Timed CSP Explorer

山川武志，福永　力

首都大学東京・数理情報科学専攻

山川武志の修士論文から
（2010年3月）

Contents

- CSP & Timed CSP
- Timed CSP Explorer
  - MLの利用
  - 時間概念の導入
  - CSP_M → MLへの変換
  - MLプロセスの実行
  - Trace検証
- Fischerアルゴリズムの検証
  （共有メモリの排他制御）

CSP & Timed CSP

- 従来のCSP：Eventによる Processの変化 $P \xrightarrow{\mu} P'$ のみ考慮
- Timed CSP：時間による Processの変化 $Q \overset{t}{\rightsquigarrow} Q'$ も考慮

CSP

- プロセス定義と従来のCSP_M記述

	CSP	CSP_M	
$Proc ::=$	$STOP$	`STOP`	
$\mid$	$SKIP$	`SKIP`	
$\mid$	$c \rightarrow Proc$	`c -> Proc`	*prefix*
$\mid$	$c?x : A \rightarrow Proc$	`c?x:A -> Proc`	*input*
$\mid$	$c!v \rightarrow Proc$	`c!v -> Proc`	*output*
$\mid$	$Proc \fatsemi Proc$	`Proc ; Proc`	*sequential*
$\mid$	$Proc \triangle Proc$	`Proc /\ Proc`	*interrupt*
$\mid$	$Proc \setminus A$	`Proc \ A`	*hiding*
$\mid$	$Proc[c \leftarrow d]$	`Proc[[c <- d]]`	*renaming*
$\mid$	$Proc \Box Proc$	`Proc [] Proc`	*external choice*
$\mid$	$Proc \sqcap Proc$	`Proc |~| Proc`	*internal choice*
$\mid$	$Proc \triangleright Proc$	`Proc [> Proc`	*timeout*
$\mid$	$Proc \parallel Proc$	`Proc || Proc`	*parallel*
$\mid$	$Proc \interleave Proc$	`Proc ||| Proc`	*interleaving*
$\mid$	$Proc [\![A]\!] Proc$	`Proc [|A|] Proc`	*sharing*
$\mid$	if b then $Proc$	`if b then Proc`	*conditional choice*
	else $Proc$	`else Proc`	

Timed CSP （1）

- Timed Event Prefix $P \overset{d}{\rightsquigarrow} P$

  イベントaが起きるまでの時間uで表す

  $(a@u \to P) \xrightarrow{a} P[0/u]$ aが起きた時，時間uのリセット

  $(a@u \to P) \overset{d}{\rightsquigarrow} (a@u \to P[u+d/u])$ aが起きず時間dのみ経過した場合

- Timeout $P \overset{d}{\triangleright} Q$

  Pになんら外部イベントが惹起されなければd時間後にPはTimeoutでQに移行

$$\frac{P \xrightarrow{a} P'}{P \overset{d}{\triangleright} Q \xrightarrow{a} P'} \qquad \frac{P \overset{d'}{\rightsquigarrow} P'}{(P \overset{d}{\triangleright} Q) \overset{d'}{\rightsquigarrow} (P' \overset{d-d'}{\triangleright} Q)}\,[0 < d' \leqq d]$$

- 遅延プロセス： $WAIT\ d = STOP \overset{d}{\triangleright} SKIP$

Timed CSP （2）

- Timed Interrupt　　$P \triangle_d Q$

  d時間内にPが終了しないときPを打切り，Qに移行

$$\frac{P \xrightarrow{\mu} P'}{(P \triangle_d Q) \xrightarrow{\mu} (P' \triangle_d Q)}[\mu \neq \surd] \qquad \frac{P \xrightarrow{\surd} P'}{P \triangle_d Q \xrightarrow{\surd} P'}$$

√ = 正常終了

- Timed Trace

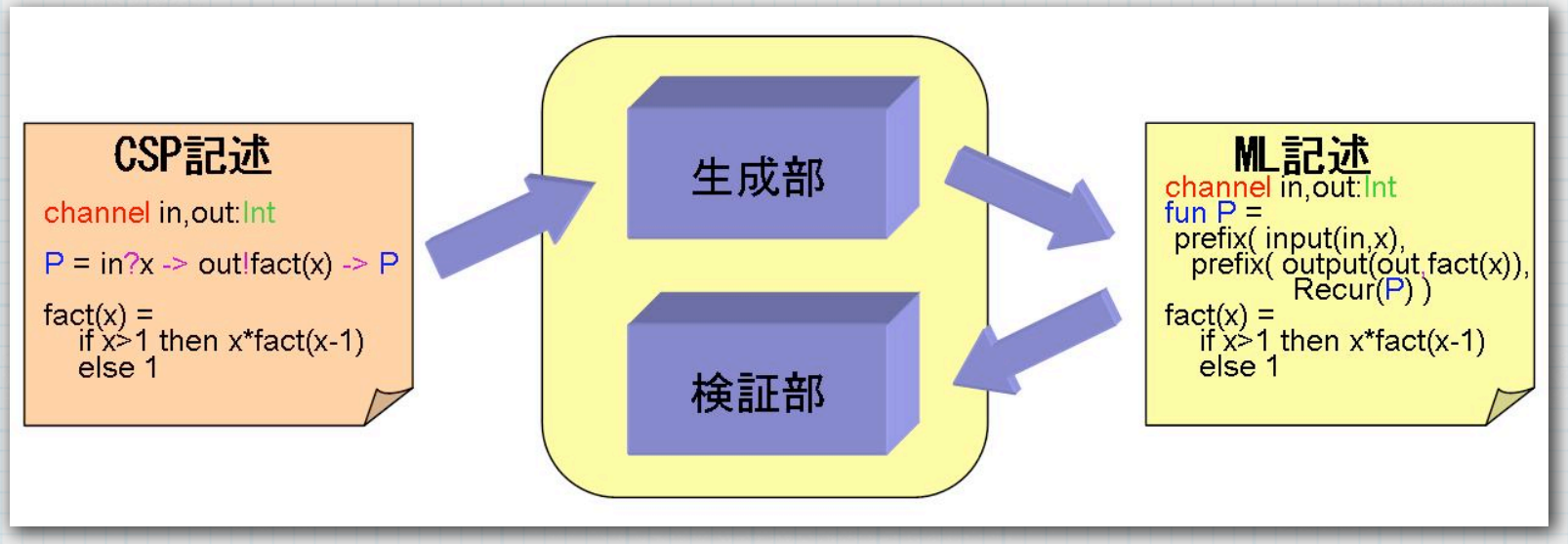
$$HELEN = (\ meet \xrightarrow{60} work \rightarrow SKIP) \overset{30}{\triangleright} work \rightarrow SKIP$$
$$CARL(d) = WAITd\ ;\ (\ meet \xrightarrow{60} home \rightarrow SKIP) \overset{45}{\triangleright} home \rightarrow SKIP$$

$HELEN \parallel CARL(d)$ のTimed Traceどうなるかはdによる.

$d = 15 \quad < (15, meet), (75, work), (75, home) >$
$d = 30 \quad < (30, work), (75, home) >$

Timed CSP Explorer

- ML（Meta Language）を主言語として採用
- Timed CSP記述プロセスの解釈，MLコード生成
- MLコード実行によりTimed Trace 解析による Refinement（Safety）検証 （FDR2のTimedプロセスへの拡張）



MLによるプロセス実装（1）

- ML datatypeの追加
  - Process
    datatype process
      = Stop| Skip| Proc of (event -> process) | Bleep
  - Event
    datatype event
      = Event of String*chanType | Time of int
  - Channel
    datatype chanType
      = Int of int | Seq of int list | String of string
        Tau | None

MLによるプロセス実装（2）

* Event prefix $a \rightarrow P$

```
prefix (Event(ch,v), P:process) =
  let
     fun temp(Event(ch',v'))=
     if ch=ch' andalso v=v' then P
     else Bleep
  in
     Proc temp
  end
```

- event 型とprocess型の数からprocess型を返す関数として定義
- temp関数の局所的定義
- 実行部：temp関数をprocess型として返す

時間概念の導入

- datatype eventにTime of intを加える
- CSPプロセスの時間概念の導入によるMLコードの改造
- Timed CSPプロセス関数の組み込み
  - fun tprefix (Event(ch,v),Time d,P:process)
  - fun timeout (P:process, Time d, Q:process)
  - fun tinterrput (P:process, Time d, Q:process)

Timed CSP_M記法の提案

Timed CSP_Mの構築

	Timed CSP prcesse	Machine Readable
Timed Event Prefix	$a\,@u \rightarrow P$	`a@u -> P`
Timeout	$P \overset{d}{\triangleright} Q$	`P [>♯d Q`
Event Prefix with time	$a \xrightarrow{d} P$	`a ->♯d P`
Timed interrupt	$P \triangle_d Q$	`P /\♯d Q`

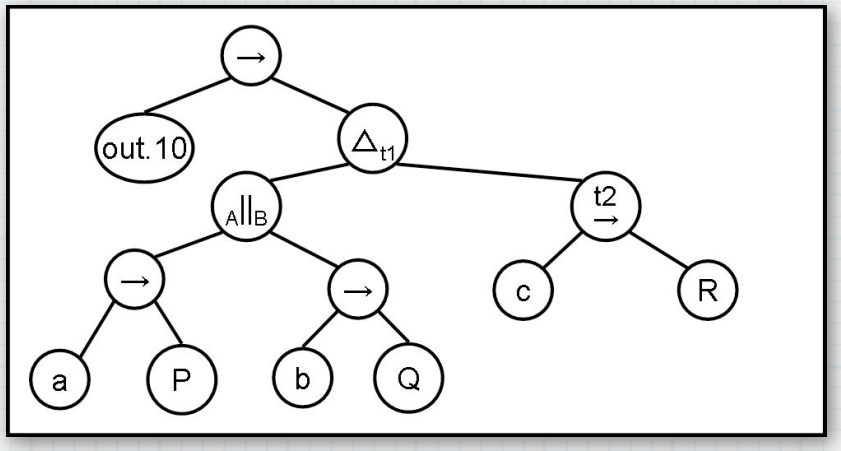
例

$a \xrightarrow{3} (P \overset{5}{\triangleright} Q)$ a -> #3 (P [>#5 Q)

CSP_M→ML変換

* 字句解析・構文解析（例）

$$TEST = out!10 \rightarrow ((a \rightarrow P \ {}_A||_B \ b \rightarrow Q) \triangle_{t1} \ c \xrightarrow{t2} R)$$

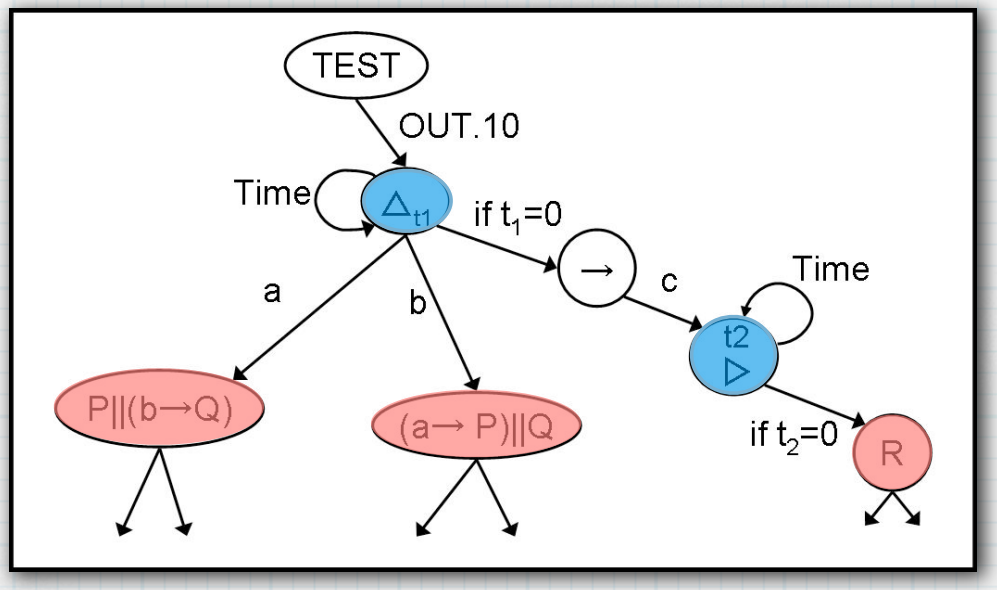



```
fun test() = prefix(out.10,
                tinterrupt(
                   parallel(
                        prefix(a,P), A, B, prefix(b,Q)
                   ),
                   Time t1,
                   tprefix(c, Time t2, R )
                )
             )
```

MLプロセスの実行例（1）

* MLプロセス=状態遷移グラフ

$$TEST = out!10 \rightarrow (\ (a \rightarrow P\ {}_A||_B\ b \rightarrow Q)\ \triangle_{t1}\ \ c \xrightarrow{t2} R\ )$$



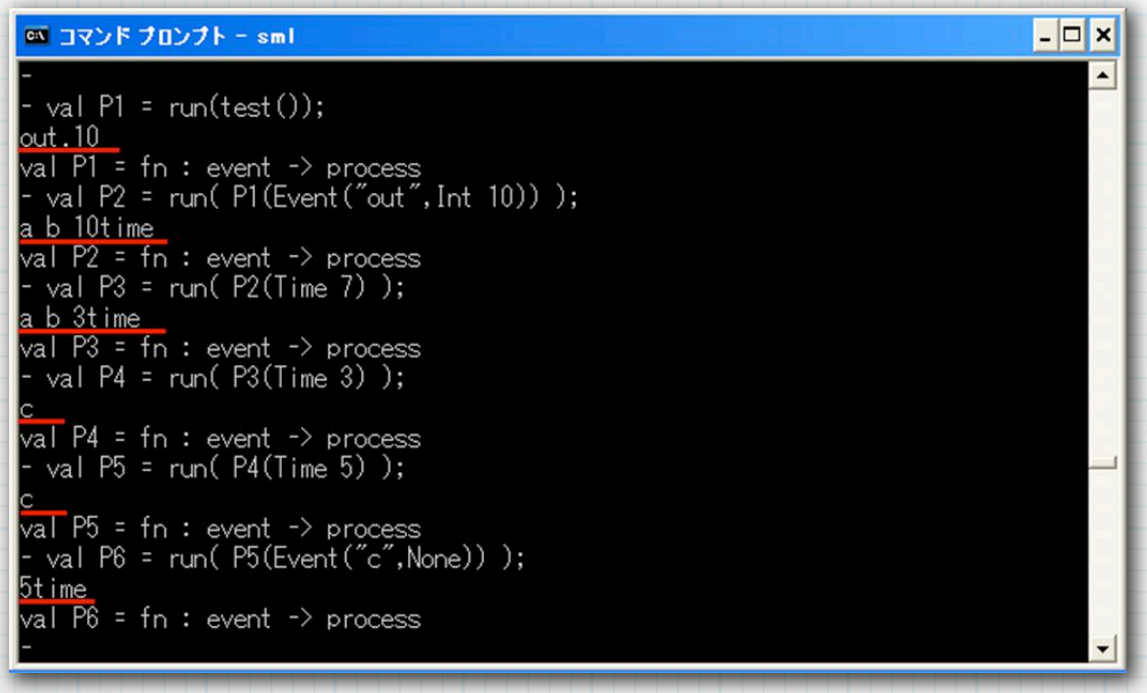
実行時
実際の値を指定

実行時
具体的プロセス指定

MLプロセスの実行例（2）

* *t*1=10,*t*2=5,*P*,*Q*,*R*=*SKIP*として実行

$$TEST = out!10 \rightarrow (\ (a \rightarrow SKIP\ {}_A||_B\ b \rightarrow SKIP) \triangle_{10}\ c \xrightarrow{5} SKIP\ )$$



次に選択可能なイベント

Refinement検証

- trace refinement（CSP traceによるsafety検証）

$$SPEC \sqsubseteq_T IMP \Longleftrightarrow traces(SPEC) \supseteq traces(IMP)$$

- timewise trace refinement

$$SPEC \ {}_T \sqsubseteq_{TF} IMP$$

$$\Longleftrightarrow \forall (s, X) \in TF[IMP] \cdot \sharp s < \infty \Rightarrow strip(s) \in traces(SPEC)$$

$$strip(< (15, meet), (45, work), (45, home) >) =< meet, work, home >$$

- *IMP*側で成立するすべての（Timed ）Traceについてその成立性を*SPEC*側でも検査する

Fischerのアルゴリズム

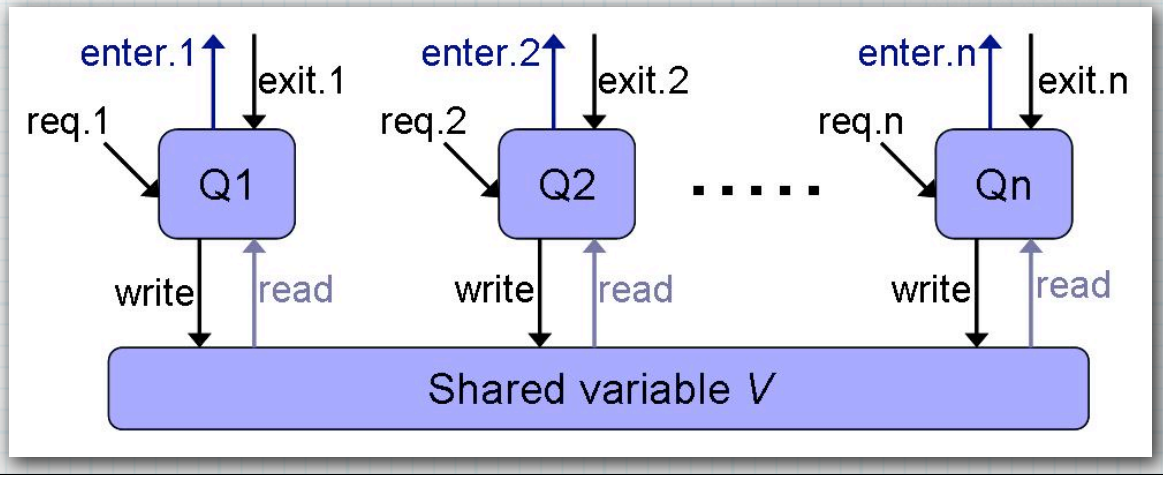
* 共有資源（メモリ）排他制御

* $Q(i) \quad = \quad req.i \rightarrow read?x \rightarrow$
  $\text{if } x \neq 0 \text{ then } SKIP$
  $\text{else } write!i \rightarrow enter.i \rightarrow exit.i \rightarrow SKIP$
  $QS = Q(1) \;|||\; Q(2)$
* $V(value) = (write?x \rightarrow V(x)) \;\square\; (read!value \rightarrow V(value))$
* $FIS = QS \;[|\; read, write \;|]\; V(0)$
* $FIS \quad sat \quad \#(s \uparrow enter.\mathbb{N}) \leq 1$

共有メモリを読みその値が0ならメモリにiを書く，
排他領域に入り，すぐに出る

全体のシステム（events read, write共有の）並列システム

enterは2回以上表れない



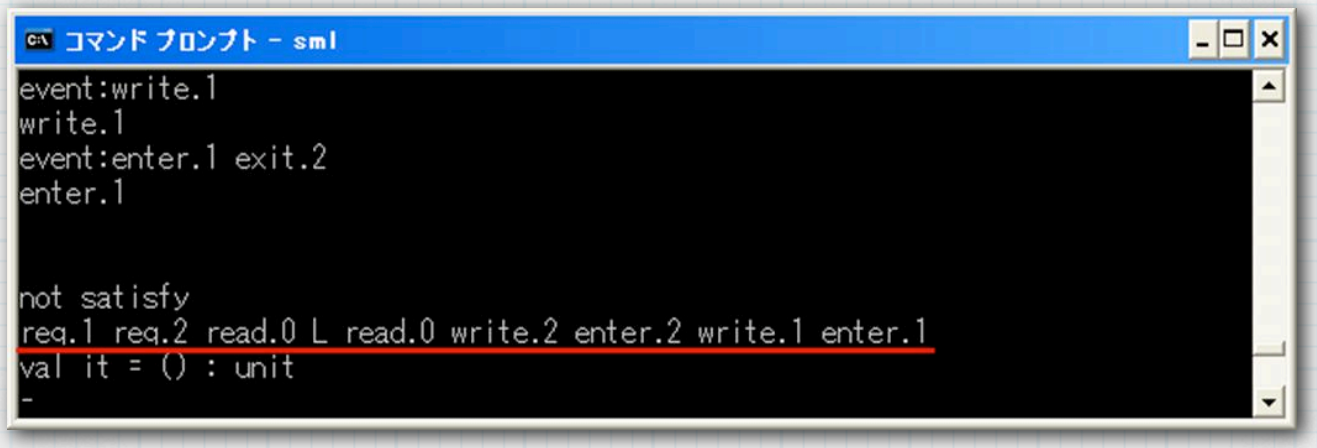
Fischerアルゴリズム検証

- Timed CSP Explorerによるトレース検証

$$SPEC = (enter.1 \rightarrow exit.1 \rightarrow SKIP \,\Box\, enter.2 \rightarrow exit.2 \rightarrow SKIP) \,|||\, RUN,$$
$$RUN = (req?x \rightarrow RUN) \,\Box\, (read?x \rightarrow RUN) \,\Box\, (write?x \rightarrow RUN$$

ここで*RUN*は*enter.i*以外の*events*を起こさせるプロセス

- enter.2, enter.1ともに起きているトレースの存在, 仕様（上記$SPEC$）を満たしていない.



Timed CSPによる Fischerアルゴリズム

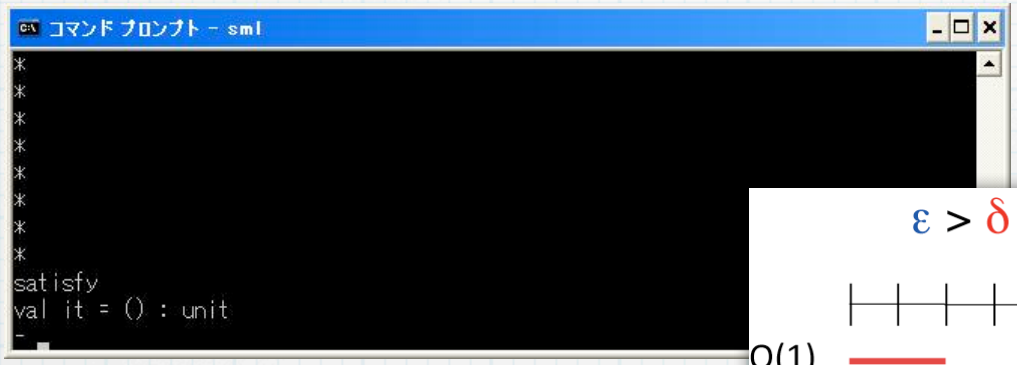
* Timed CSPを用いてQ(i)プロセスを再定義

$$
\begin{aligned}
Q(i) = {} & req.i \rightarrow read?x \rightarrow \\
& \text{if } x \neq 0 \text{ then } SKIP \\
& \text{else } (\ write!i \xrightarrow{\epsilon} read?y \rightarrow \\
& \qquad \text{if } y \neq i \text{ then } SKIP \\
& \qquad \text{else } enter.i \rightarrow exit.i \rightarrow STOP \ ) \overset{\delta}{\triangleright} STOP
\end{aligned}
$$

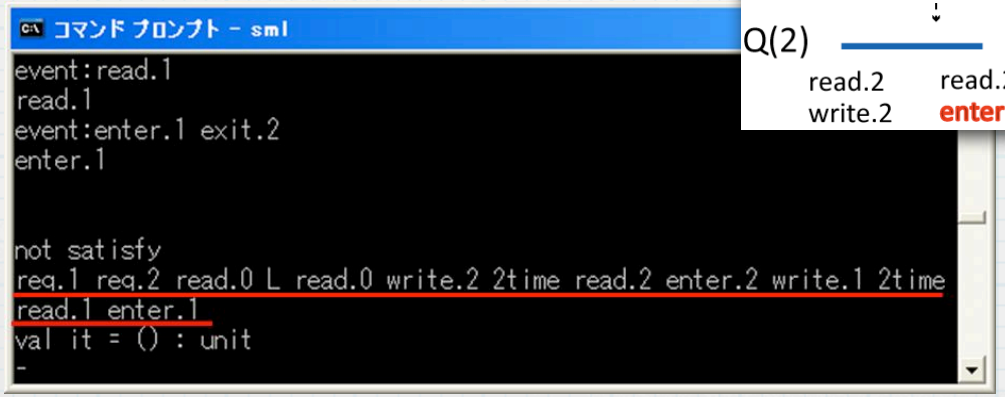
* 共有メモリに*i*を書く際待ち時間（ε）の導入
* 共有領域滞留時間の上限設定（δ）
* ε> δ を保つ必要がある

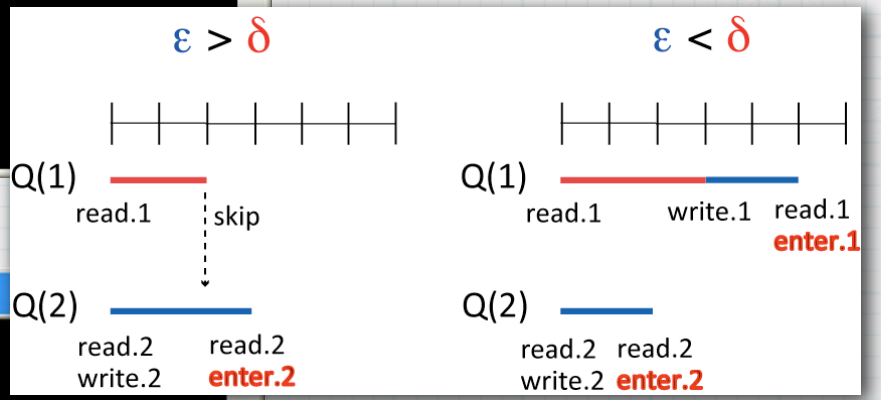
Timed Trace Refinement 検証結果

* ε> δ 成功



* ε< δ 失敗 ε=2, δ =3





まとめ

- 世の中に存在するできるだけ多くのTimed CSP記述例を用いてまじめにデバッグ（そのような例を募集中）
- Timed Trace→Timed Failure Refinementまで実装する