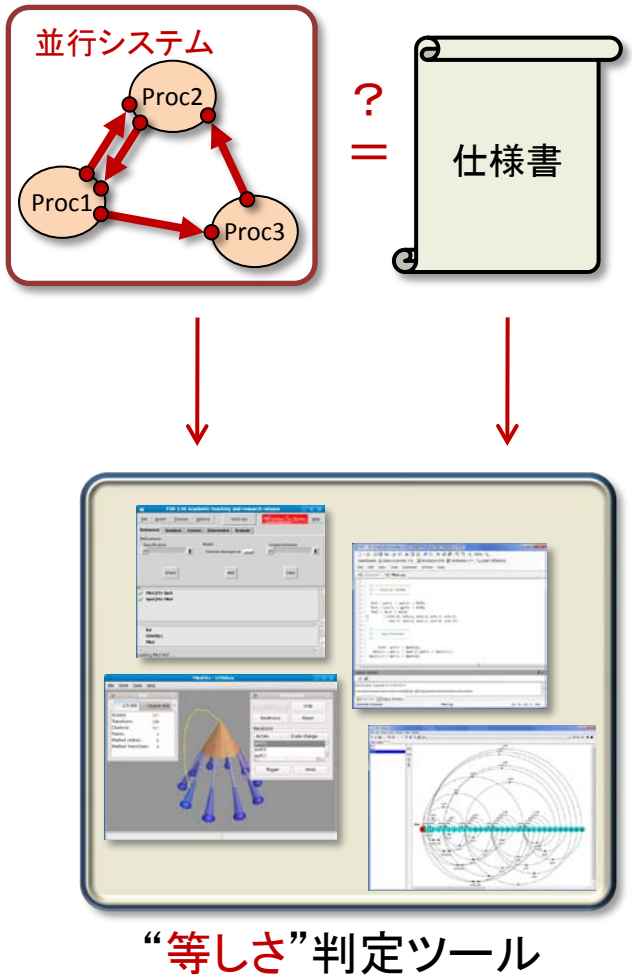
# プロセス代数の基本と関連ツールの紹介

産業技術総合研究所
情報技術研究部門
磯部祥尚

第5回CSP研究会資料　2010.07.10

# 講演内容

1. **プロセス代数**
   - ✓ プロセス代数とは？
   - ✓ プロセス代数による記述例
   - ✓ プロセス代数による解析例
   - ✓ 解析ツールの紹介
2. **等価性**
   - ✓ 強双模倣等価
   - ✓ 弱双模倣等価
   - ✓ 分岐双模倣等価
   - ✓ 失敗発散等価
3. **例題：ABプロトコル**
   - ✓ ABプロトコルの概要
   - ✓ 各プロセスの動作
4. **mCRL2による解析例**
   - ✓ mCRL2の特徴
   - ✓ ABプロトコルのmCRL2記述
   - ✓ 分岐双模倣等価の判定
   - ✓ 状態遷移図の表示
5. **FDR2による解析例**
   - ✓ FDR2の特徴
   - ✓ ライブロックの解消
   - ✓ 失敗発散等価の判定
6. **π計算**
   - ✓ π計算の特徴
   - ✓ チャネル渡しの例
   - ✓ MWBによる検証例
7. **まとめ**
   - ✓ 各ツールの情報
   - ✓ 等しさの強さ



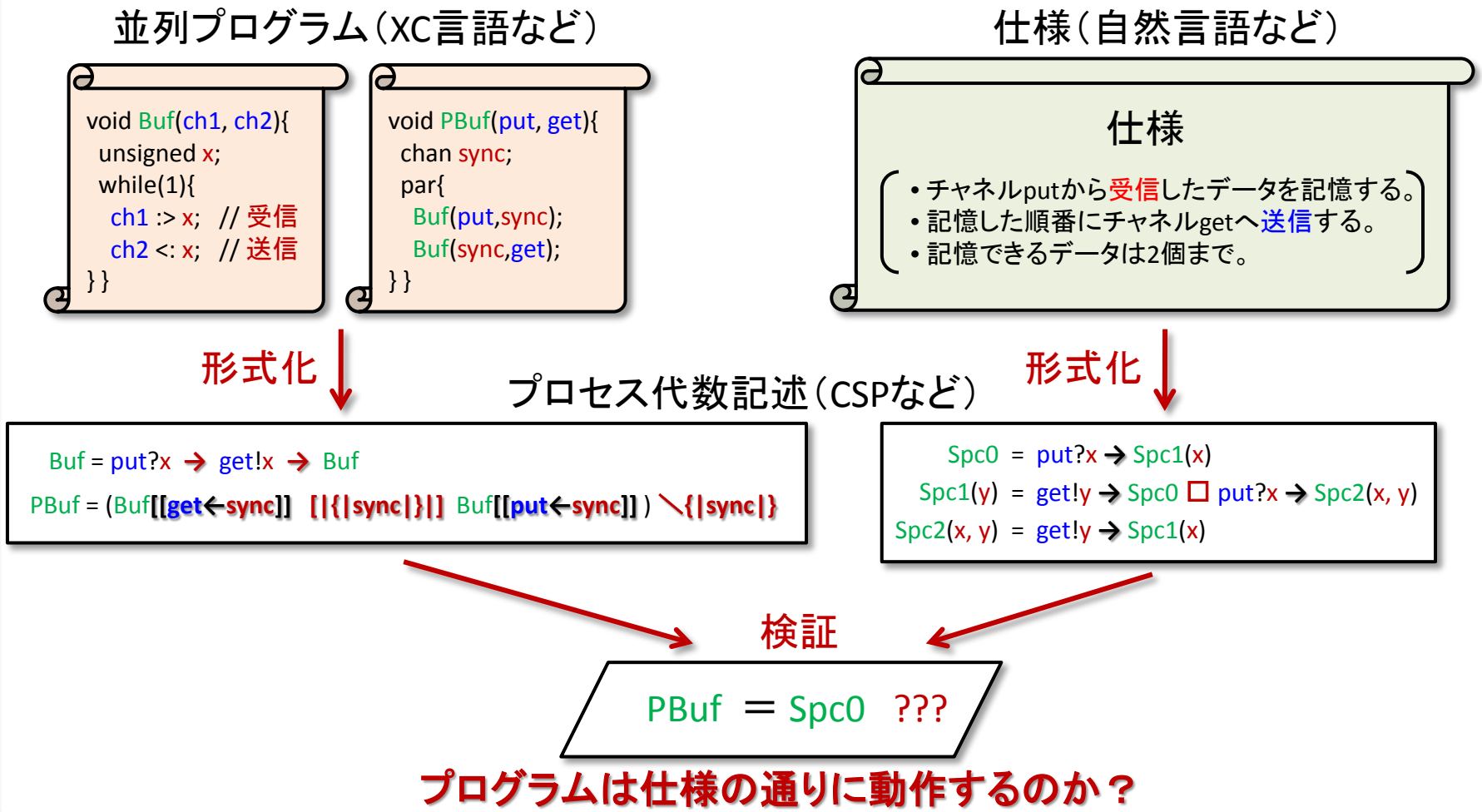

- プロセス代数とは？
- 動作が“等しい”とは？
- ツールで何ができるのか？

# プロセス代数

- プロセス代数とは？
- プロセス代数による記述例
- プロセス代数による解析例
- プロセス代数関連ツールの紹介

# プロセス代数

■ **プロセス代数**： 並行システムの構造や動作を記述して解析するための理論



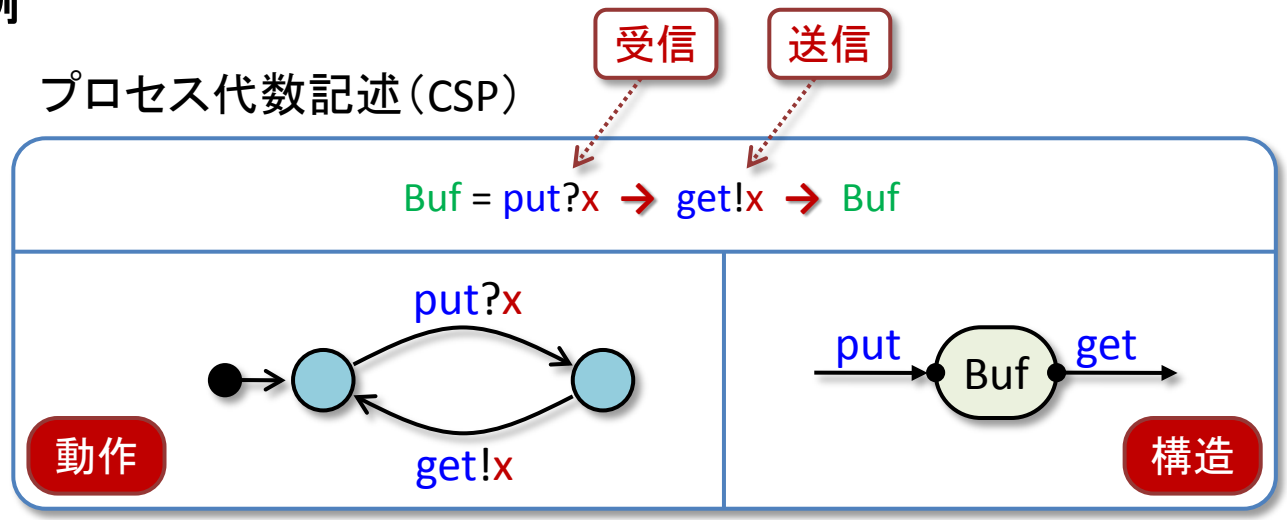
# プロセス代数による記述例(プログラム)

■ (並列)バッファの例

プログラム

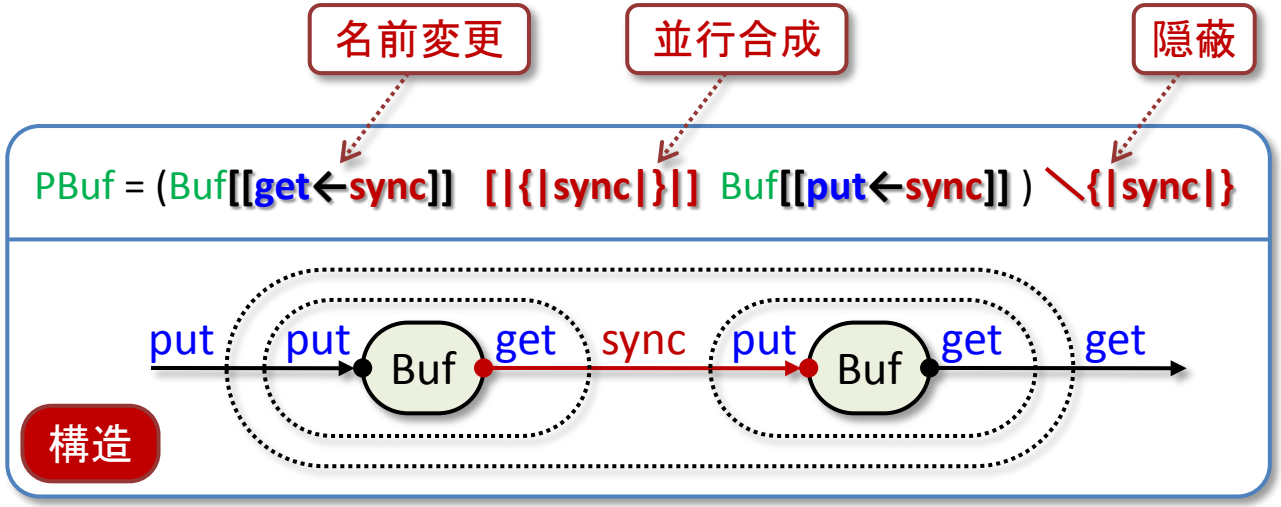
```
void Buf(put, get){
  unsigned x;
  while(1){
    put :> x;   // 受信
    get <: x;   // 送信
}}
```

プロセス代数記述(CSP)




並列プログラム

```
void PBuf(put, get){
  chan sync;
  par{
    Buf(put,sync);
    Buf(sync,get);
}}
```



# プロセス代数による記述例（仕様）

■ 容量2のバッファの仕様の例

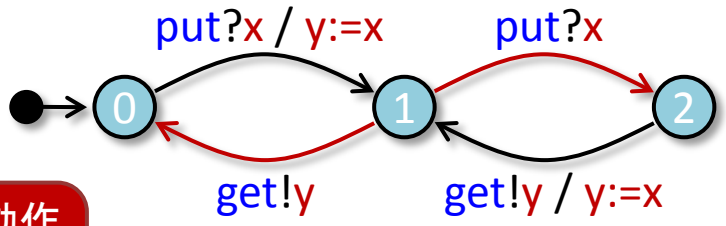
- チャネルputから受信したデータを記憶する。
- 記憶した順番にチャネルgetへ送信する。
- 記憶できるデータは2個まで。

選択

Spc0 = put?x → Spc1(x)

Spc1(y) = get!y → Spc0 □ put?x → Spc2(x, y)

Spc2(x, y) = get!y → Spc1(x)




動作


構造

# プロセス代数による解析例

■ 並列バッファPBufは容量2のバッファの仕様Spc0の通りに動作する？

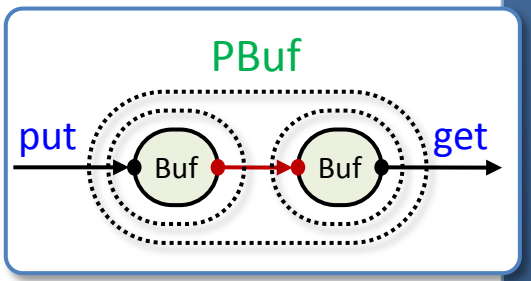
Buf = put?x → get!x → Buf

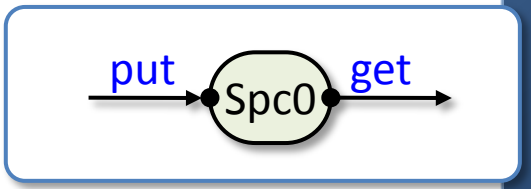
PBuf = (Buf[[get←sync]] [|{|sync|}|] Buf[[put←sync]]) ＼{|sync|}

Spc0 = put?x → Spc1(x)

Spc1(y) = get!y → Spc0 □ put?x → Spc2(x, y)

Spc2(x, y) = get!y → Spc1(x)




PBuf $=_{FD}$ Spc0　PBufとSpec0は失敗発散等価である。（FD: Failures-Divergence）

PBuf $=_{WB}$ Spc0　PBufとSpec0は弱双模倣等価である。（WB: Weak-Bisimulation）

PBuf $\neq_{SB}$ Spc0　PBufとSpec0は強双模倣等価でない。（SB: Strong-Bisimulation）

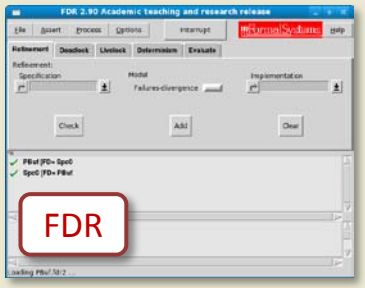
様々な“動作の等しさ”がある

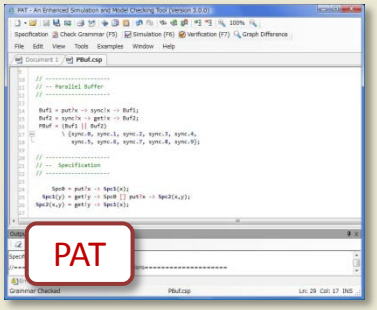
# “動作の等しさ”判定ツールの例

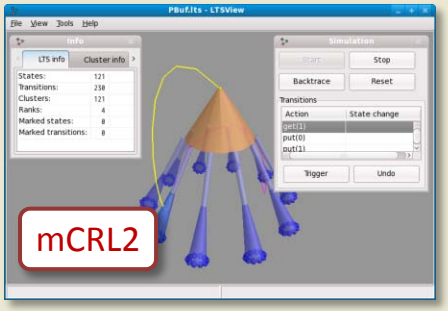
■ “動作の等しさ” 判定ツールの例

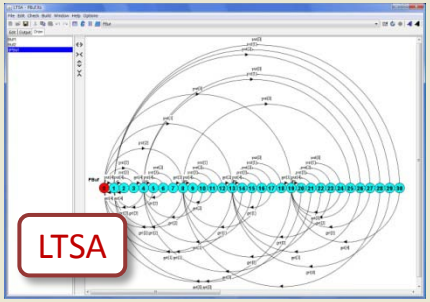
| ツール | 理論 | 等価性 |
|---|---|---|
| FDR | CSP | T, FD, etc |
| PAT | CSP# | T, FD, etc |
| mCRL2 | ACP | SB, BB, etc |
| CWB | CCS | SB, WB, etc |
| MWB | π計算 | SB, WB, etc |
| LTSA | FSP（CSP系） | WB (min) |

モデル検査器




■ 代表的なプロセス代数

- CSP：C.A.R.Hoare, オクスフォード大学, 1978
- CCS：R. Milner, エジンバラ大学, 1980
- ACP：J.A. Bergstra and J.W. Klop, アムステルダム数学センター（現：CWI）, 1982
- π計算：R. Milner, J. Parrow and D. Walker, エジンバラ大学, 1992

T：トレース等価
FD：失敗発散等価
WB：弱双模倣等価
BB：分岐双模倣等価
SB：強双模倣等価

CSP, CCS, ACP の記述力には大差はない。π計算はCCS＋チャネル渡し。

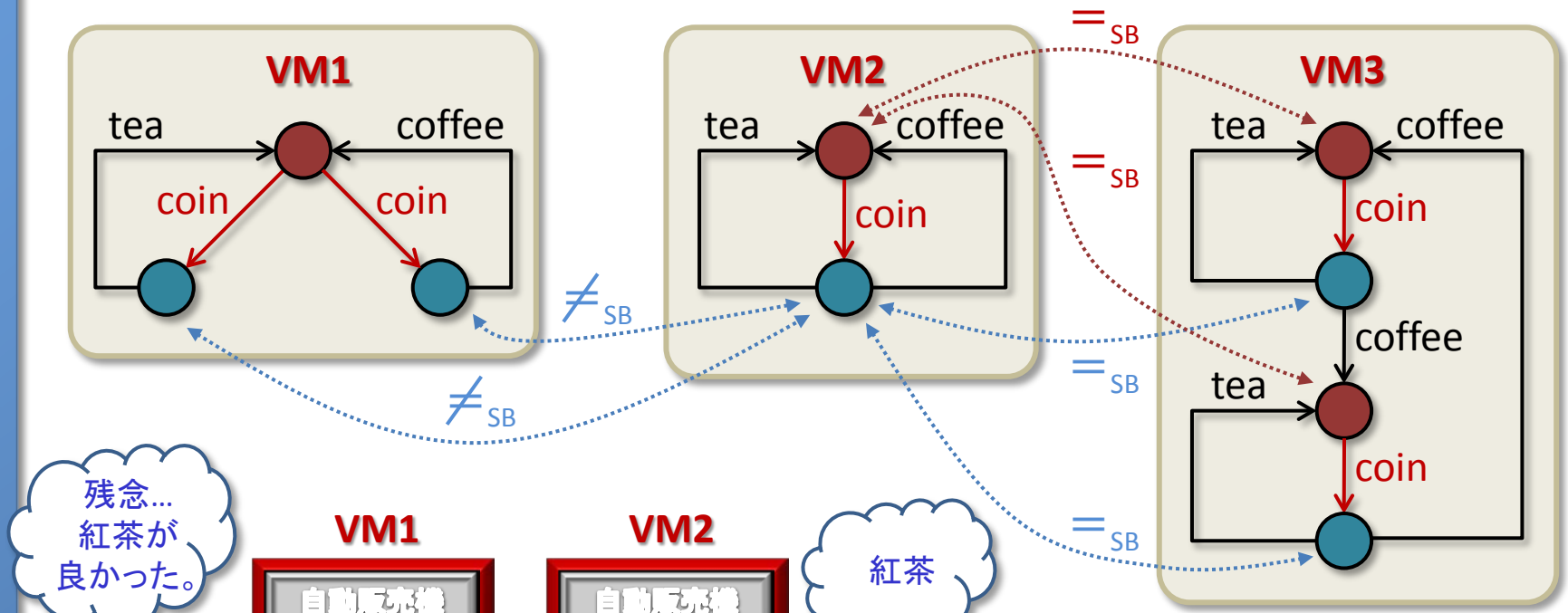
CSP：Communicating Sequential Process
CCS：Calculus of Communicating Systems
ACP：Algebra of Communicating Process

# 等価性

- 強双模倣等価
- 弱双模倣等価
- 分岐双模倣等価
- 失敗発散等価

# 強双模倣等価 $=_{SB}$ （Strong Bisimulation Equiv）

■ 状態遷移図において、状態の組が強双模倣等価であるとは、互いに同じイベントを実行でき、実行後の状態の組も強双模倣等価であること。





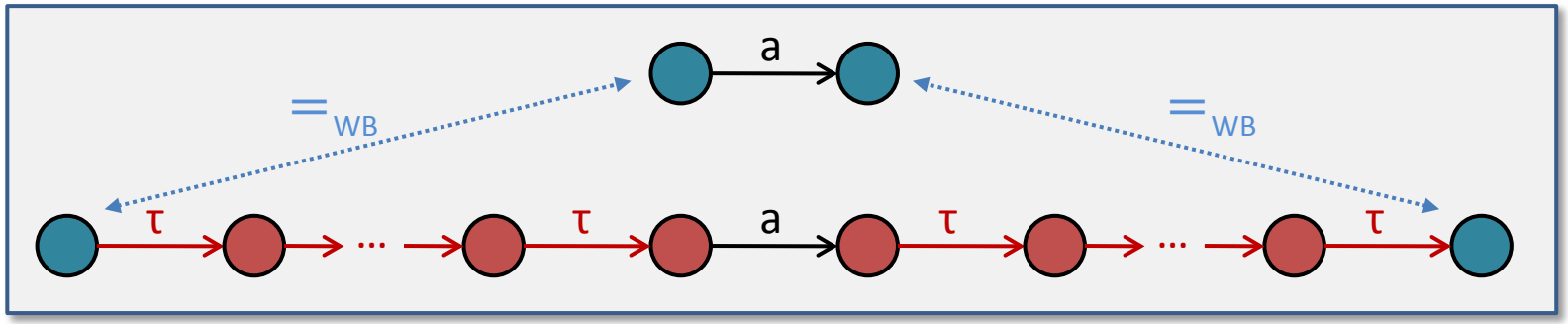
$VM2 =_{SB} VM3$

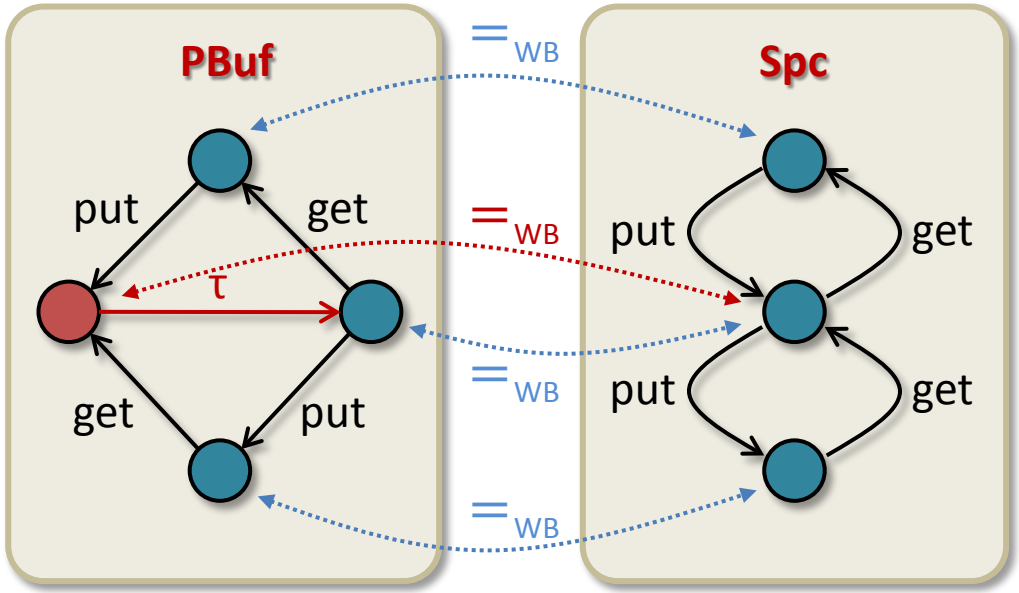
$VM1 \neq_{SB} VM2$

比較：$VM1 =_{T} VM2$

# 弱双模倣等価 $=_{WB}$ （Weak Bisimulation Equiv）

- 弱双模倣等価では有限個の 内部イベント τ による遷移を無視する（条件が弱い）。




簡単のためデータを省略

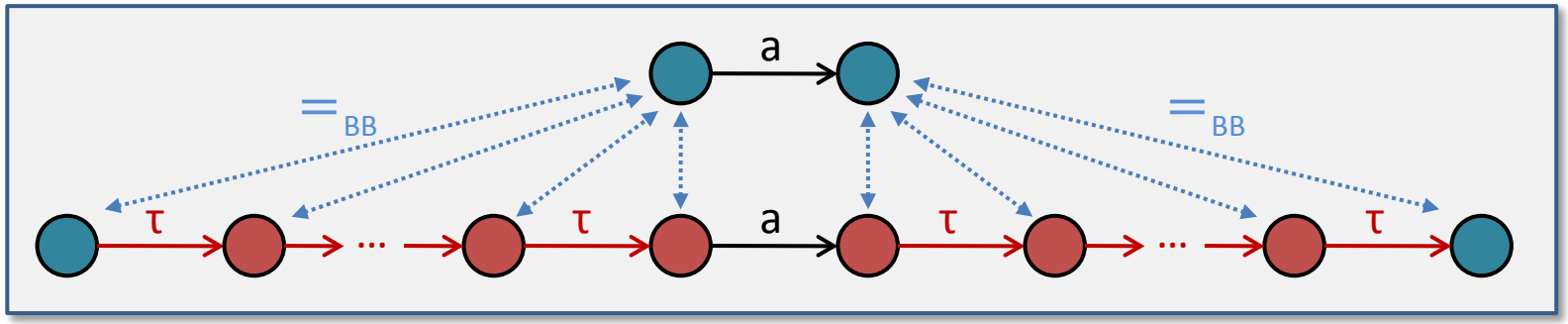
$PBuf =_{WB} Spc$

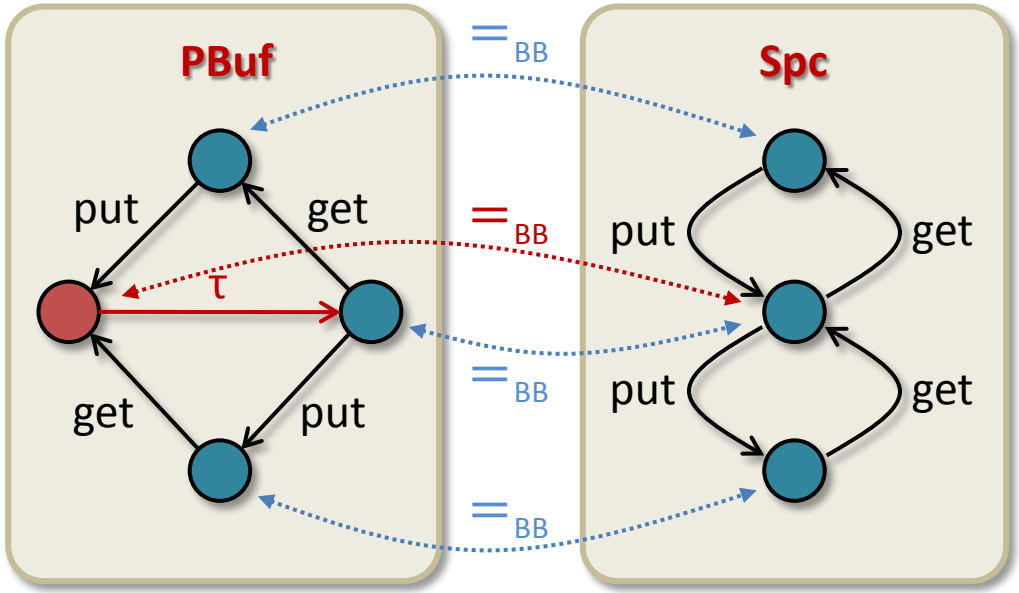
$PBuf \neq_{SB} Spc$

WB：弱双模倣等価
BB：分岐双模倣等価
SB：強双模倣等価

# 分岐双模倣等価 $=_{BB}$ （Branching Bisimulation Equiv）

■ 分岐双模倣等価では無視する内部イベント τ による遷移は等しさを維持する。




簡単のためデータを省略

弱い等しさ

$PBuf =_{WB} Spc$

$PBuf =_{BB} Spc$

$PBuf \neq_{SB} Spc$

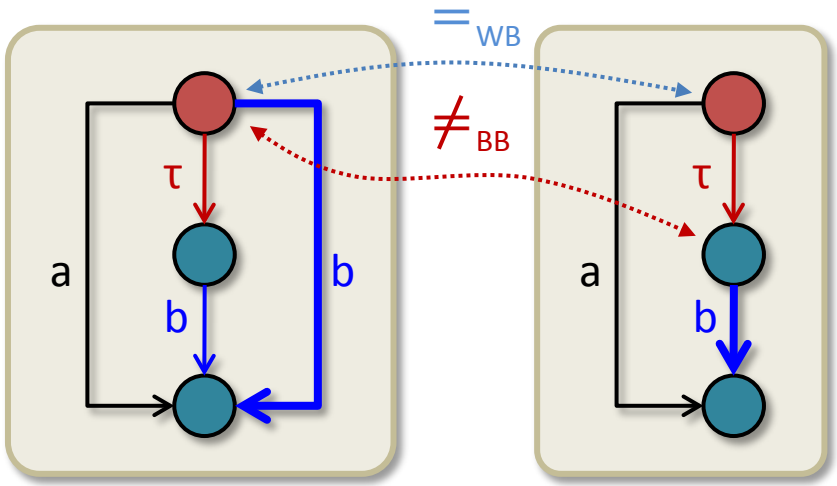
強い等しさ

WB：弱双模倣等価
BB：分岐双模倣等価
SB：強双模倣等価

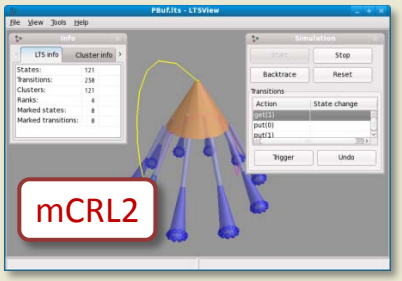
# 弱双模倣等価と分岐双模倣等価の比較

■ 弱双模倣等価であり、分岐双模倣等価ではない例

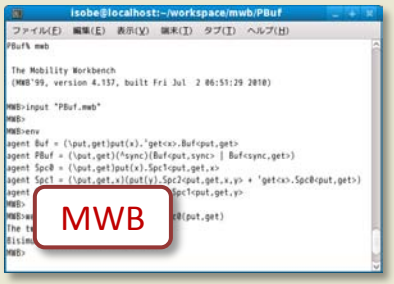
WB ：弱双模倣等価
BB ：分岐双模倣等価
SB ：強双模倣等価




■ 現実的なシステムを解析する場合は違いがでないことが多い。
⇒ 解析ツールを選ぶ場合、“弱”か“分岐”かの違いはあまり気にしなくてよい。



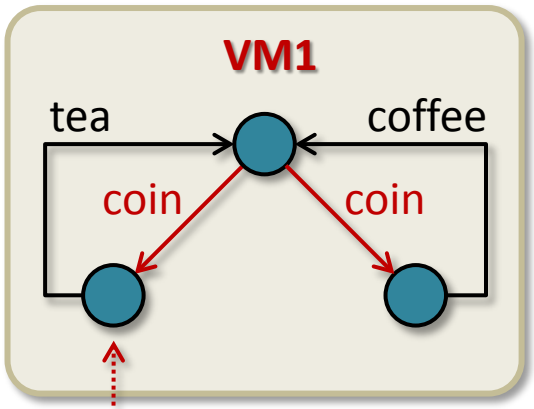

- 分岐双模倣等価
- GUI
- データ表現力



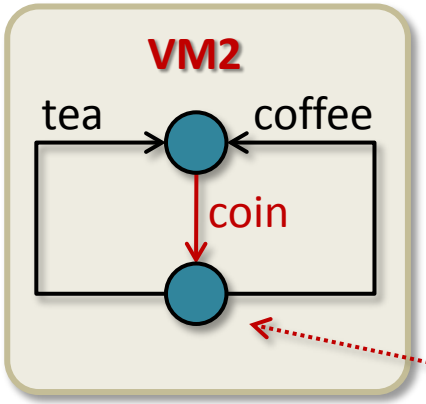

- 弱双模倣等価
- チャネル渡し

# 失敗発散等価 $=_{FD}$ （Failures Divergence Equiv）

■ 失敗発散等価とは、安定状態へのトレースとその安定状態で実行できないイベントとライブロックの有無が同じであること。




$\neq_{FD}$



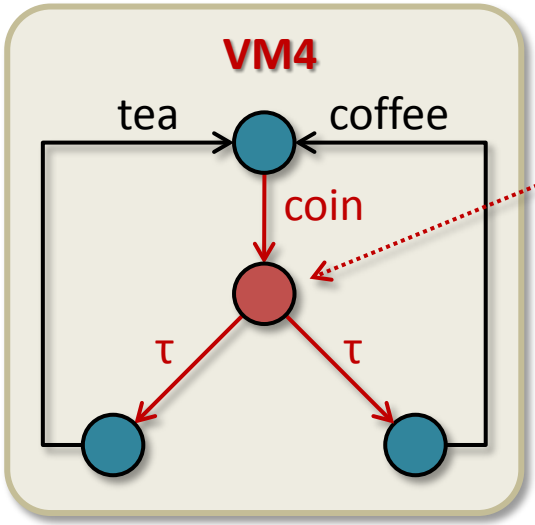


coinの後必ずcoffeeを実行できる

coinの後coffeeを実行できない

$=_{FD}$

$\neq_{WB}$




不安定状態

$VM1 \neq_{FD} VM2$

$VM2 =_{FD} VM4$

$VM2 \neq_{WB} VM4$

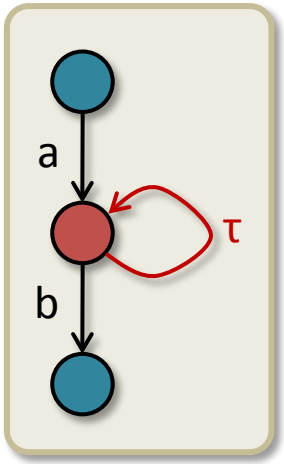
WB：弱双模倣等価
FD：失敗発散等価

# 失敗発散等価と弱双模倣等価の比較

WB：弱双模倣等価
FD：失敗発散等価

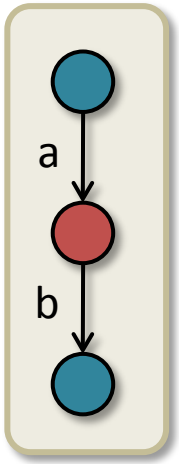
■ ライブロックの取り扱いに注意！

ABτ




$\neq_{FD}$

$=_{WB}$



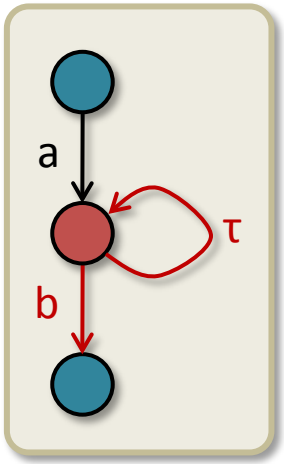

AB

$ABτ \neq_{FD} AB$

$ABτ =_{WB} AB$

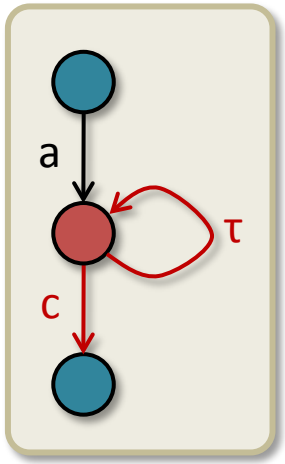
（失敗発散等価はライブロックを考慮する。
弱双模倣等価はライブロックを無視する。）

ABτ




$=_{FD}$

$\neq_{WB}$




ACτ

$ABτ =_{FD} ACτ$

$ABτ \neq_{WB} ACτ$

（失敗発散等価はライブロックをもつ
動作をほとんど区別できない。）

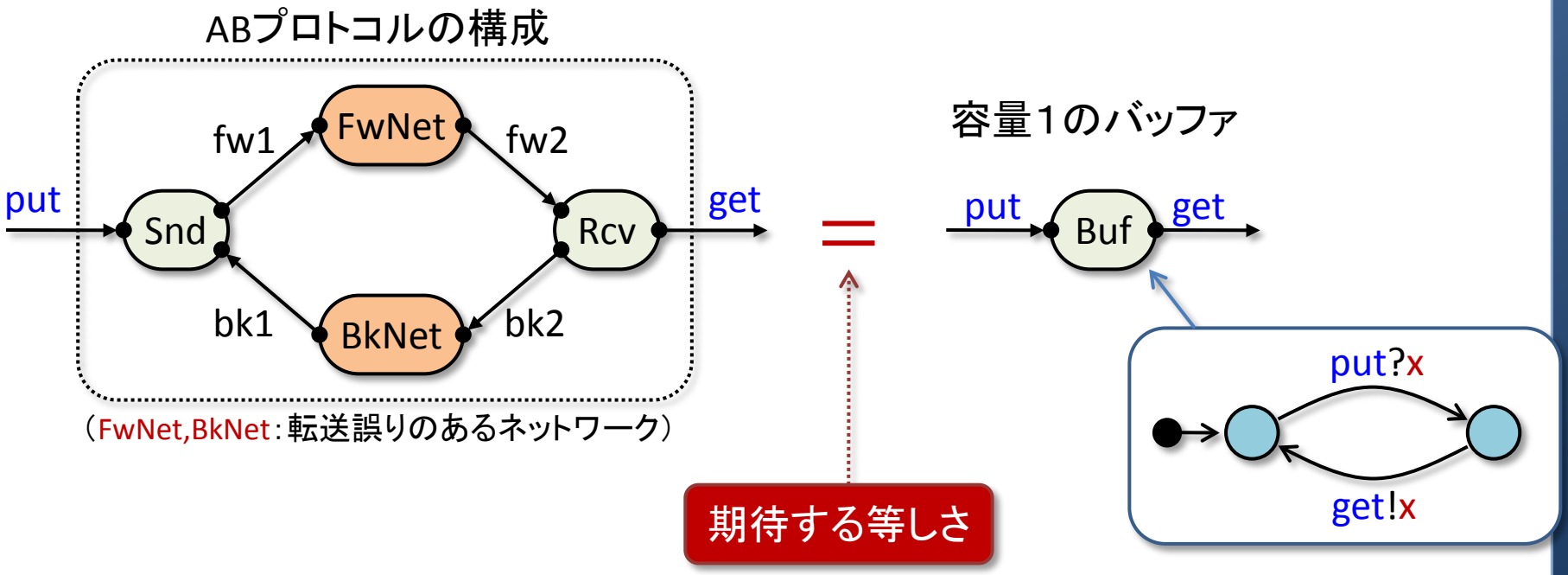
重要：失敗発散等価性の前にライブロックフリー性を確認すること

# 例題：ABプロトコル

- ABプロトコルの概要
- 各プロセスの動作

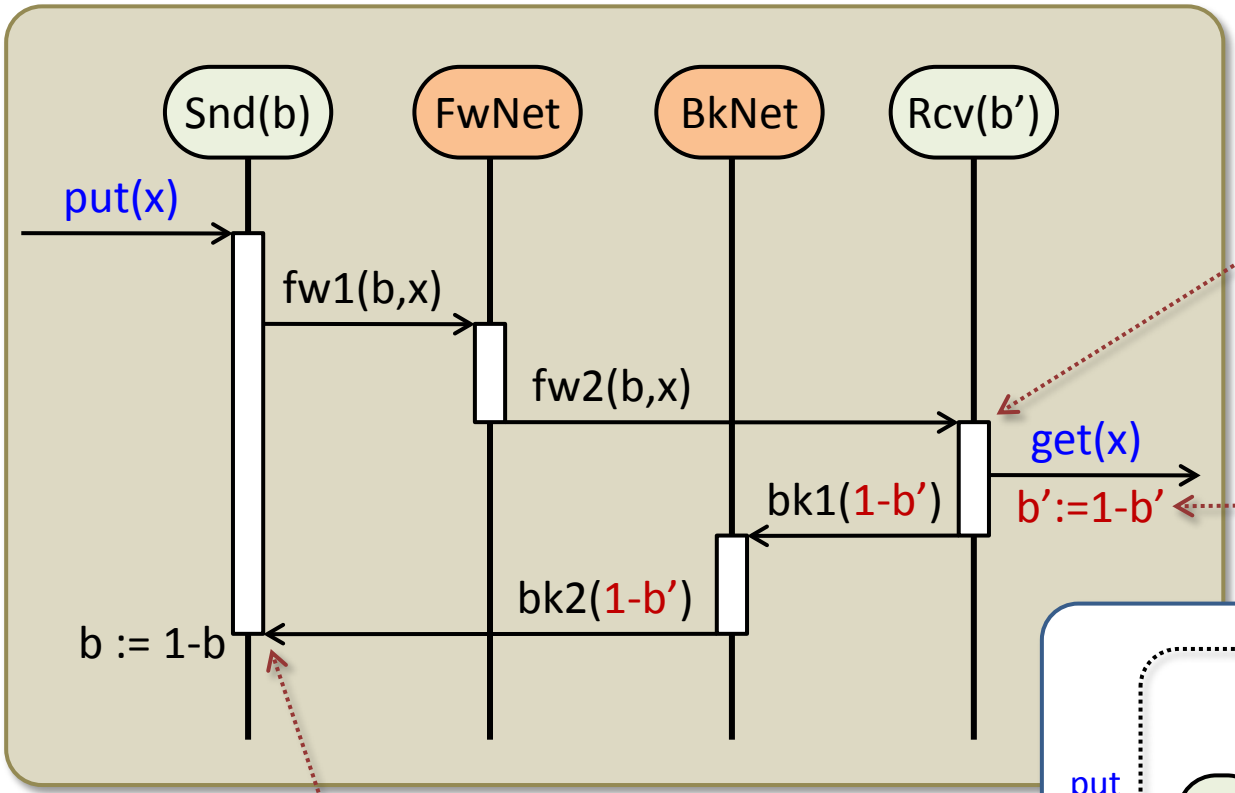
# ABプロトコル概要

■ **ABプロトコル（Alternating Bit Protocol）:**
転送誤り（データの消失と複製）をもつネットワークでデータを送受信するためのプロトコル



# ABプロトコル（動作例1）

■ 送信側と受信側のビット{0,1}を調べて正しいデータが複製データかをチェックする。

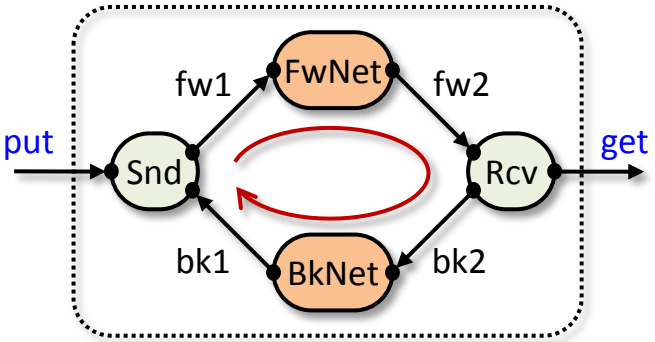



（初期状態ではb'=b）

b'=bなので受信成功

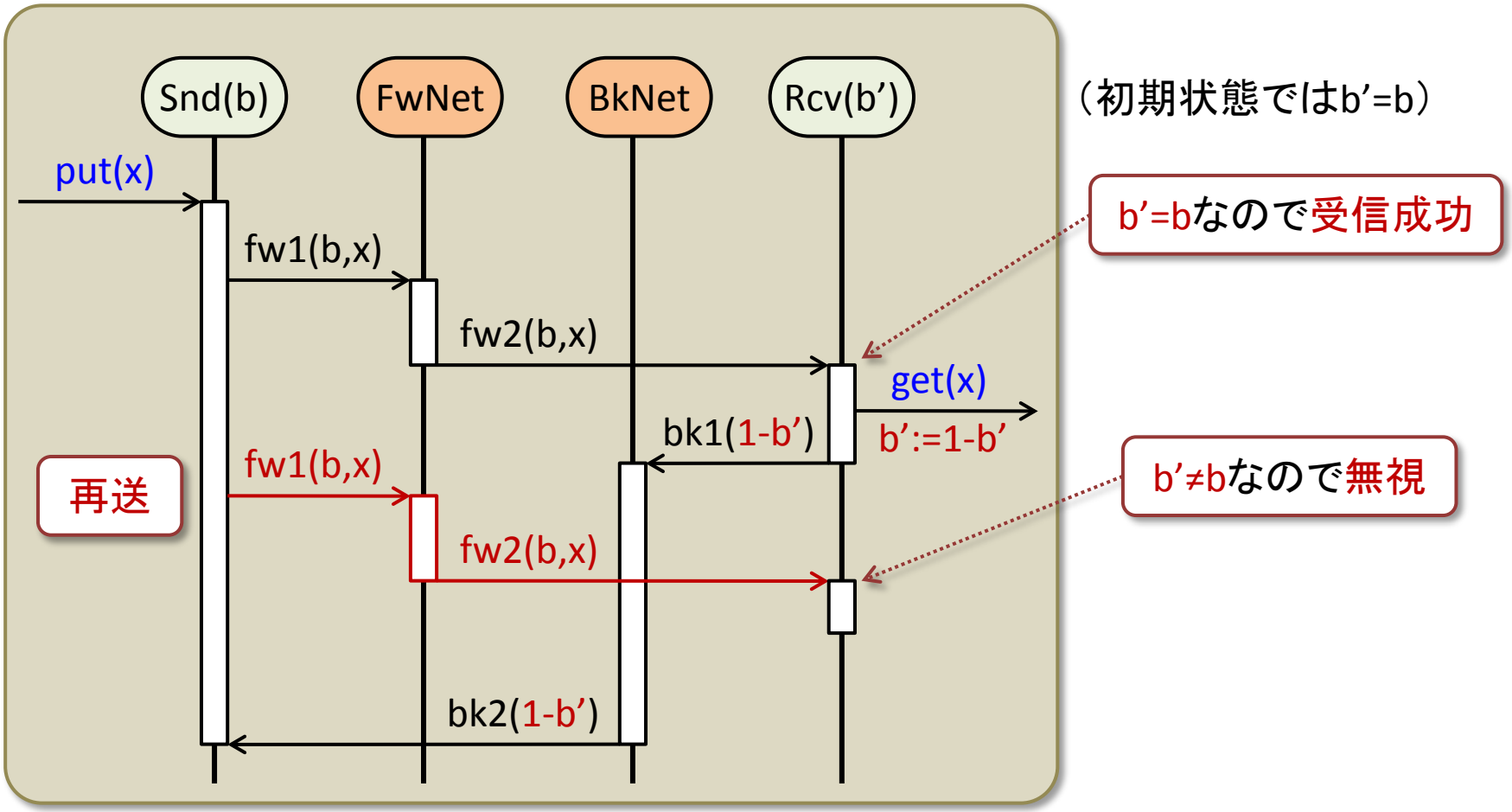
受信側のビット反転

1-b'=bなので転送完了
（送信側のビットも反転）



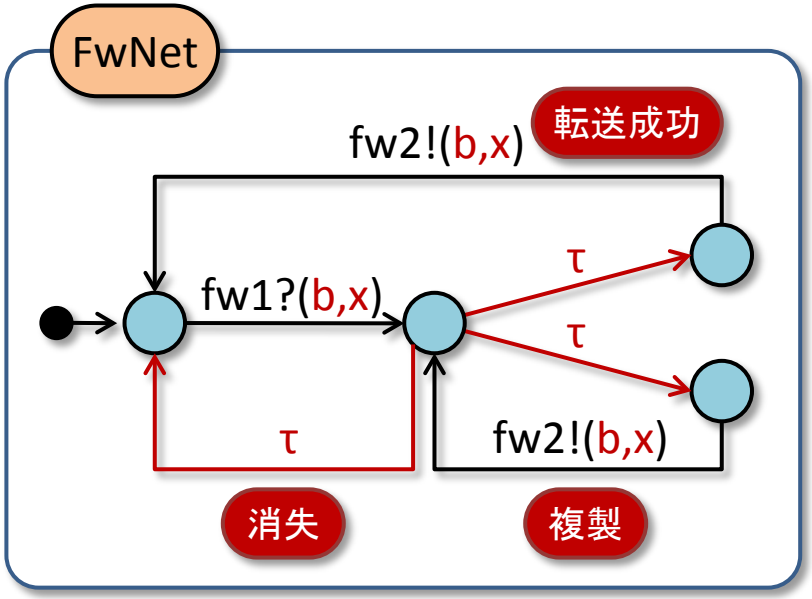
# ABプロトコル（動作例2）

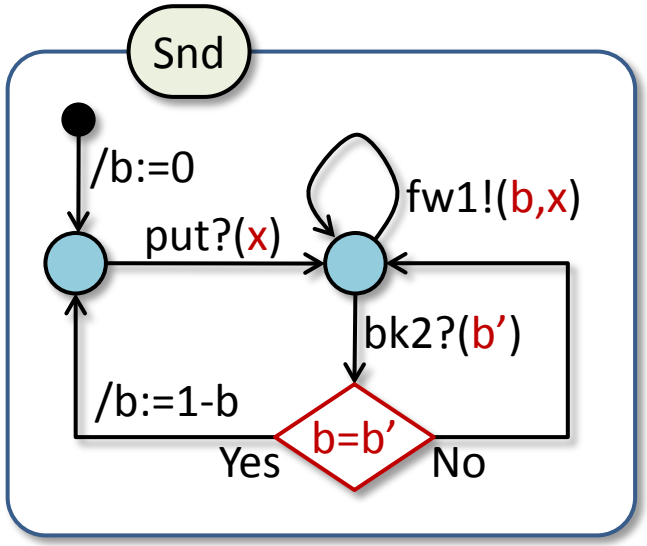
■ 送信側はデータ消失に備えて適時再送する（無駄な再送は無視される）。

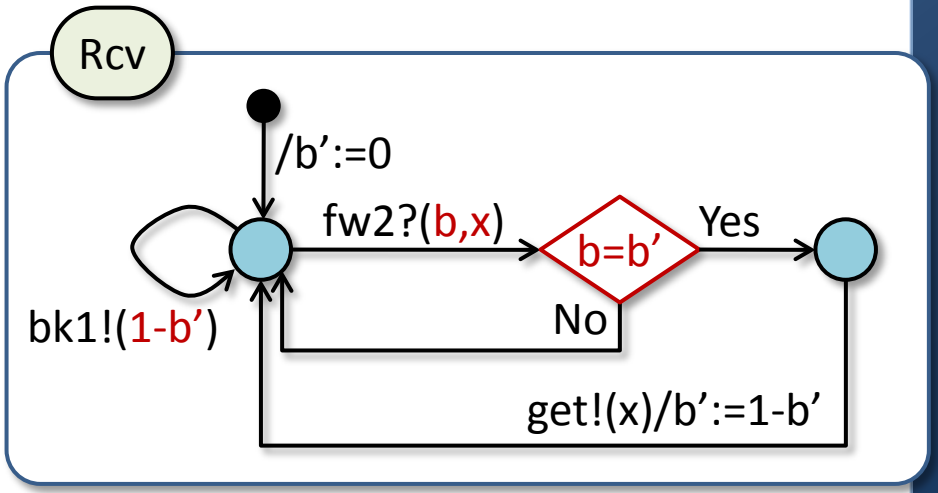


# ABプロトコル（各プロセスの動作）

■ FwNet（BkNetも同様）、Snd、Rcv の動作



# mCRL2による解析例

- mCRL2の特徴
- ABプロトコルのmCRL2記述
- 分岐双模倣等価の判定
- 状態遷移図の表示

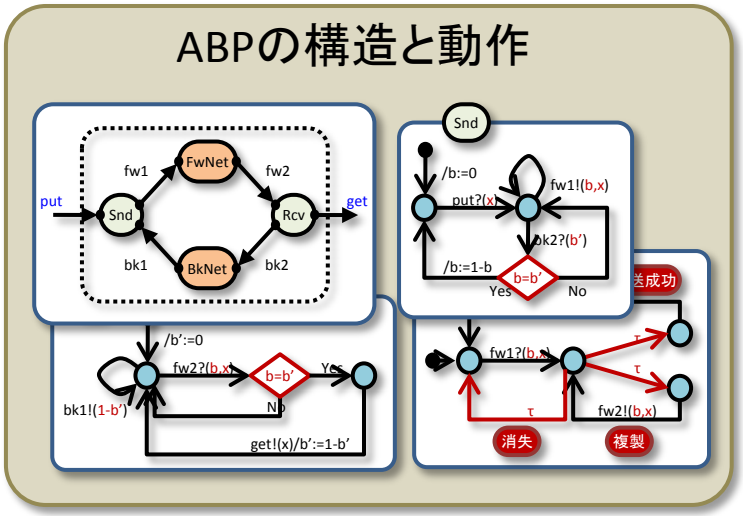
# mCRL2: モデル検査器

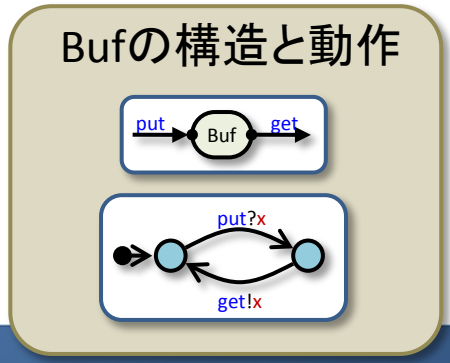
## mCRL2: プロセス代数ACPベースのモデル検査器

- 開発元: アイントホーフェン工科大学
- ライセンス: Boost license （フリー）
- URL: http://www.mcrl2.org/mcrl2/wiki/index.php/Home

特徴:

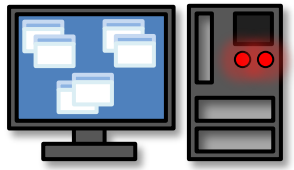
- ✓ 分岐双模倣等価性などの判定
- ✓ グラフィカルな状態遷移図の表示
- ✓ シミュレータによる1ステップごとの動作確認
- ✓ リアルタイムシステムの検証も可能



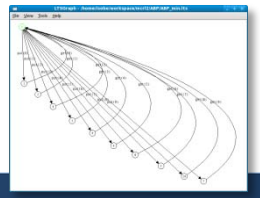

mCRL2 記述言語（ACP系）

mCRL2 記述言語（ACP系）




分岐双模倣等価性判定


状態遷移図



# mCRL2による記述

■ ABPの構造と動作をmCRL2言語（プロセス代数ACP系）で記述する。

```
% ------------------------------------------------------
%                    chennel
% ------------------------------------------------------

map Dmax : Nat;
eqn Dmax = 10;

act
  put, get     : Nat;
  ch1, ch2     : Bool # Nat;
  fwchs1, fwchr1, fwch1 : Bool # Nat;
  fwchs2, fwchr2, fwch2 : Bool # Nat;
  bkchs1, bkchr1, bkch1 : Bool # Nat;
  bkchs2, bkchr2, bkch2 : Bool # Nat;

% ------------------------------------------------------
%                  Lossy network
% ------------------------------------------------------

proc
 Net = sum b:Bool,x:Nat. (x<Dmax) -> ch1(b,x).NetX(b,x) <> delta;
 NetX(b:Bool,x:Nat) = tau.ch2(b,x).Net + tau.ch2(b,x).NetX(b,x) + tau.Net;

 FwNet = rename({ch1->fwchr1,ch2->fwchs2},Net);
 BkNet = rename({ch1->bkchr1,ch2->bkchs2},Net);
```

FwNet

BkNet

```
% ------------------------------------------------------
%                Sender & Reciver
% ------------------------------------------------------

proc
 Snd(b:Bool) = sum x:Nat. (x<Dmax) -> put(x). SndX(b,x) <> delta;
SndX(b:Bool,x:Nat) = fwchs1(b,x).SndX(b,x)
                   + bkchr2(b,0).Snd(!b)
                   + bkchr2(!b,0).SndX(b,x);

proc
 Rcv(b:Bool) = (sum x:Nat. fwchr2(b,x).RcvX(b,x))
             + (sum x:Nat. fwchr2(!b,x).Rcv(b))
             + bkchs1(!b,0).Rcv(b);

 RcvX(b:Bool, x:Nat) = get(x). Rcv(!b);

% ------------------------------------------------------
%                      ABP
% ------------------------------------------------------

 ABP = hide({fwch1,fwch2,bkch1,bkch2},
       allow({put,get,fwch1,fwch2,bkch1,bkch2},
        comm({fwchs1|fwchr1->fwch1, fwchs2|fwchr2->fwch2,
              bkchs1|bkchr1->bkch1, bkchs2|bkchr2->bkch2},
         Snd(true) || FwNet || BkNet || Rcv(true)
        )));

init
 ABP;
```

Snd(b)

Rcv(b)

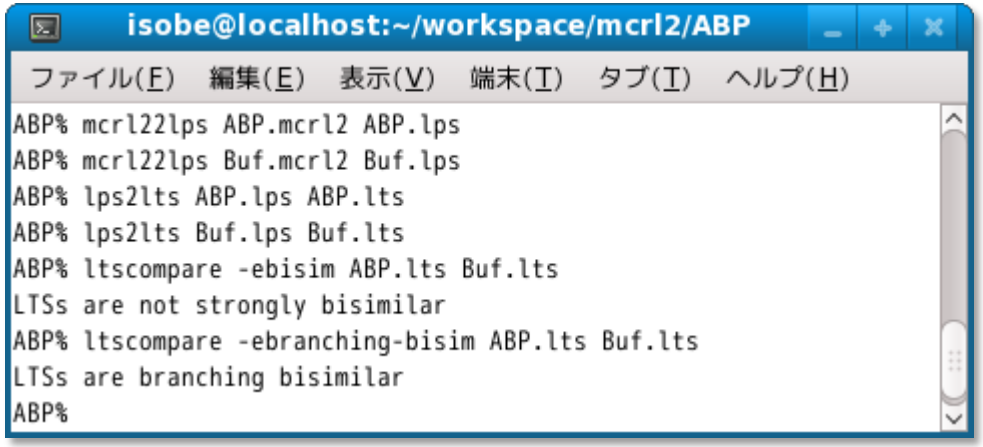
ABP

# mCRL2による等価性判定

- mCRL2コマンドによるABプロトコル（ABP）と容量1のバッファ（Buf）の等価性判定

$ABP \not\equiv_{SB} Buf$

$ABP \equiv_{BB} Buf$



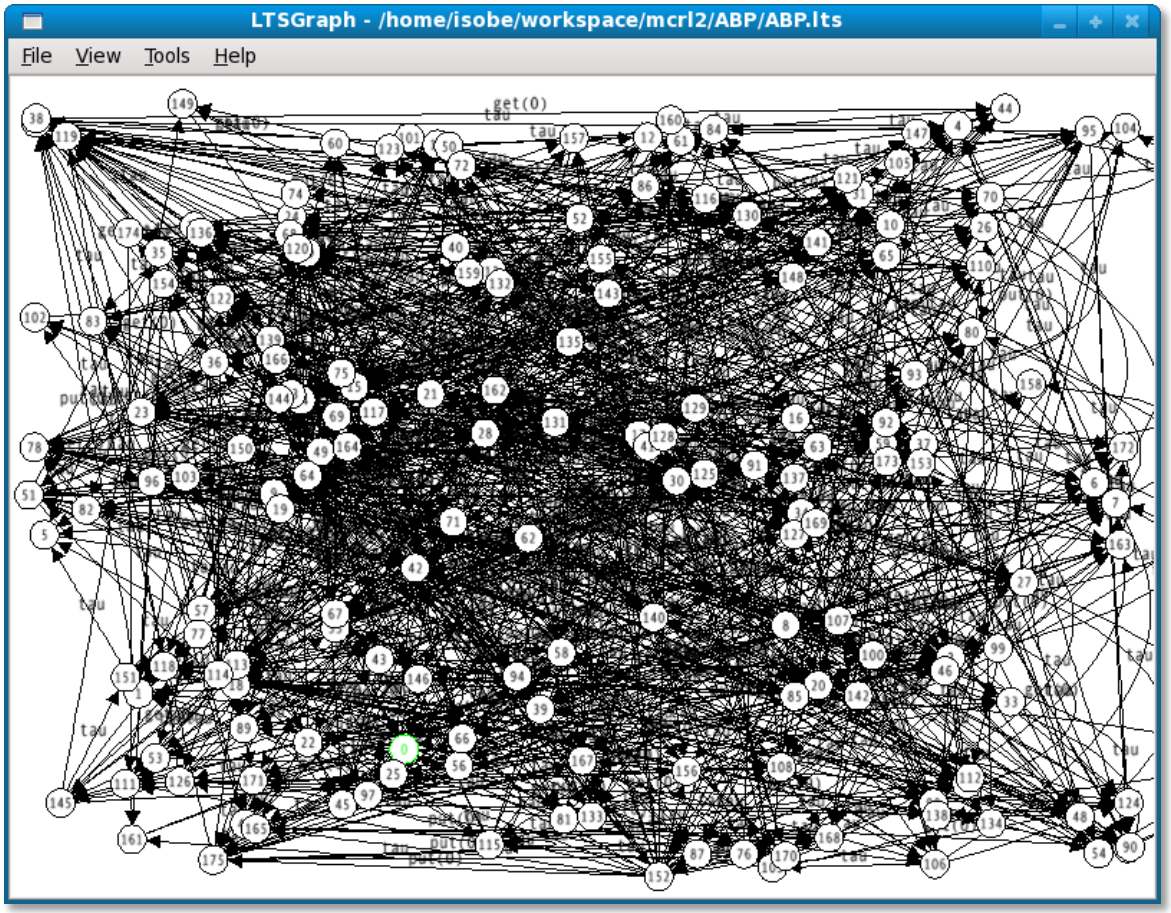

（転送するデータを{0,...,9}に制限して検証）

SB：強双模倣等価
BB：分岐双模倣等価

ABPの状態遷移図の情報
状態数：3848
遷移数：28248

# mCRL2による状態遷移図の表示1

- mCRL2によるABプロトコル（ABP）の状態遷移図の表示



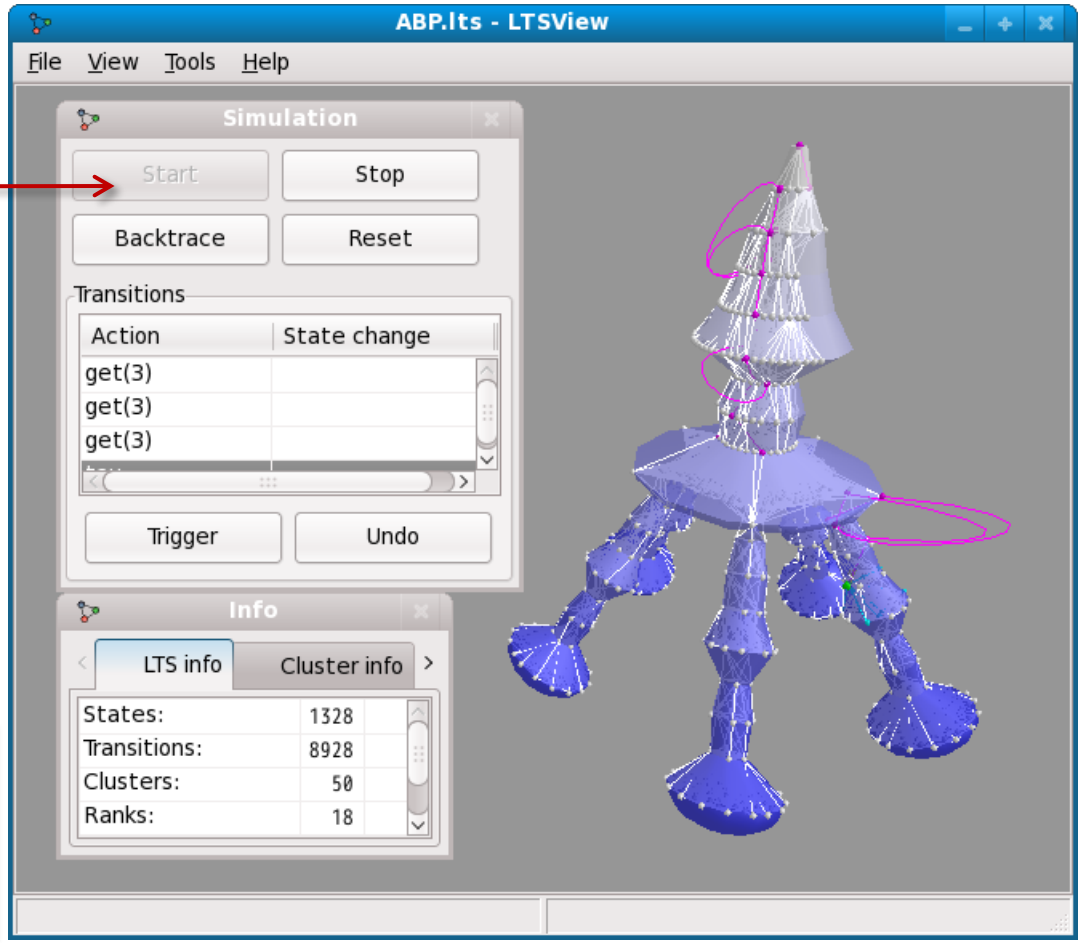

ABPの状態遷移図の情報
状態数：176
遷移数：960

（転送するデータが1種類{0}の場合でも…）

# mCRL2による状態遷移図の表示2

- mCRL2では状態遷移図を立体的に表示する機能がある。

シミュレータにより動作を1ステップごとに確認することができる

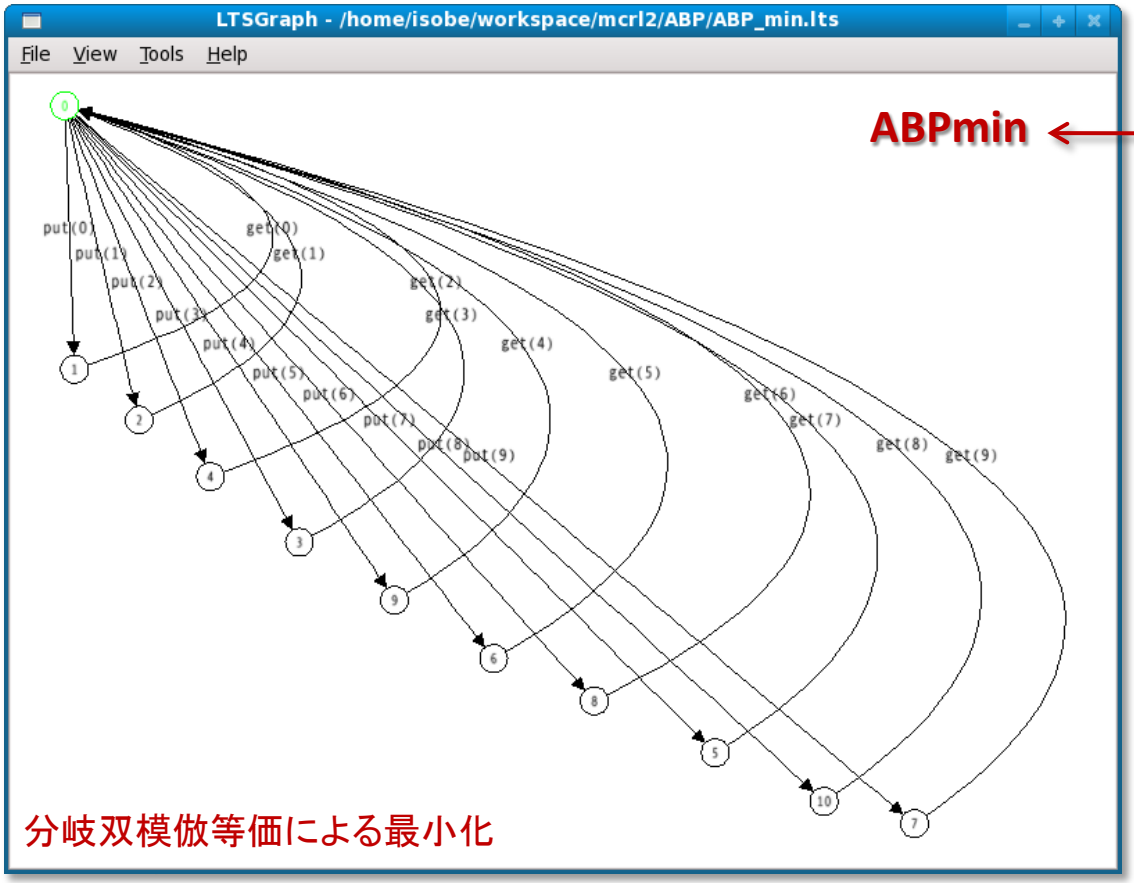



ABPの状態遷移図の情報
状態数: 1328
遷移数: 8928

（転送するデータが5種類{0,...,4}の場合）

# mCRL2による状態数最小化

- mCRL2によりABプロトコル（ABP）の状態数を最小化した結果の表示




ABPmin

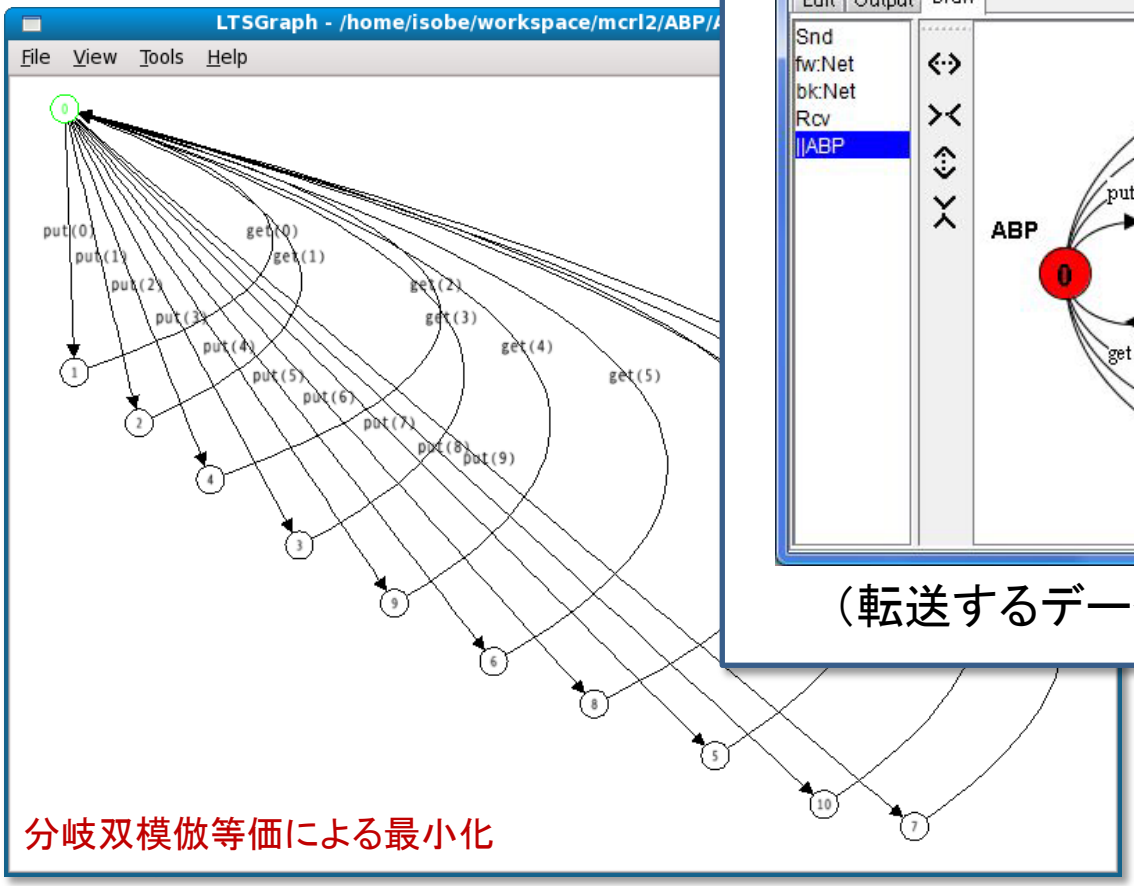
ABP と分岐双模倣等価で、状態数が最小の状態遷移図

$ABP =_{BB} ABPmin$

（転送するデータが10種類{0,...,9}の場合）

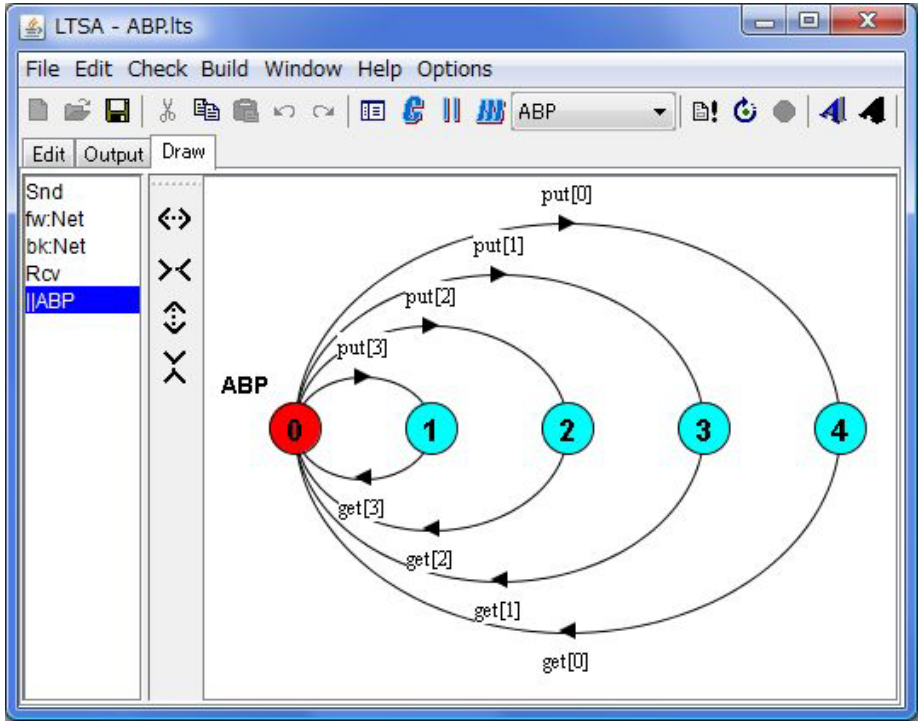
# mCRL2による状態数最小

- mCRL2によりABプロトコル（ABP）の




（転送するデータが10種類{0,...,9}の場合）

LTSA（Labelled Transition System Analyser）では
弱双模倣等価による状態数の最小化ができる。




（転送するデータが5種類{0,...,4}の場合）

# FDR2による解析例

- FDR2の特徴
- ライブロックの解消
- 失敗発散等価の判定

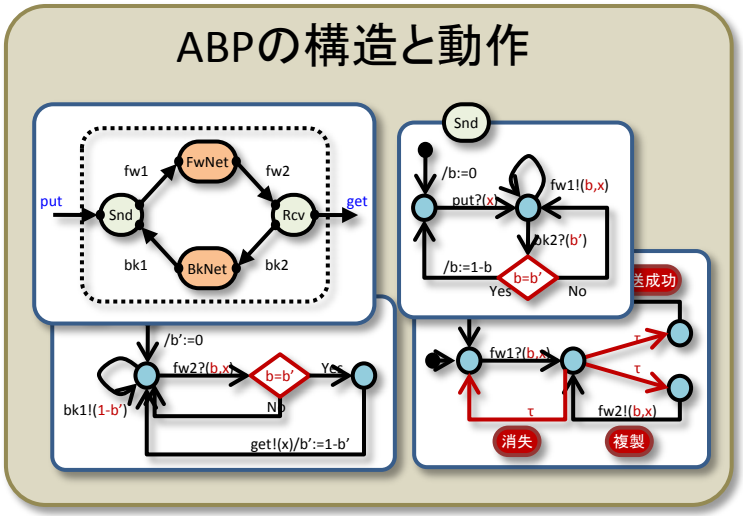
# FDR2: モデル検査器

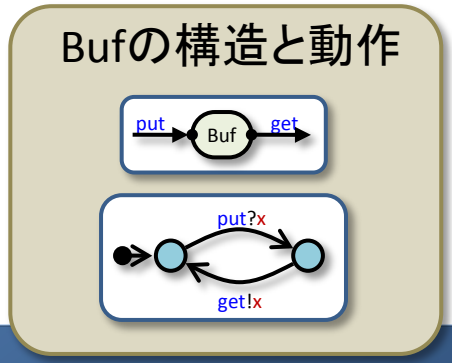
## FDR2: プロセス代数CSPベースのモデル検査器

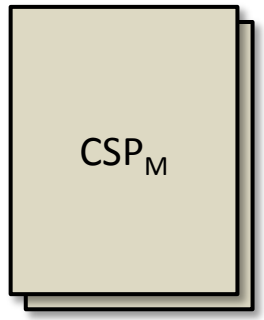
- 開発元：オクスフォード大学、Formal Systems (Europe) Ltd.
- ライセンス：アカデミック目的ならばフリー
- URL：http://web.comlab.ox.ac.uk/projects/concurrency-tools/

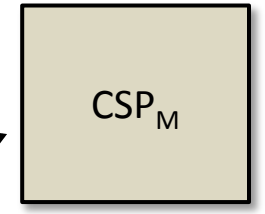
特徴：

✓ 失敗発散詳細化（等価性）などの判定
✓ グラフィカルなデバッガ
✓ ProBEによる1ステップごとの動作確認

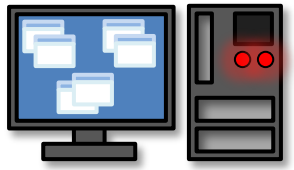



失敗発散等価性判定



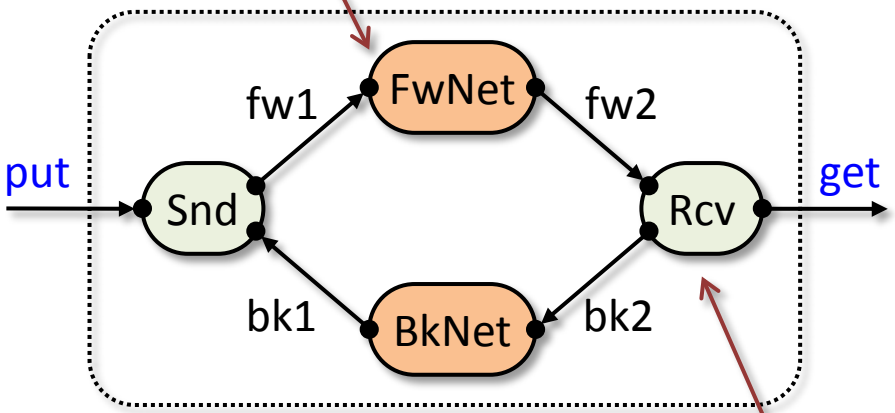
FDR2

# ABPの修正（ライブロックの解消）

- **ABプロトコル（ABP）と容量1のバッファ（Buf）は失敗発散等価ではない！**
  理由：ABPはライブロックをもつため。

  ⇒ ライブロックを防ぐ対策：

現在：無限回連続して転送に失敗する可能性がある。
修正：連続して失敗する回数を有限回に制限する。




公平性

現在：データを受信せずに返信のみ送り続けることができる。
修正：データの受信と返信を交互に実行するように制限する。

# $CSP_M$による記述

■ 修正したABPの構造と動作を$CSP_M$（Machine readable CSP）で記述する。

```
-------------------------------------------------
-- channel
-------------------------------------------------
Dmax =9
Nmax = 3

nametype Data = {0..Dmax}
nametype Bit = {0,1}

channel fw1, fw2 : Bit.Data
channel bk1, bk2 : Bit.Data
channel ch1, ch2 : Bit.Data

channel put, get : Data

-------------------------------------------------
-- Lossy channel
-------------------------------------------------
  Net(c) = ch1?x -> Net'(c,x)
Net'(c,x) = if (c<Nmax)
        then ch2!x -> Net(Nmax)
         |~| ch2!x -> Net'(c+1,x)
         |~| Net(c+1)
        else ch2!x -> Net(0)

Network = Net(0)[[ch1<-fw1,ch2<-fw2]] ||| Net(0)[[ch1<-bk1,ch2<-bk2]]
```

連続して失敗する最大数 3

FwNet　　BkNet

```
-------------------------------------------------
-- Sender and Recevier
-------------------------------------------------
  Snd(b) = put?x -> Snd'(b,x)
Snd'(b,x) = fw1!b.x -> Snd'(b,x)
      [] bk2?b'.x' -> (if (b==b') then Snd(1-b) else Snd'(b,x))

  Rcv(b) = fw2?b'.x -> (if (b==b') then get!x -> Rcv(1-b) else Rcv(b))
      [] bk1!(1-b).0 -> Rcv(b)

    Fair = fw2?x -> bk1?x -> Fair
FairRcv(b) = Rcv(b) [|{|fw2,bk1|}|] Fair
-------------------------------------------------
-- ABP
-------------------------------------------------
ABP = ((Snd(0) [|{|fw1,bk2|}|] Network)¥{|fw1,bk2|}
     [|{|fw2,bk1|}|]FairRcv(0))¥{|fw2,bk1|}

-------------------------------------------------
-- Specification
-------------------------------------------------
Buf = put?x -> get!x -> Buf

-------------------------------------------------
-- verification
-------------------------------------------------

assert Buf [FD= ABP
assert ABP [FD= Buf
assert ABP :[livelock free [FD]]
```

Snd(b)

Rcv(b)

受信と返信の公平性

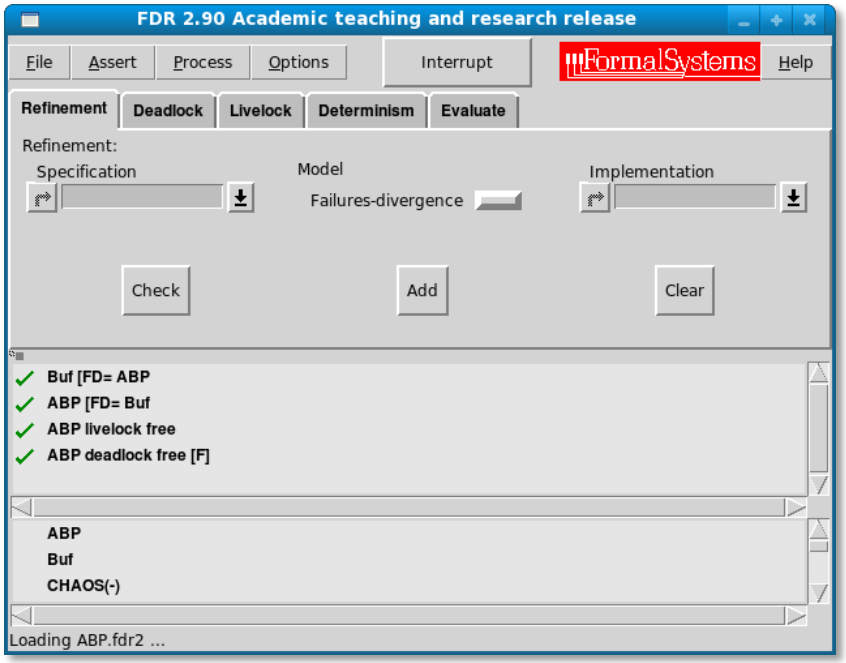
ABP

# FDR2による検証

- FDR2によるABプロトコル（修正版ABP）と容量1のバッファ（Buf）の等価性判定

$ABP =_{FD} Buf$

失敗発散等価




$$ABP =_{FD} Buf \Leftrightarrow ABP \sqsubseteq_{FD} Buf \wedge Buf \sqsubseteq_{FD} ABP$$

失敗発散等価関係

失敗発散詳細化関係

# π計算

- π計算の特徴
- チャネル渡しの例
- MWBによる検証例

# π計算

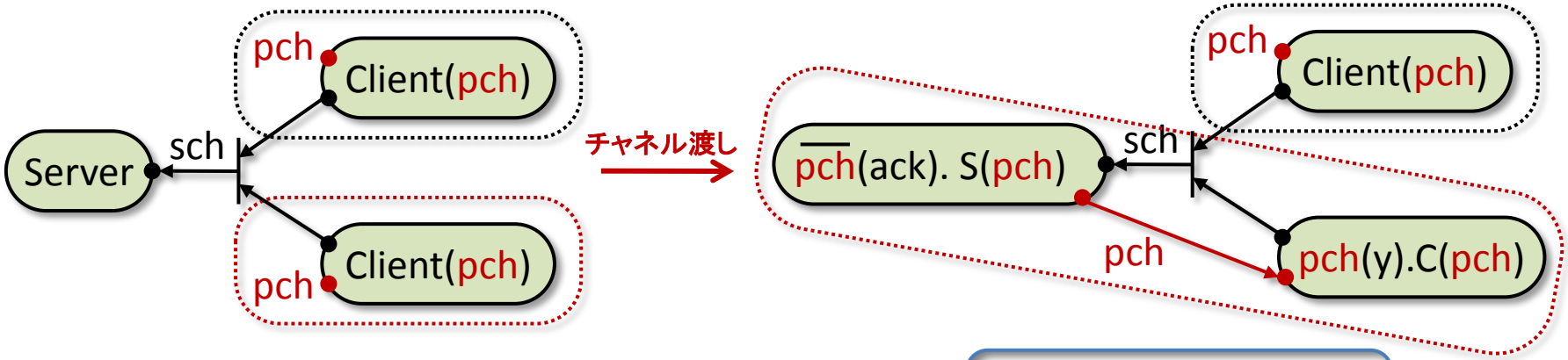
- π計算：チャネル渡しができるようにCCSを拡張したプロセス代数
  - チャネル渡し：チャネル名を渡して新しいチャネルを動的に生成すること

Sys = Server | (ν pch) Client | (ν pch) Client

Server = sch(x).$\overline{x}$(ack). S(x)

Client(pch) = $\overline{sch}$(pch). pch(y).C(pch)

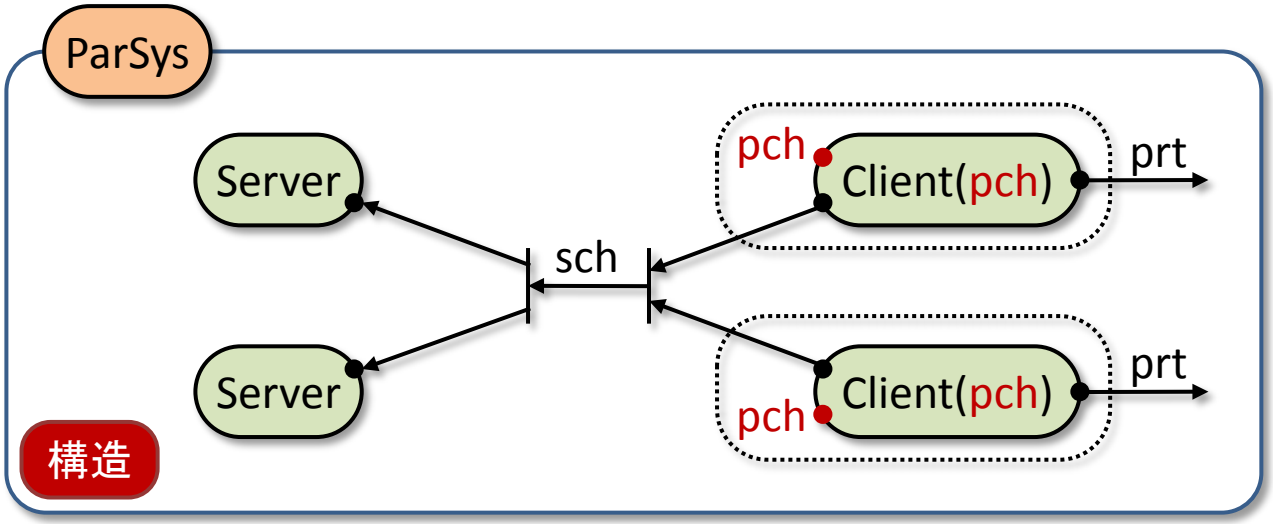
1. 共用チャネルschによるチャネル渡し
2. 専用チャネルpchによる返信

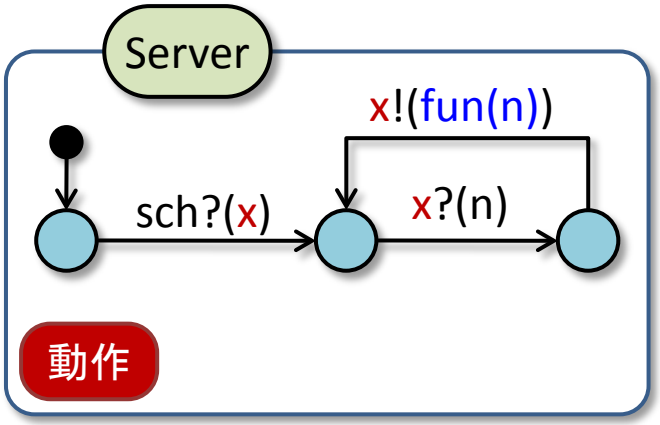


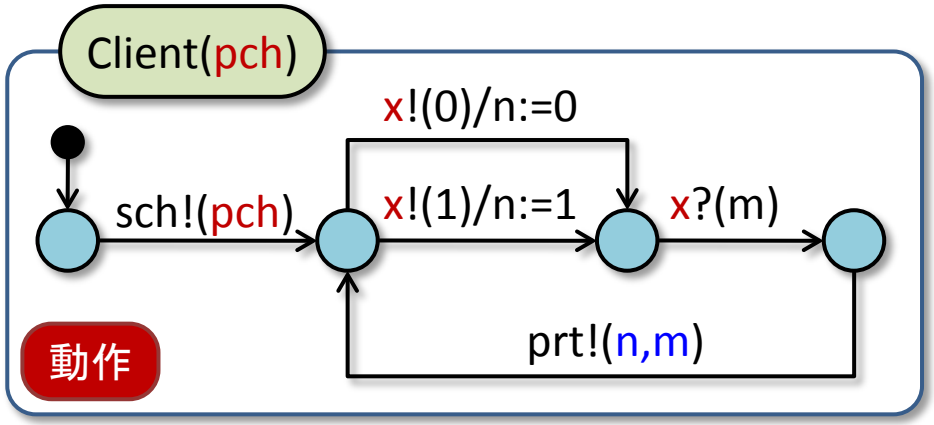

pchの制限範囲が広がる！

# ParSys: チャネル渡しの例

- **ParSys**：各クライアントは1つのサーバと専用チャネルを生成し、計算を依頼する。

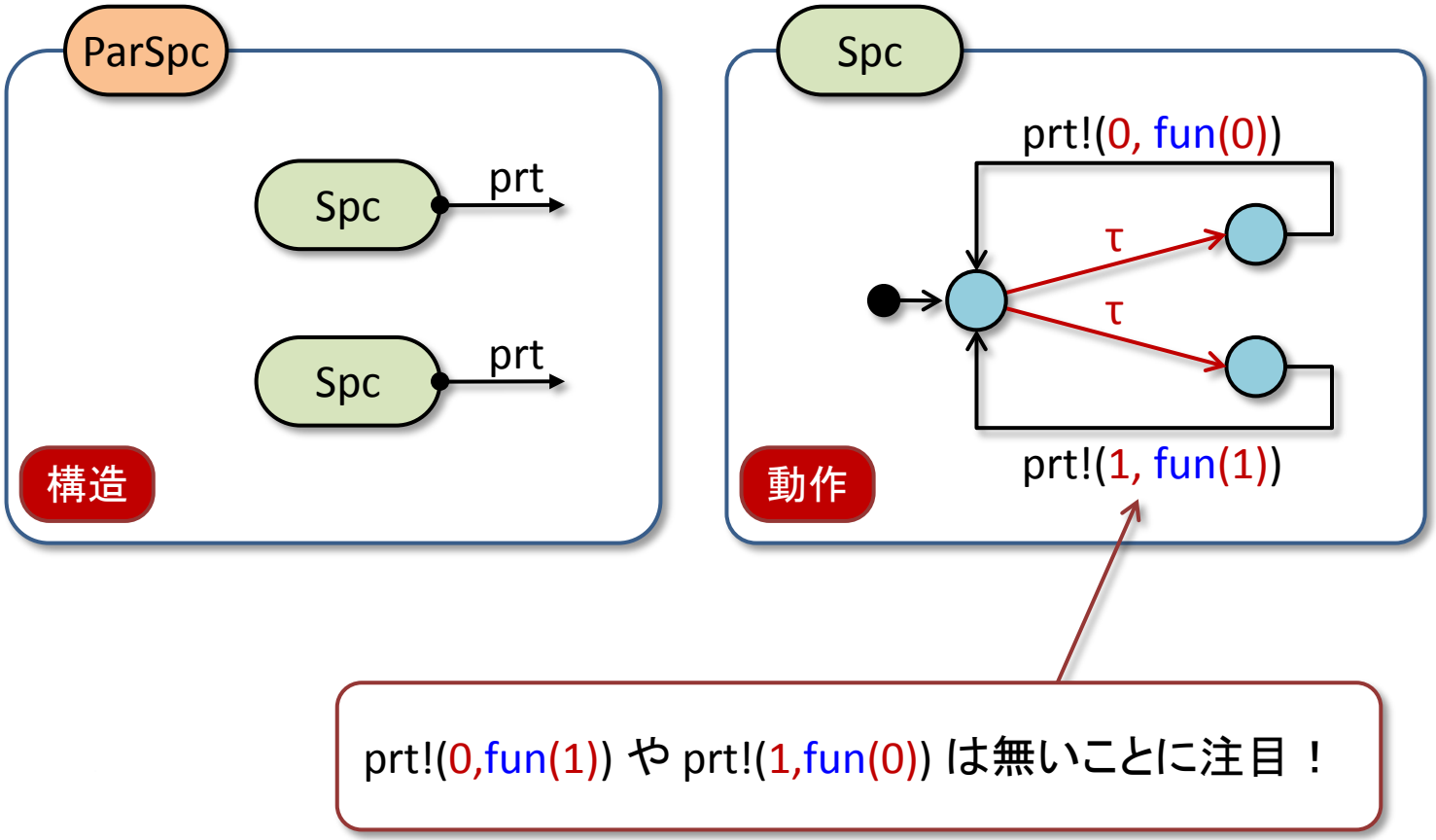


# ParSysの仕様

- **ParSpc** : Clientが自分で関数計算する場合（通信がないので簡単）。

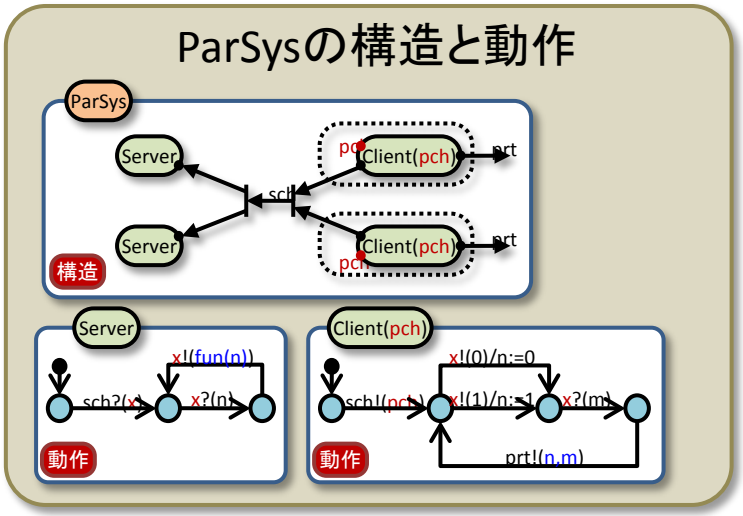


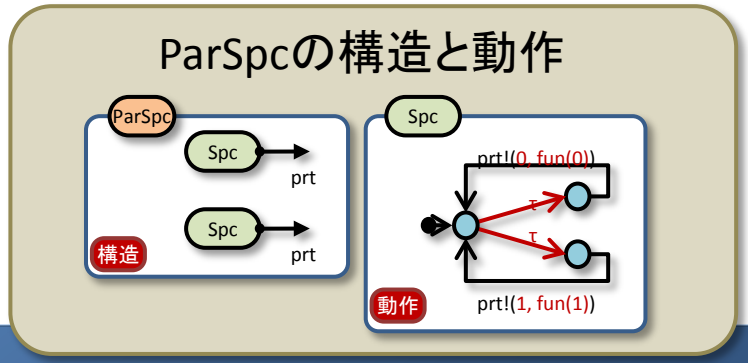
# MWB（Mobility Workbench）: モデル検査器

- **MWB: プロセス代数（π計算）ベースのモデル検査器**
  - 開発元：ウプサラ大学
  - URL：http://www.it.uu.se/research/group/mobility/mwb

特徴：

- ✓ 弱双模倣等価などの判定
- ✓ チャネル渡しを表現可能
- ✓ 記号処理による解析

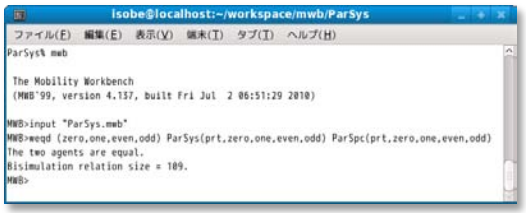


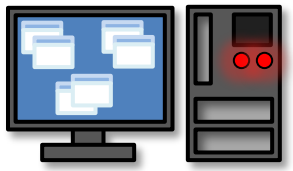




π計算

π計算

弱双模倣等価性判定



MWB

# MWBによる記述

- ParSysとParSpcの構造と動作をπ計算（MWB）で記述する。

```
(* ----- Client ----- *)

agent Client(pch,sch,prt,zero,one) ¥
  = 'sch<pch>.CLoop(pch,sch,prt,zero,one)

agent CLoop(pch,sch,prt,zero,one) ¥
  = 'pch<zero>.CWait(pch,sch,prt,zero,one,zero) ¥
  + 'pch<one >.CWait(pch,sch,prt,zero,one,one)

agent CWait(pch,sch,prt,zero,one,n) ¥
  = pch(m).'prt<n,m>.CLoop(pch,sch,prt,zero,one)

agent Clients(sch,prt,zero,one) ¥
  = (^pch)Client(pch,sch,prt,zero,one) ¥
  | (^pch)Client(pch,sch,prt,zero,one)

(* ----- Server ----- *)

agent Server(sch,zero,one,even,odd) ¥
  = sch(x).SLoop(x,sch,zero,one,even,odd)

agent SLoop(x,sch,zero,one,even,odd) ¥
  = x(n).(  [n=zero] 'x<even>.SLoop(x,sch,zero,one,even,odd)
          + [n=one ] 'x<odd>.SLoop(x,sch,zero,one,even,odd))

agent Servers(sch,zero,one,even,odd) ¥
  = Server(sch,zero,one,even,odd) ¥
  | Server(sch,zero,one,even,odd)
```

Client(pch)

Server

ParSys

```
(* ----- System ----- *)

agent ParSys(prt,zero,one,even,odd) ¥
  = (^sch) (Clients(sch,prt,zero,one) | Servers(sch,zero,one,even,odd))

(* ----- specification ----- *)

agent Spc(prt,zero,one,even,odd) ¥
  = t.'prt<zero,even>.Spc(prt,zero,one,even,odd) ¥
  + t.'prt<one ,odd >.Spc(prt,zero,one,even,odd)

agent ParSpc(prt,zero,one,even,odd) ¥
  = Spc(prt,zero,one,even,odd) ¥
  | Spc(prt,zero,one,even,odd)
```

Spc

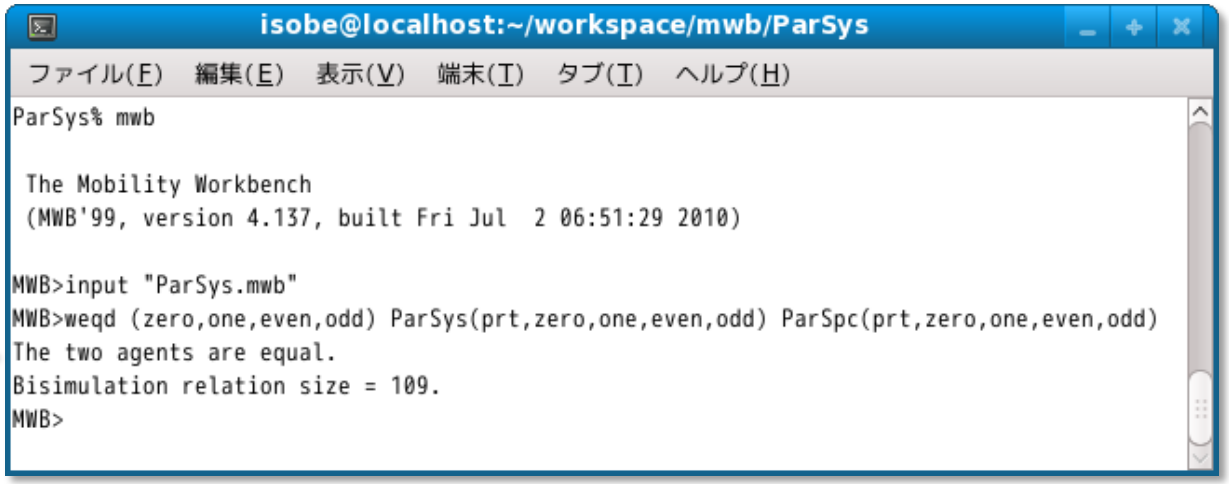
ParSpc

この例では次のような簡単な関数を計算している。

fun(0) = even
fun(1) = odd

# MWBによる検証

- MWBによるParSysとParSpcの弱双模倣等価性の判定




$ParSys =_{WB} ParSpc$ 弱双模倣等価

補足：π計算では変数の具体化の方法によっていくつかの弱双模倣が存在する。
正確にはMWBでは open weak bisimulation equiv. を判定する。

# まとめ

- 各ツールの情報
- 等しさの強さ

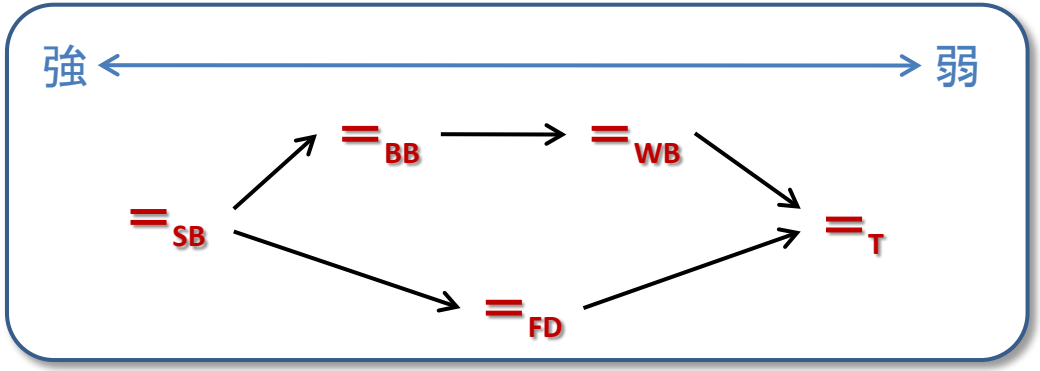
# まとめ

CSP：Communicating Sequential Process
CCS：Calculus of Communicating Systems
ACP：Algebra of Communicating Process

■ “動作の等しさ”判定ツールの例

| ツール | 理論 | 等価性 | URL |
|---|---|---|---|
| FDR | CSP | T, FD, etc | http://web.comlab.ox.ac.uk/projects/concurrency-tools/ |
| PAT | CSP# | T, FD, etc | http://www.comp.nus.edu.sg/~pat/ |
| mCRL2 | ACP | SB, BB, etc | http://www.mcrl2.org/mcrl2/wiki/index.php/Home |
| CWB | CCS | SB, WB, etc | http://homepages.inf.ed.ac.uk/perdita/cwb/ |
| MWB | π計算 | SB, WB, etc | http://www.it.uu.se/research/group/mobility/mwb |
| LTSA | FSP | WB (min) | http://www.doc.ic.ac.uk/ltsa/ |

■ 等価性の強さ（強いとより多くを区別する）




T：トレース等価
FD：失敗発散等価
WB：弱双模倣等価
BB：分岐双模倣等価
SB：強双模倣等価

# 付録: なぜプロセス“代数”？

- 代数による証明
- プロセス代数による証明

# “等しさ”の証明（代数）

■ 分配則や交換則を適用して書換えにより等式を証明できる。

**例**

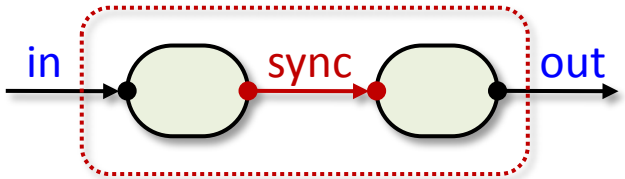
$$(a + b)(a + c) = a^2 + (b + c)a + bc$$

**証明**

$$\begin{aligned}
(a + b)(a + c) &= a(a + c) + b(a + c) && \text{（分配則）}\\
&= (a^2 + ac) + (ba + bc) && \text{（分配則）}\\
&= a^2 + (ac + ba) + bc && \text{（結合則）}\\
&= a^2 + (ca + ba) + bc && \text{（交換則）}\\
&= a^2 + (ba + ca) + bc && \text{（交換則）}\\
&= a^2 + (b + c)a + bc && \text{（分配則）}
\end{aligned}$$

**分配則** $x(y + z) = xy + xz$

# “等しさ”の証明(プロセス代数)




■ プロセス代数の規則を適用して書換えにより等式を証明できる。

**例**

(in?x → sync!x → STOP **[|{|sync|}|]** sync?x → out!x → STOP)＼**{|sync|}**
= in?x → out!x → STOP

**証明**

(in?x → sync!x → STOP **[|{|sync|}|]** sync?x → out!x → STOP)＼**{|sync|}**
= (in?x → (sync!x → STOP **[|{|sync|}|]** sync?x → out!x → STOP))＼**{|sync|}** （並列則）
= in?x → (sync!x → STOP **[|{|sync|}|]** sync?x → out!x → STOP)＼**{|sync|}** （隠蔽則）
= in?x → (sync!x → (STOP **[|{|sync|}|]** out!x → STOP))＼**{|sync|}** （同期則）
= in?x → (STOP **[|{|sync|}|]** out!x → STOP)＼**{|sync|}** （隠蔽則）
= in?x → (out!x → (STOP **[|{|sync|}|]** STOP))＼**{|sync|}** （並列則）
= in?x → out!x → (STOP **[|{|sync|}|]** STOP)＼**{|sync|}** （隠蔽則）
= in?x → out!x → STOP＼**{|sync|}** （停止則）
= in?x → out!x → STOP （停止則）

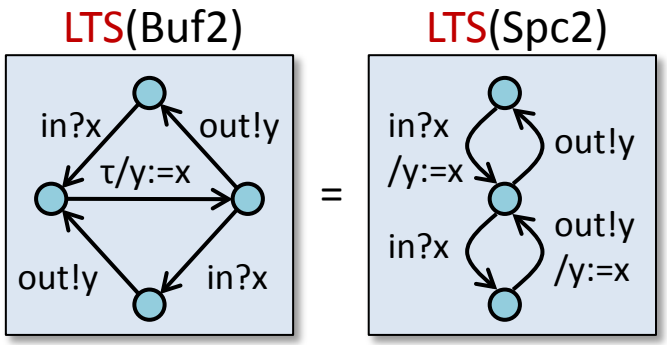
**同期則** (c!v→P) **[|{|c|}|]** (c?x→Q(x)) = c!v→ (P **[|{|c|}|]** Q(v))

# “等しさ”を証明(定義)する3つの方法

領域(表示的意味論)

traces(Buf2) = { ⟨in?0, out!0⟩, ⟨in?0, in?1, out!0, out!1⟩, ... } = traces(Spc2)

状態遷移図(操作的意味論)




モデル検査器

書換え(公理的意味論)

書換え　書換え　書換え

Buf2 = in?x→Buf2'(x) = ⋯ = Spc2

定理証明器