightarrow{(*)} Q' P' \dagger Q' \quad (5)$$

$$\text{From } (*) \frac{Q \xrightarrow{\alpha} Q'}{S | Q \xrightarrow{\alpha} S | Q'} \text{ And from } (\dagger) (S | P \text{TEST}', S | Q') \in R \quad (6)$$

3) com

$$\frac{S \xrightarrow{\alpha} S' \quad P \xrightarrow{\bar{\alpha}} P'}{S|P \xrightarrow{\tau} S'|P'} \quad \text{Since } P \mid Q, Q \xrightarrow[\star]{\bar{\alpha}} Q' \mid P' \xrightarrow[\dagger]{\alpha} Q' \quad (7)$$

$$\text{From } (\star) \frac{S \xrightarrow{\alpha} S' \quad Q \xrightarrow{\bar{\alpha}} Q'}{S|Q \xrightarrow{\tau} S'|Q'} \quad \text{And from } (\dagger)(S'|P', S'|Q') \in R. \quad (8)$$

Likewise $S|Q \xrightarrow{\alpha} S'|Q' \dots$

PFor $S+P \quad S+Q$ use the relation **Picture from phone**

$P|Q \quad Q|P$ prove that $R = \{ (P|Q, Q|P) \}$ is a bisimilar

For an LTS $(S, \text{ACT}, \rightarrow)$ It's saturation is the LTS $S, \text{ACT}, \Rightarrow$ where \Rightarrow is the least relation s.t.

We can ignore internal computation

$$\frac{\text{fll}}{S \rightarrow \llbracket \tau A \rrbracket} \quad (9)$$