計算モデル特論

プロセス計算

国立情報学研究所
佐藤一郎
E-mail: ichiro@nii.ac.jp

記述例

$Match \stackrel{def}{=} strike.0$ 

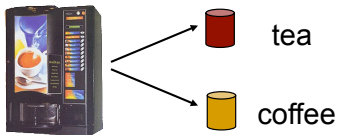
$VM \stackrel{def}{=} coin.\overline{coffee}.VM$ 

$Clock \stackrel{def}{=} tick.Clock$ 

選択処理（Choice Operator）

- 条件文や分岐を表す
  - *P* と *Q* がプロセス式ならば、P+Qもプロセス式
- P+Qにおいて、
  - もしPの方が先にアクションを実行すれば P+QはPとして振る舞う、逆にQの方が先にアクションを実行すればP+QはQとして振る舞う

$VM = coin.(button1.\overline{coffee}.VM + buton2.\overline{tea}.VM)$



Example

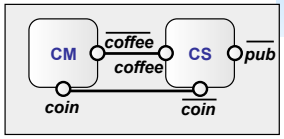
$CM \stackrel{def}{=} coin.\overline{coffee}.CM$ 

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

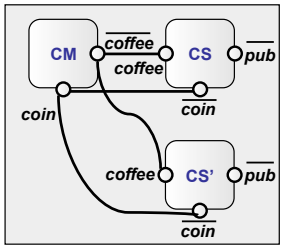
$CS \mid CM$ 

Example

CM | CS



CS|CM|CS′



記述例

- 通信プロトコル
  - 送信側(S)、受信側(R)、通信チャネル(C)

S = send.S' | T
S' = $\overline{\text{data}_S}$.$\overline{\text{start}}$.(ack.S+timeout.S')

T = start.$\overline{\text{timeout}}$.T

C = data_S.($\overline{\text{data}_R}$.C+τ.C) + ack_R.($\overline{\text{ack}_S}$.C+τ.C)

R = data_R.$\overline{\text{ack}_R}$.$\overline{\text{receive}}$.R

自動販売機の例

$$Ven1 = c.(juice.\overline{getj}.Ven1 + c.beer.\overline{getb}.Ven1)$$
$$Ven2 = c.juice.\overline{getj}.Ven1 + c.c.beer.\overline{getb}.Ven2$$

c:100円を入れる
$juice$:ジュースのボタンを押す,$beer$:(ビールのボタンを押す)
$\overline{getj}$:ジュースを取り出す,$\overline{getb}$:ビールを取り出す

$Ven1 \xrightarrow{c} p$においてpが次にできる相互作用は
cとjuiceの両方
$ven2 \xrightarrow{c} q$においてqが次にできる相互作用は
cかjuiceかのどちらか一方

記述例

- 通信プロトコル
  - 送信側(S)、受信側(R)、通信チャネル(C)

S = send.S'
S' = $\overline{\text{data}_S}$.$\overline{\text{start}}$.(ack_S.$\overline{\text{stop}}$.S+timeout.S')

T = start.($\overline{\text{timeout}}$.T+stop.T)

C = data_S.($\overline{\text{data}_R}$.C+τ.C) + ack_R.($\overline{\text{ack}_S}$.C+τ.C)

R = data_R.$\overline{\text{ack}_R}$.$\overline{\text{receive}}$.R

展開

- 操作意味論に従って式を展開

S|T|C|R =
send.S' | start.($\overline{\text{timeout}}$.T+stop.T) |
data_S.($\overline{\text{data}}_R$.C+τ.C) + ack_R.($\overline{\text{ack}}_S$.C+τ.C) |
data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

send
→ $\overline{\text{data}}_S$.$\overline{\text{start}}$.(ack_S.$\overline{\text{stop}}$.S+timeout.S') |
start.($\overline{\text{timeout}}$.T+stop.T) |
data_S.($\overline{\text{data}}_R$.C+τ.C) + ack_R.($\overline{\text{ack}}_S$.C+τ.C) |
data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R (※A)

展開

τ(data_S)
→ $\overline{\text{start}}$.(ack_S.$\overline{\text{stop}}$.S+timeout.S') |
start.($\overline{\text{timeout}}$.T+stop.T) |
$\overline{\text{data}}_R$.C+τ.C | data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

τ(start)
→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
$\overline{\text{data}}_R$.C+τ.C | data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

二つの展開がありうる (※B)

τ(data_R)
→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
C | $\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

展開

→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
data_S.($\overline{\text{data}}_R$.C+τ.C) + ack_R.($\overline{\text{ack}}_S$.C+τ.C) |
$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

τ(ack_R)
→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
$\overline{\text{ack}}_S$.C+τ.C | $\overline{\text{receive}}$.R

τ($\overline{\text{receive}}$)
→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
$\overline{\text{ack}}_S$.C+τ.C | R

二つの展開がある (※C)

τ(ack_S)
→ $\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T | C | R

τ(stop)
→ S | T | C | R

展開

(※B)→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
$\overline{\text{data}}_R$.C+τ.C | data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

τ
→ ack_S.$\overline{\text{stop}}$.S+timeout.S' | $\overline{\text{timeout}}$.T+stop.T |
C | data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

τ(timeout)
→ S' | T | C | data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R

= data_S.$\overline{\text{start}}$.(ack_S.$\overline{\text{stop}}$.S+timeout.S') |
start.($\overline{\text{timeout}}$.T+stop.T) |
data_S.($\overline{\text{data}}_R$.C+τ.C) + ack_R.($\overline{\text{ack}}_S$.C+τ.C) |
data_R.$\overline{\text{ack}}_R$.$\overline{\text{receive}}$.R （※Aと同じ）

展開

(※C) → $ack_S.\overline{stop}.S+timeout.S' \mid \overline{timeout}.T+stop.T \mid \overline{ack}_S.C+\tau.C \mid R$

二つの展開がある

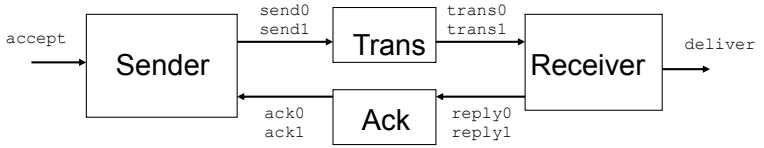
$\xrightarrow[\tau(send)]{\tau(ack_S)}$ $\overline{stop}.S+timeout.S' \mid \overline{timeout}.T+stop.T \mid C \mid R$
$S \mid T \mid C \mid R$

または

$\xrightarrow[\tau(timeout)]{\tau}$ $ack_S.\overline{stop}.S+timeout.S' \mid \overline{timeout}.T+stop.T \mid C \mid R$
$S \mid T \mid C \mid R$

Ichiro Satoh

Alternating Bit Protocol



Sender,Receiverが0,1の状態をもち、相手から異なる状態の信号を受け取ると状態を反転させる

初期状態　Sender=0, Receiver=1

途中の通信路で信号の消失や複製があっても信頼性のある通信が可能

$$(\mathrm{Accept}(0)|\mathrm{Trans}|\mathrm{Ack}|\mathrm{Replying}(0))\backslash A \approx Spec$$
$$\mathrm{Spec} = accept.\overline{deliver}.\mathrm{Spec}$$

Ichiro Satoh

ABPの記述(Sender,Receiver)

$$\begin{aligned}
Send(b) &= \overline{send_b}.Sending(b)\\
Sending(b) &= \tau.Send(b) + ack_b.Accept(\hat{b})\\
&\quad + ack_{\hat{b}}.Sending(b)\\
Accept(b) &= accept.Send(b)\\
Reply(b) &= \overline{reply_{\hat{b}}}.Replying(b)\\
Replying(b) &= \tau.Reply(b) + trans_b.Deliver(\hat{b})\\
&\quad + trans_{\hat{b}}.Replying(b)\\
Deliver(b) &= \overline{deliver}.Reply(b)
\end{aligned}$$

ここで$\hat{0} = 1, \hat{1} = 0$

Ichiro Satoh

ABPの記述(Trans,Ack)

$$\begin{aligned}
Trans &= send_0.(Trans + \overline{trans_0}.Trans)\\
&\quad + send_1.(Trans + \overline{trans_1}.Trans)\\
Ack &= reply_0.(Ack + \overline{ack_0}.Ack)\\
&\quad + reply_1.(Ack + \overline{ack_1}.Ack)
\end{aligned}$$

$$(\mathrm{Accept}(0)|\mathrm{Trans}|\mathrm{Ack}|\mathrm{Replying}(0))\backslash A \approx Spec$$

where $A = \{\mathrm{send0}, \mathrm{send1}, \mathrm{trans0}, \mathrm{trans1}, \mathrm{ack0}, \mathrm{ack1}, \mathrm{reply0}, \mathrm{reply1}\}$

Ichiro Satoh