プロセス代数CSPのモデル検査器FDRの紹介

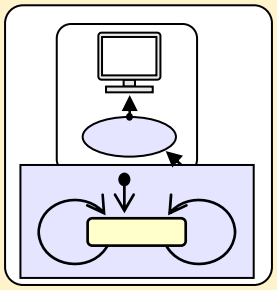
磯部 祥尚

産業技術総合研究所AIST（ベンチャー開発センターに出向中）＆
国立情報学研究所NII（トップエスイーでCSP関係の講座を担当）

第1回CSP研究会　2009年3月6日

講義：並行システムの検証と実装

逐次システムや仕様



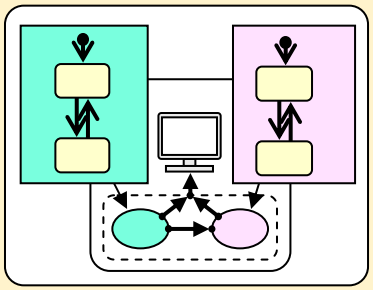
モデル化

プロセス代数CSP
によるモデル化

CSPモデル

```
SRseq(K) = SendRec(0,0,0,K)
SendRec(n,m,i,K)
= i<K & disp!send.n
  → SendRec((n+1)%N,m,i+1,K) □
  i>0 & disp!receive.m
  → SendRec(n,(m+1)%N,i-1,K)
```

並行システムや仕様



モデル化

CSPモデル

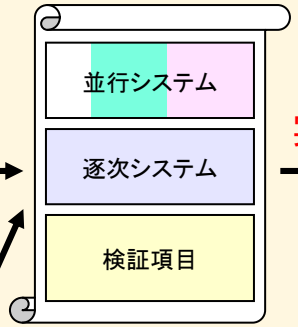
```
SRconc
= (Sender(0) [| {| chan |} |]
   Receiver(0))＼{| chan |}

Sender(n)
 = disp!send.n → Sender'(n)
Sender'(n)
 = chan!n → Sender((n+1)%N)

Receiver(m)
= chan?m → Receiver'(m)
Receiver'(m)
= disp!receive.m → Receiver(m)
```

検証

実装

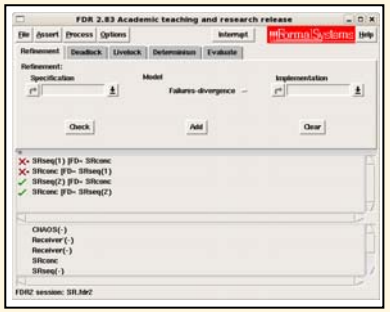
モデル検査器
FDRによる検証

FDRスクリプト

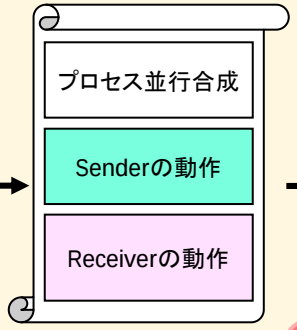


実行

FDR

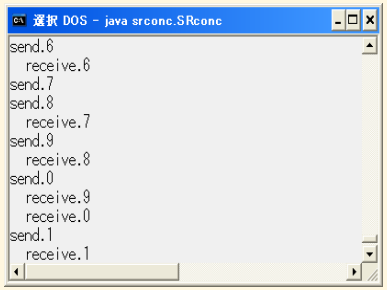


JCSPプログラム



実行

Java



Javaライブラリ
JCSPによる実装

講演内容

(1) この講演で用いる例題（リーダー決定問題）の紹介

(2) CSP（FDR）によるモデル化の例

(3) FDRによる検証例

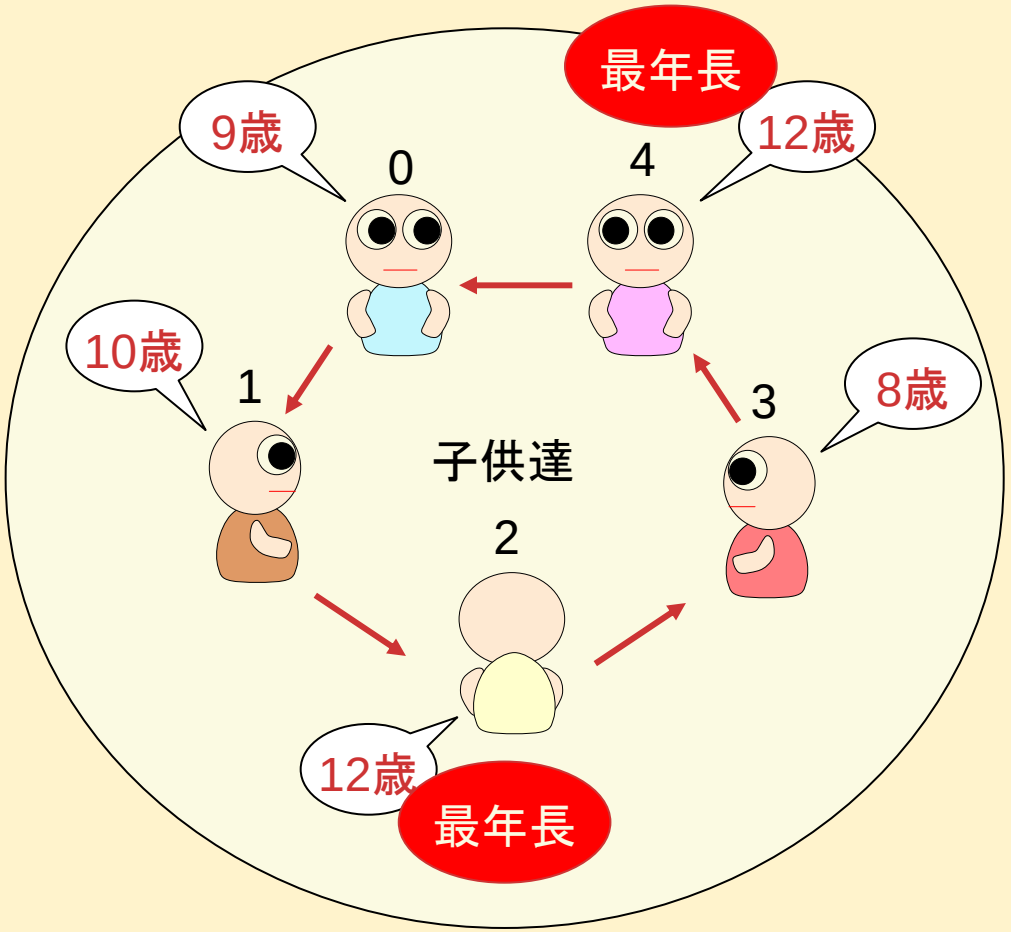
(4) FDRによるデバッグの例

(5) JCSPによる実装例

リーダー決定問題

リーダー決定問題（最年長者は誰？）

問題：輪になって座ったM人の子供たちから最年長者（複数可）を探す。



条件

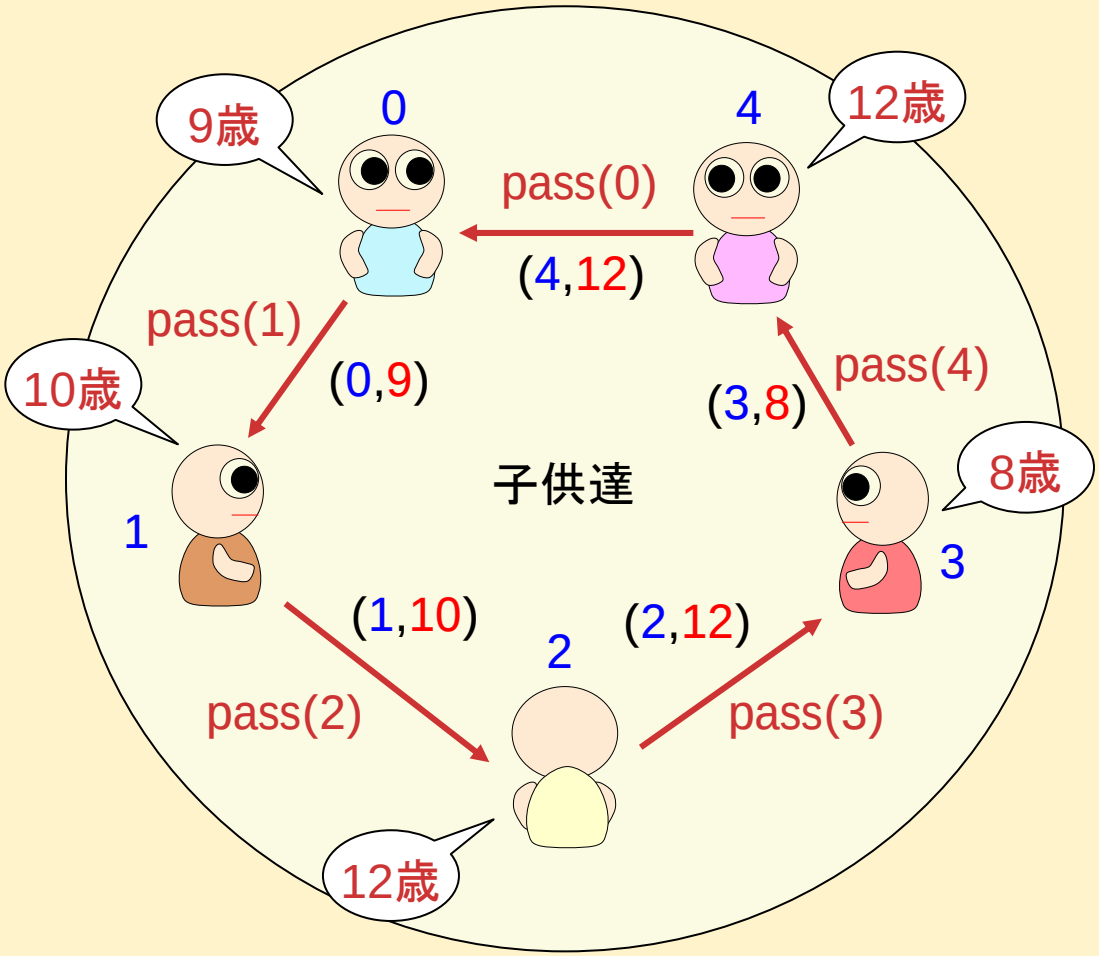
(1) 各子供は右隣の子供にのみ話しかけることができる。

(2) 各子供は自分の名前（ID）と年齢の他に、同時に2人まで他人の名前（ID）と年齢を覚えることができる。

(3) 大小の判断はできる。

チャネルの設定

子供たちは一方向のpassチャネルでつながれている。

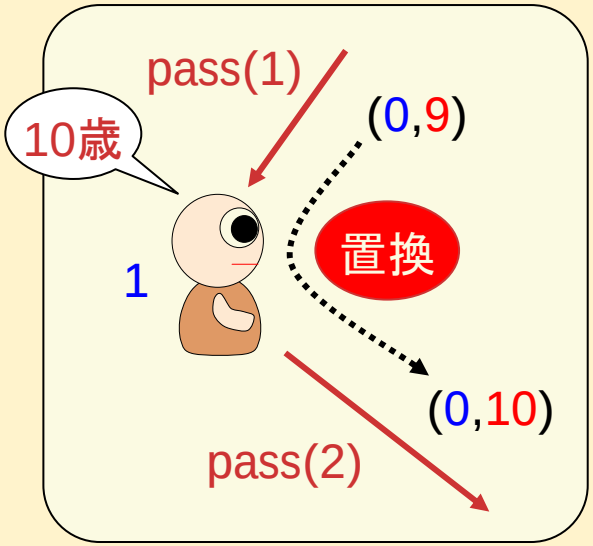


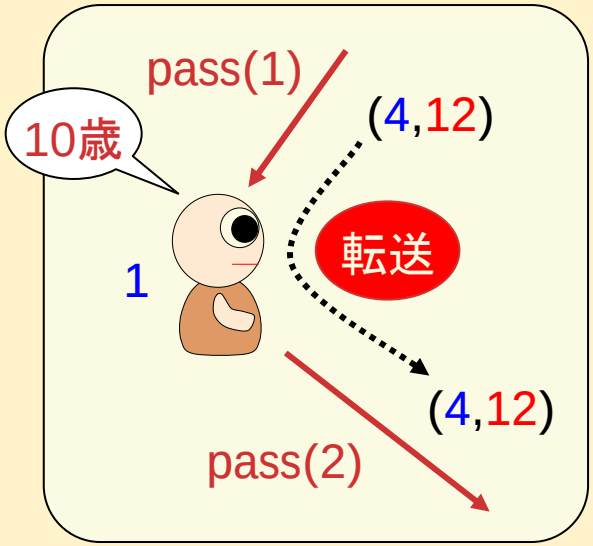
- IDがidの左側のチャネル: pass(id)
- IDがidの右側のチャネル: pass((id+1)%N)
- passで送受信するデータ: （ID, 年齢）

データの転送方法

子供たちはある規則に従ってデータ転送を繰り返す。

- 受取った年齢が自分より上ならばそのまま転送。
- 受取った年齢が自分より下ならば自分の年齢に置き換えて転送。

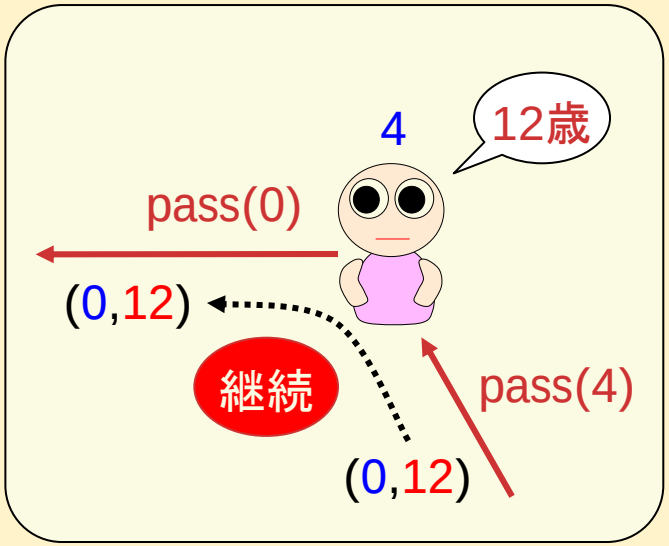


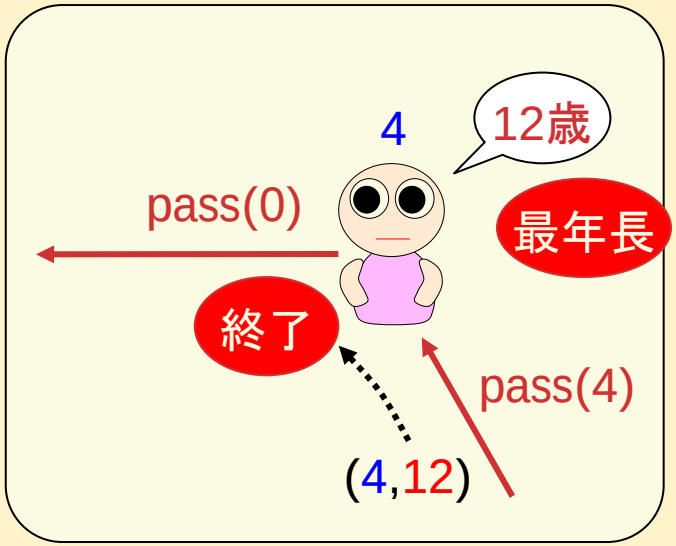


データの転送方法

子供たちはある規則に従ってデータ転送を繰り返す。

- 受取ったIDが自分のIDでないならば転送を継続。
- 受取ったIDが自分のIDならば転送を終了。
  ⇒このとき受け取った年齢が自分と同じならばこれが最大値。

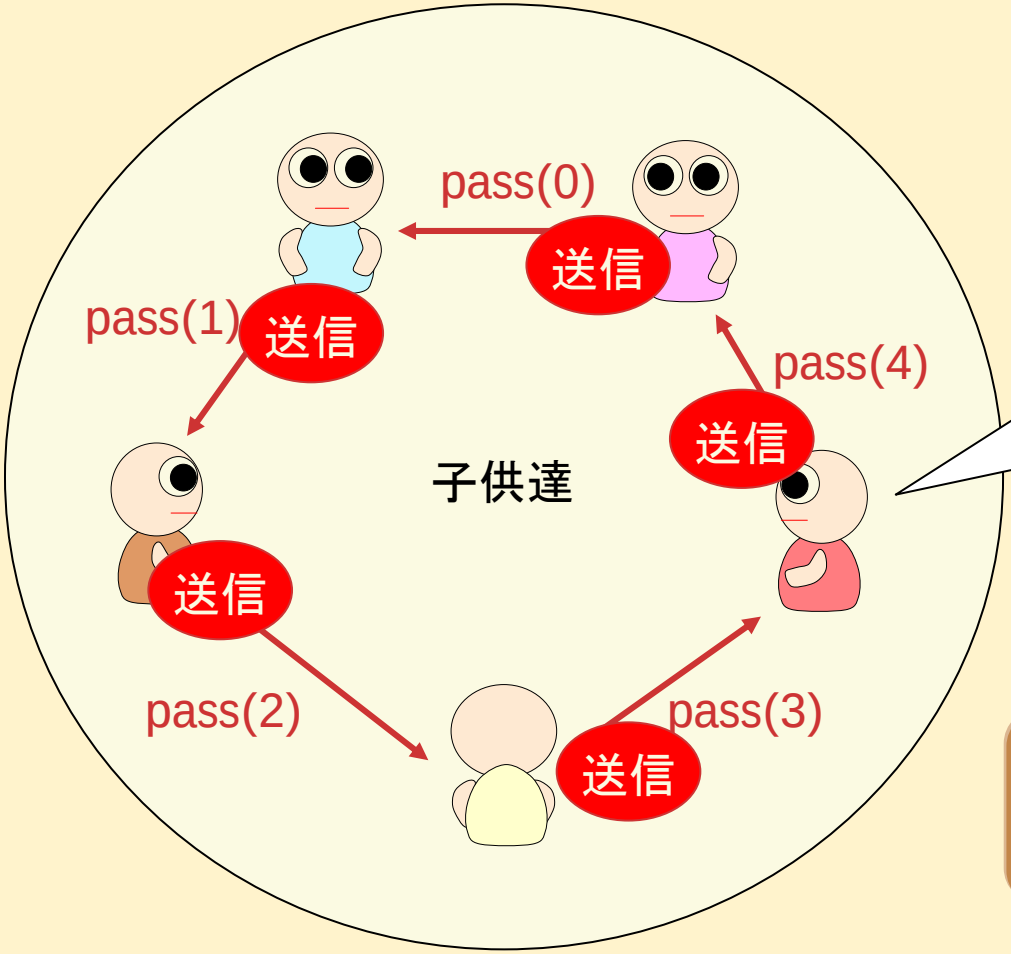




モデル化

各子供の動作

全員が送信してから受信をしようとすると何もできない（デッドロック）

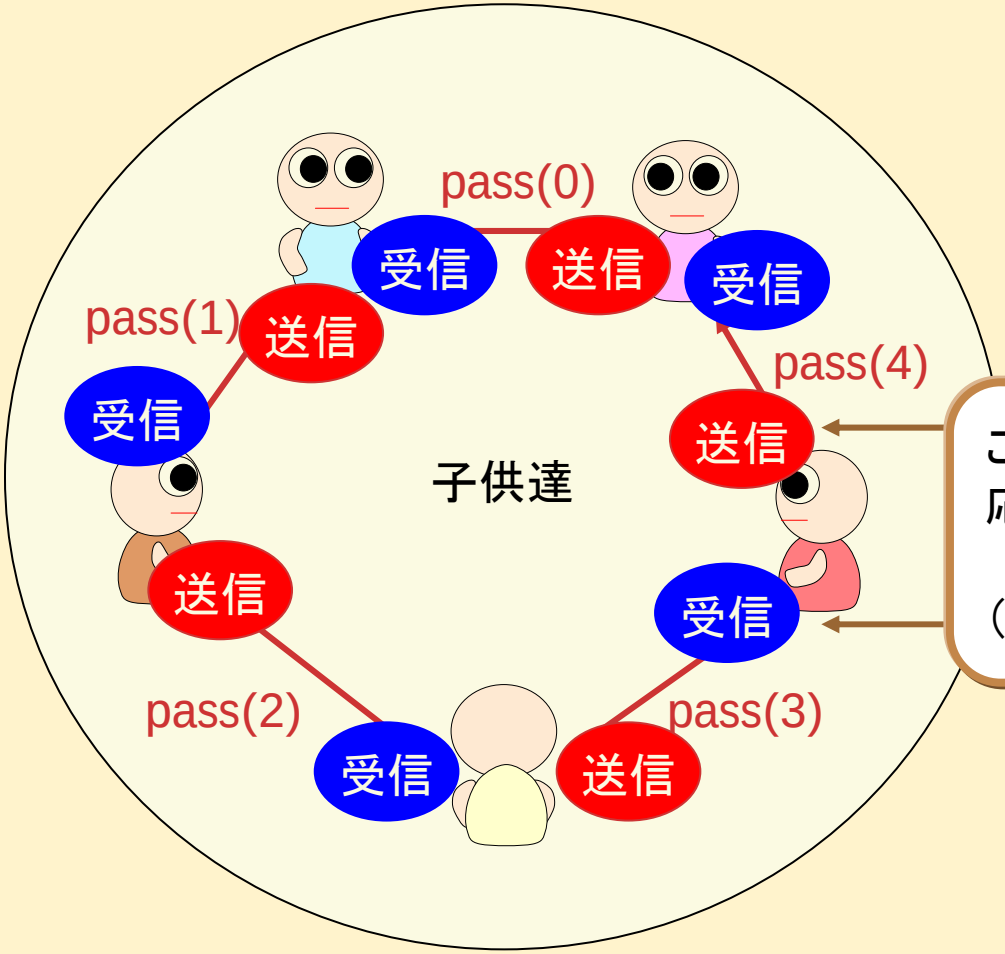


誰か最初に受信して～

passは送信と受信が同時に実行される同期型チャネルとする。

各子供の動作

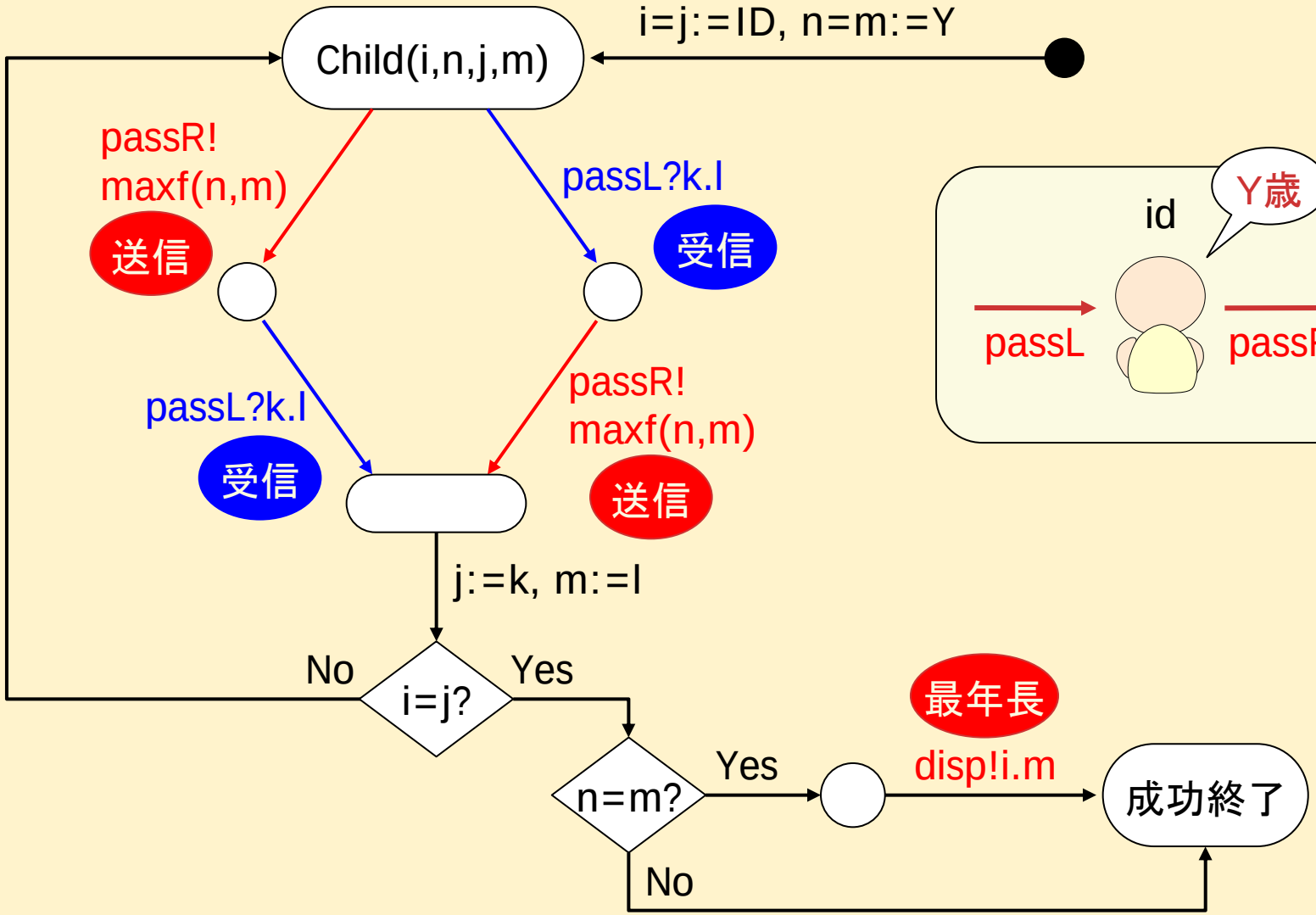
送信か受信を選択的に実行できるようにする。



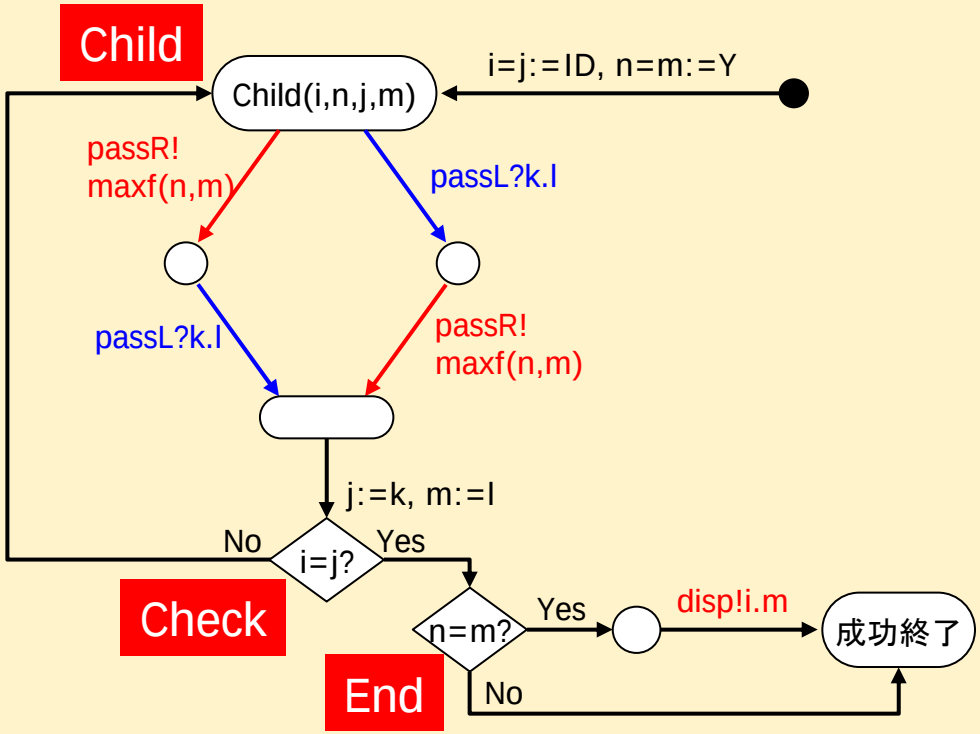
この送信と受信は外部の状況に応じて選択的に実行できる。

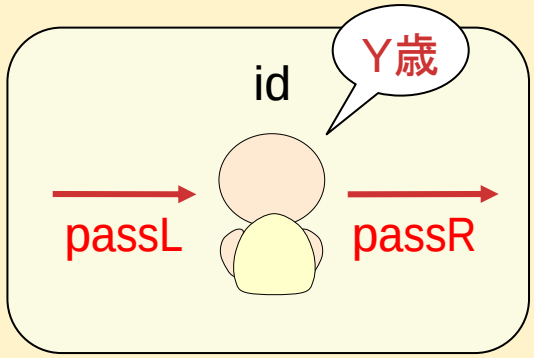
（JCSP的にいえばAlternativeにする）

各子供のモデル化


Child(i,n,j,m)
i=j:=ID, n=m:=Y
passR!
maxf(n,m)
送信
passL?k.l
受信
passL?k.l
受信
passR!
maxf(n,m)
送信
j:=k, m:=l
No
i=j?
Yes
n=m?
Yes
No
最年長
disp!i.m
成功終了
id
Y歳
passL
passR

各子供のFDRスクリプト



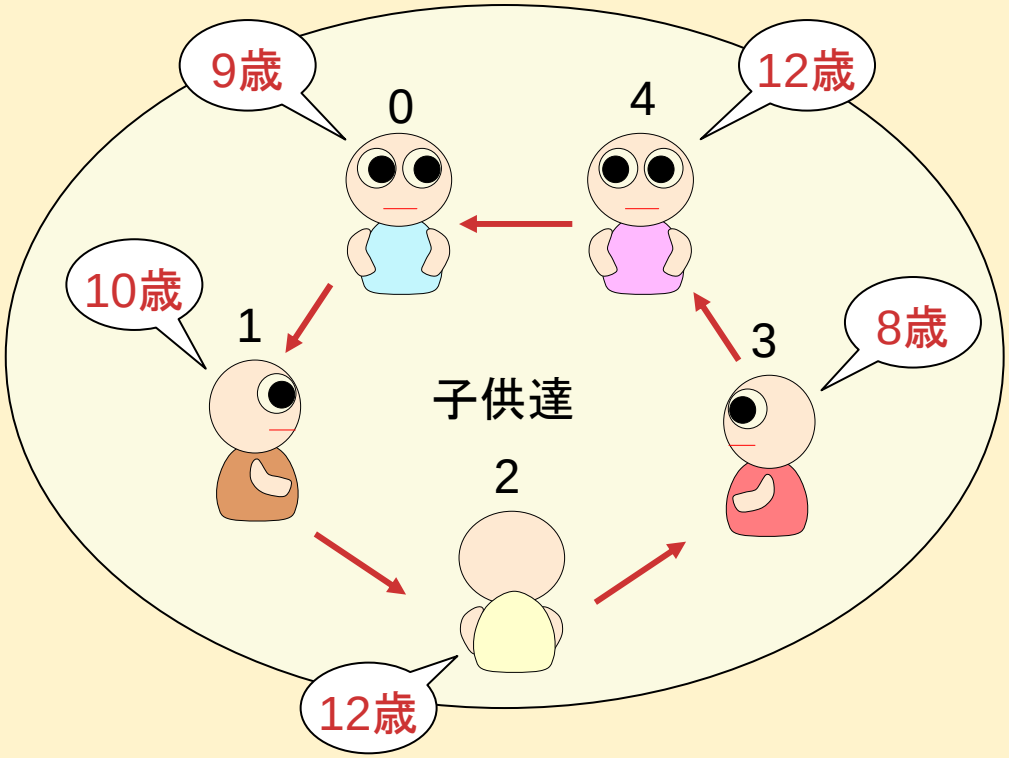


Child(i,n,j,m) = pass.right(i)!j.maxf(n,m) -> pass.i?k.l -> Check(i,n,k,l) []
pass.i?k.l -> pass.right(i)!j.maxf(n,m) -> Check(i,n,k,l)

Check(i,n,j,m) = if (i==j) then End(i,n,m) else Child(i,n,j,m)

End(i,n,m) = if (n==m) then disp.i.m -> SKIP else SKIP

子供達の並行合成




```
Sys = (|| i:Id @ [Ev(i)] Child(i,read(list,i),i,read(list,i))) ¥ {|pass|}
```

FDR(CSP)の並行合記号
（共有するチャネル名を接続）

passの隠蔽
（見えるのは結果dispのみ）

仕様のFDRスクリプト

仕様： 最年長の子供のIDを順不同で表示する。

IDがiの子供の年齢が最年長かを判定
（listは子供全員の年齢のリスト）

```
Spec = || i:Id@[Ev2(i)]   if (read(list,i) == maxl(list))
                          then disp.i.read(list,i) -> SKIP else SKIP
```

最年長の子供のIDを表示

並行合成なので複数いるときは順不同

検証

FDRスクリプトの全体

```
-- 準備

 N = 5        -- 整数の最大値
 M = 5        -- 人数

Nat = {0..N}      -- 整数の集合
 Id = {0..(M-1)}  -- 個人ID

right(i) = (i+1)%M    -- 右隣のID

list = <0, 1, 4, 2, 1>  -- 初期値

-- read(s,n) 列sの番めを読み出す

read(s,0) = head(s)
read(s,n) = read(tail(s),n-1)

-- maxf(n,m) nかmの大きい方を返す

maxf(n,m) = if (n > m) then n else m

-- maxl(s) 列s中の最大値を返す

maxl(<>) = 0
maxl(<n>) = n
maxl(<n>^s) = if n >= maxl(s) then n else maxl(s)

-- channel 宣言

-- pass.i.k.n     左隣からiへId付値k.nを送受信
-- disp.k.n       最大値n(kは所有者のID)を表示

channel pass : Id.Id.Nat
channel disp  : Id.Nat
```

```
-- プロセス

Child(i,n,j,m) = pass.right(i)!j.maxf(n,m) -> pass.i?k.l -> Check(i,n,k,l) [] -- 先に左に送信
                 pass.i?k.l -> pass.right(i)!j.maxf(n,m) -> Check(i,n,k,l)    -- 先に右から受信

Check(i,n,j,m) = if (i==j) then End(i,n,m) else Child(i,n,j,m)    -- 自分のIdが戻ったら終了

End(i,n,m) = if (n==m) then disp.i.m -> SKIP else SKIP

-- プロセスの合成

 Ev(i) = {| pass.i, pass.right(i), disp.i |}
  Sys = (|| i:Id @ [Ev(i)] Child(i,read(list,i),i,read(list,i))) ¥ {|pass|}

-- 仕様:最大値がdispから順不同に出力される

Ev2(i) = {| disp.i |}

  Spec = || i:Id@[Ev2(i)]
          if (read(list,i) == maxl(list))
          then disp.i.read(list,i) -> SKIP else SKIP
```

```
-- 検証

assert Spec [FD= Sys
assert Sys [FD= Spec
```

データ部（型、関数、チャネルなど）

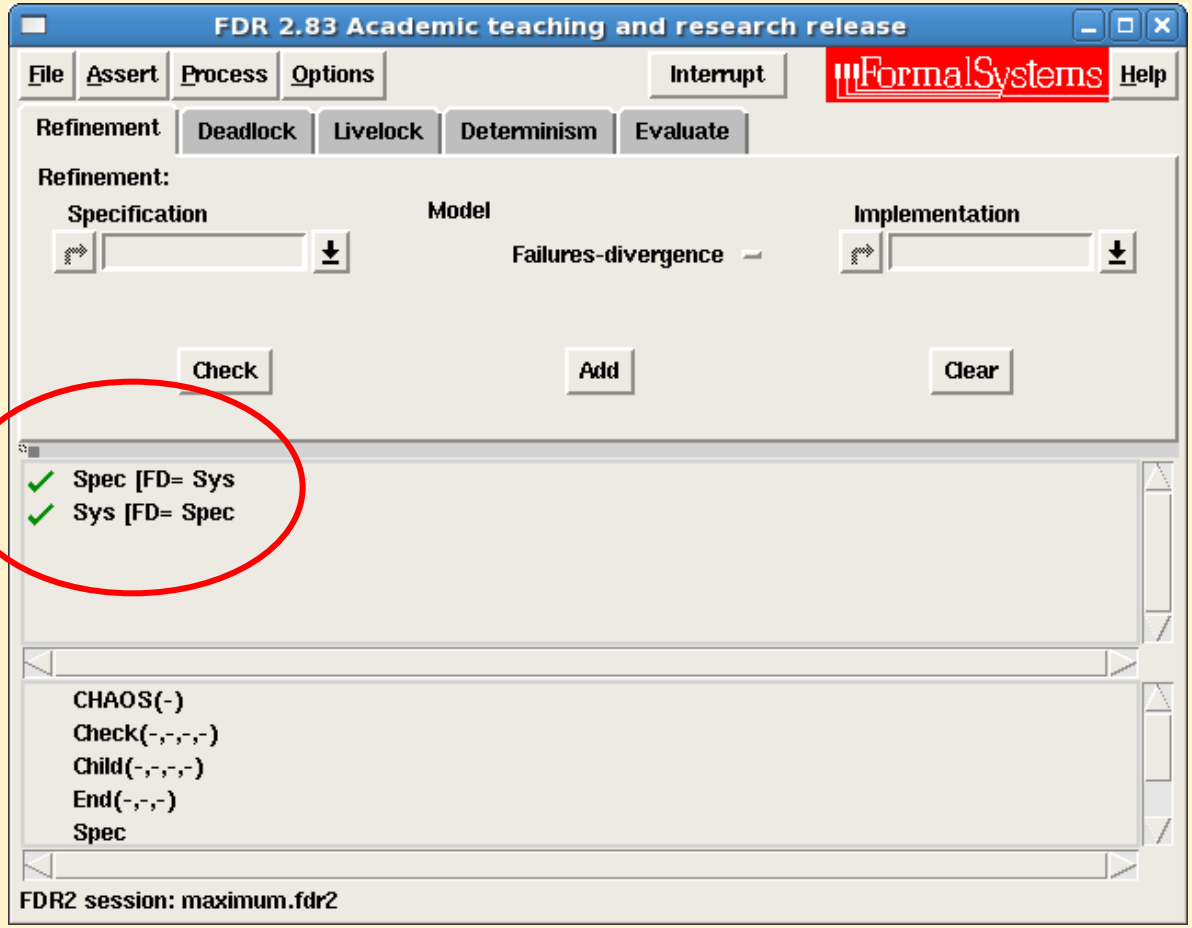
プロセス部（並行システム、仕様など）

検証部（詳細化関係、デッドロックなど）

FDRによる検証例

FDRスクリプト

入力



検証結果
並行動作と仕様は等価

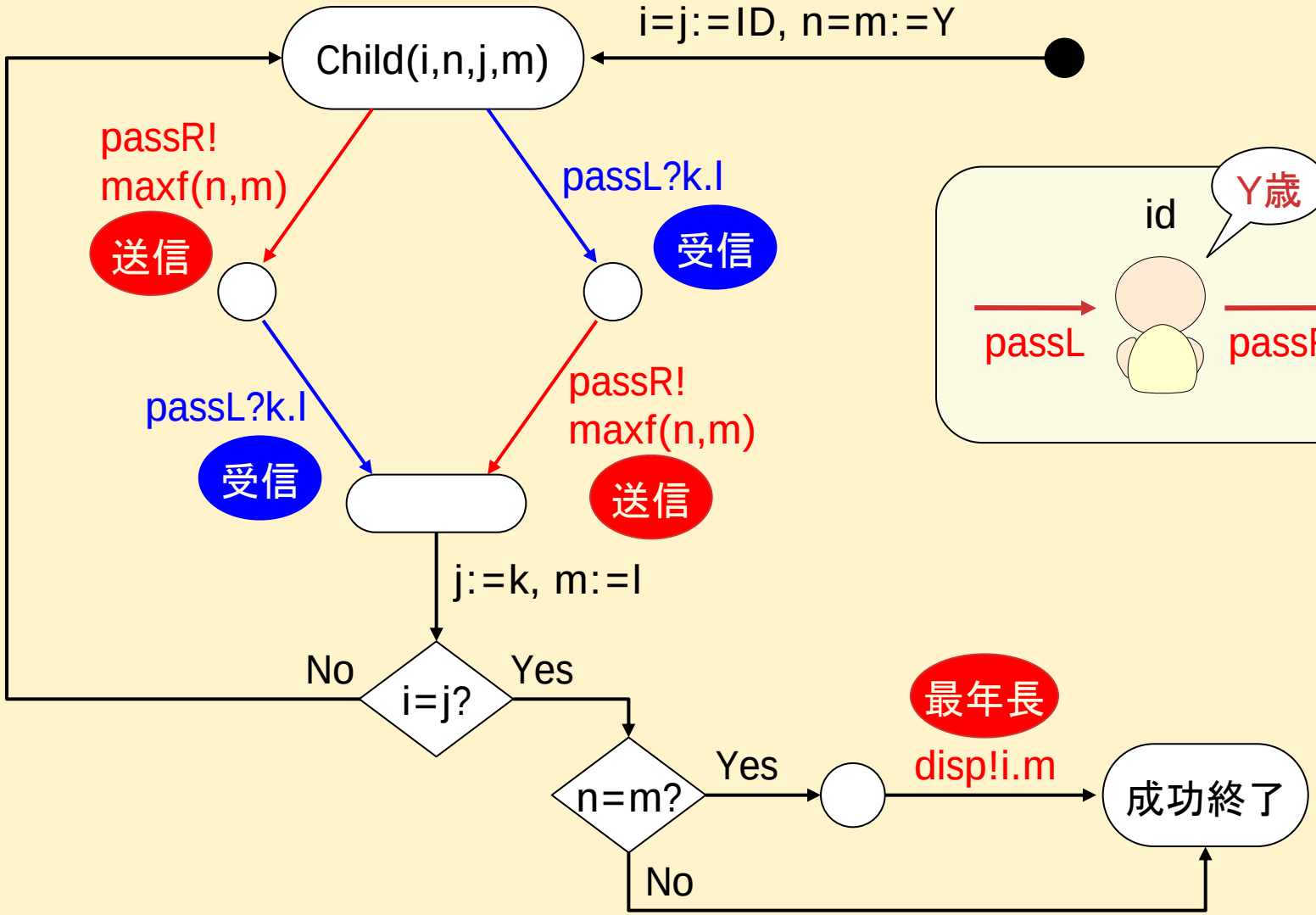
FDRで検証可能な詳細化関係の種類と特徴

- CSPにはいくつかの種類の詳細化関係がある（検証目的で使い分ける）。
- FDRではCSPの次の3種類の詳細化関係を検証できる。

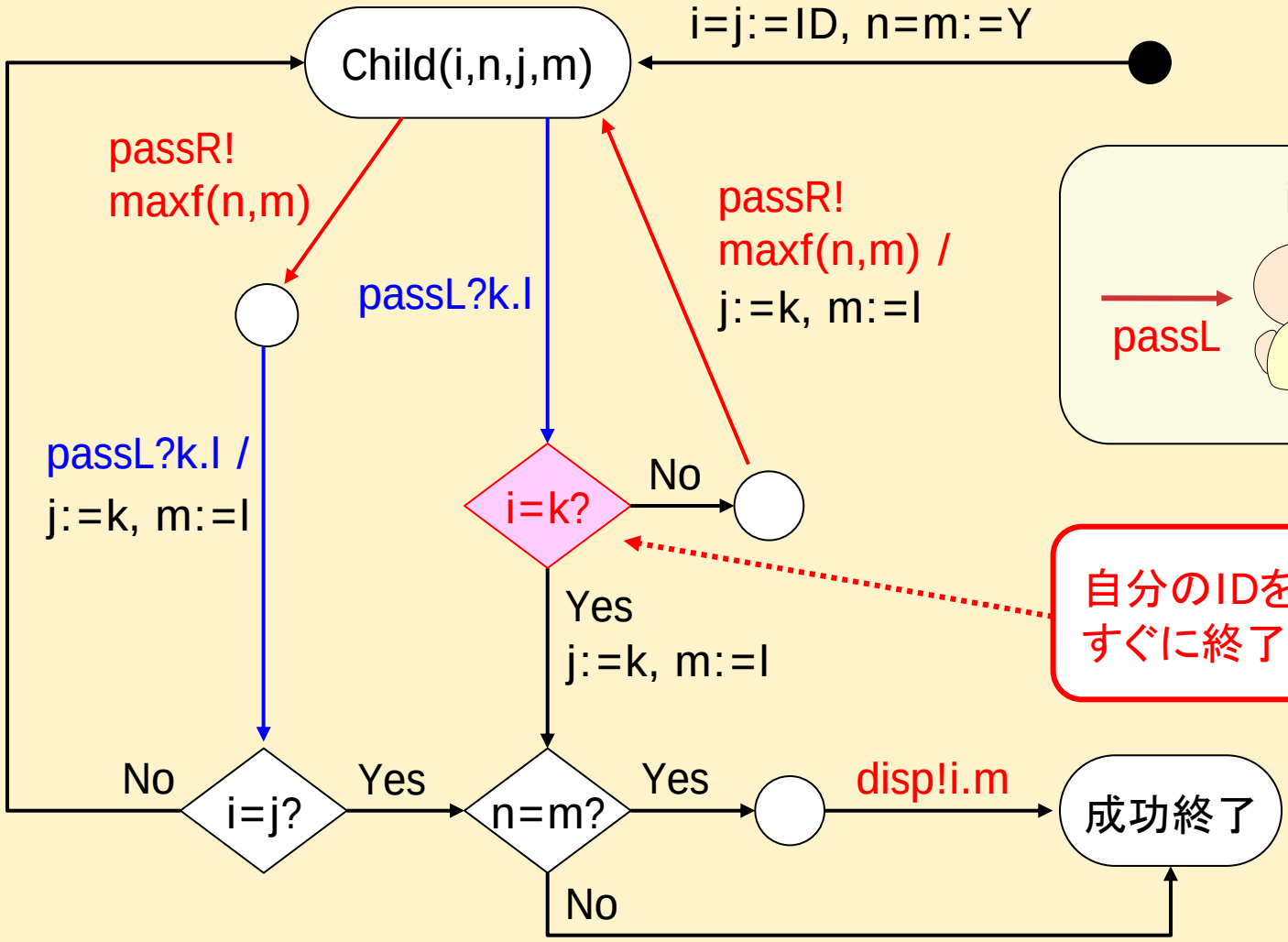
$P \sqsubseteq_T Q$	トレースに基づく最も簡単な詳細化関係： PからQへの詳細化の過程で危険なトレースが入り込まない。 （安全性検証）
$P \sqsubseteq_F Q$	トレースと拒否に基づく詳細化関係： PからQへの詳細化の過程で拒否されるイベントが増えない。 （デッドロックフリー性検証、活性検証）
$P \sqsubseteq_{FD} Q$	トレースと拒否と内部ループに基づく詳細化関係： PからQへの詳細化の過程で内部ループが増えない。 （ライブロックフリー性検証）

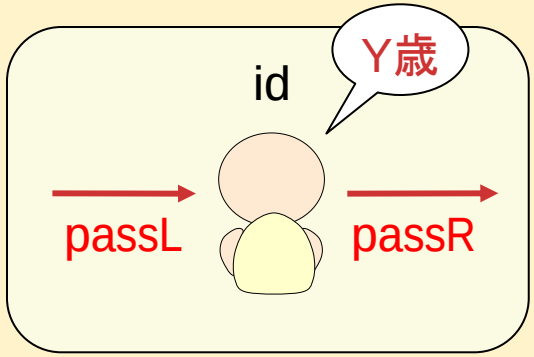
デバッグ

各子供のモデル化

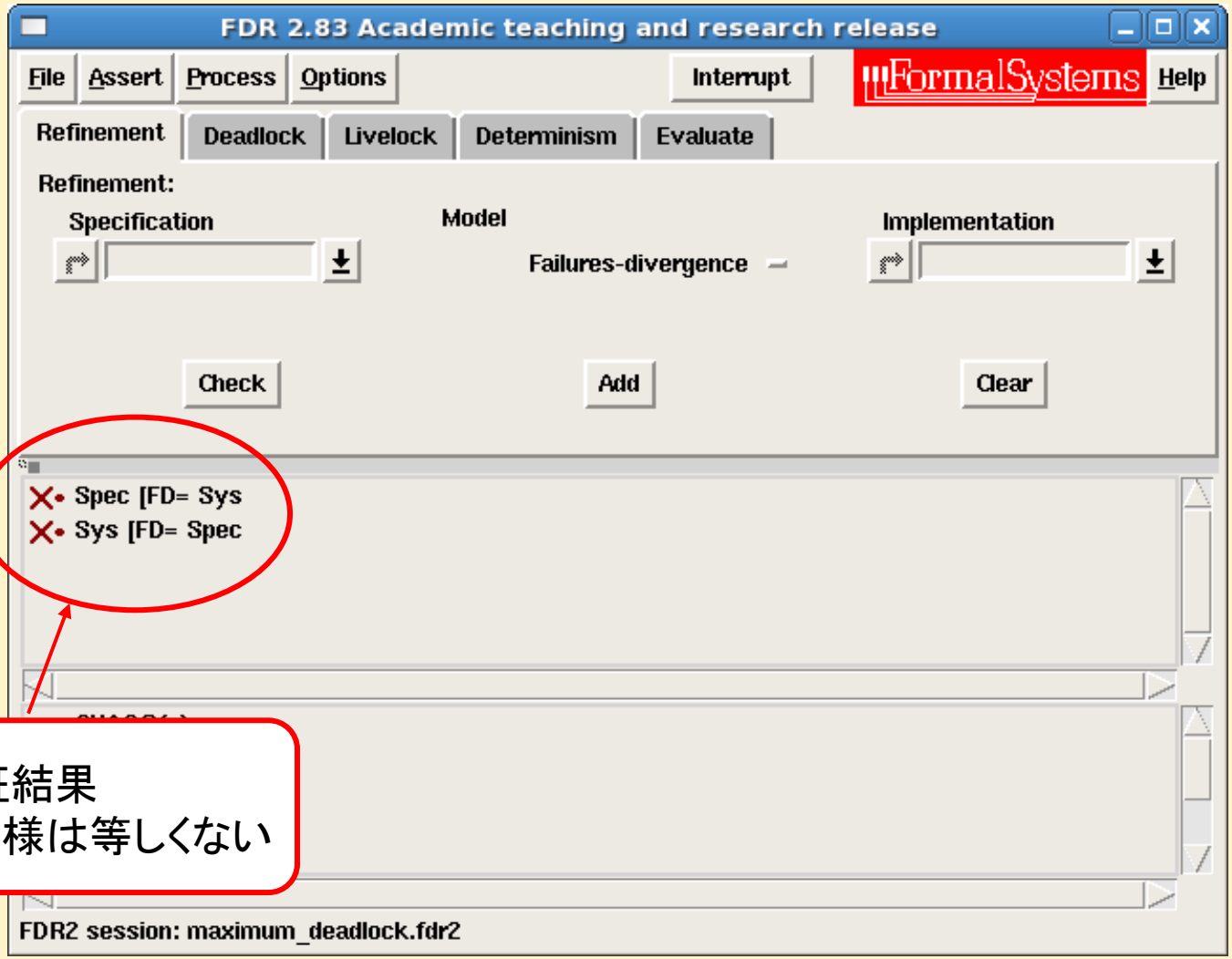

Child(i,n,j,m)
i=j:=ID, n=m:=Y
passR!
maxf(n,m)
送信
passL?k.l
受信
passL?k.l
受信
passR!
maxf(n,m)
送信
j:=k, m:=l
No
i=j?
Yes
n=m?
Yes
No
最年長
disp!i.m
成功終了
id
Y歳
passL
passR

子供のモデル化（デッドロック版）





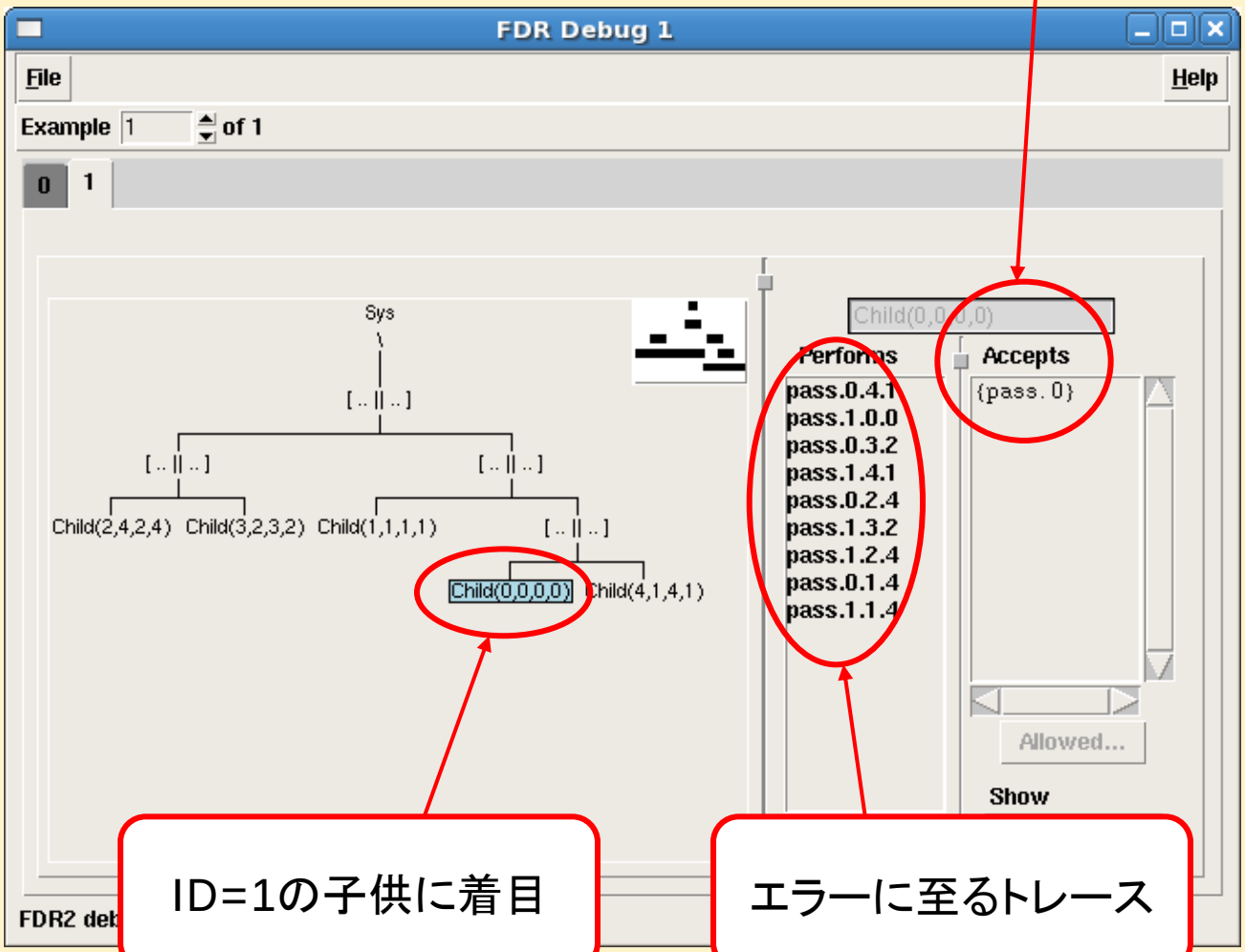
FDRによる検証（デッドロック版）



検証結果
並行動作と仕様は等しくない

デバッガの起動（デッドロック版）

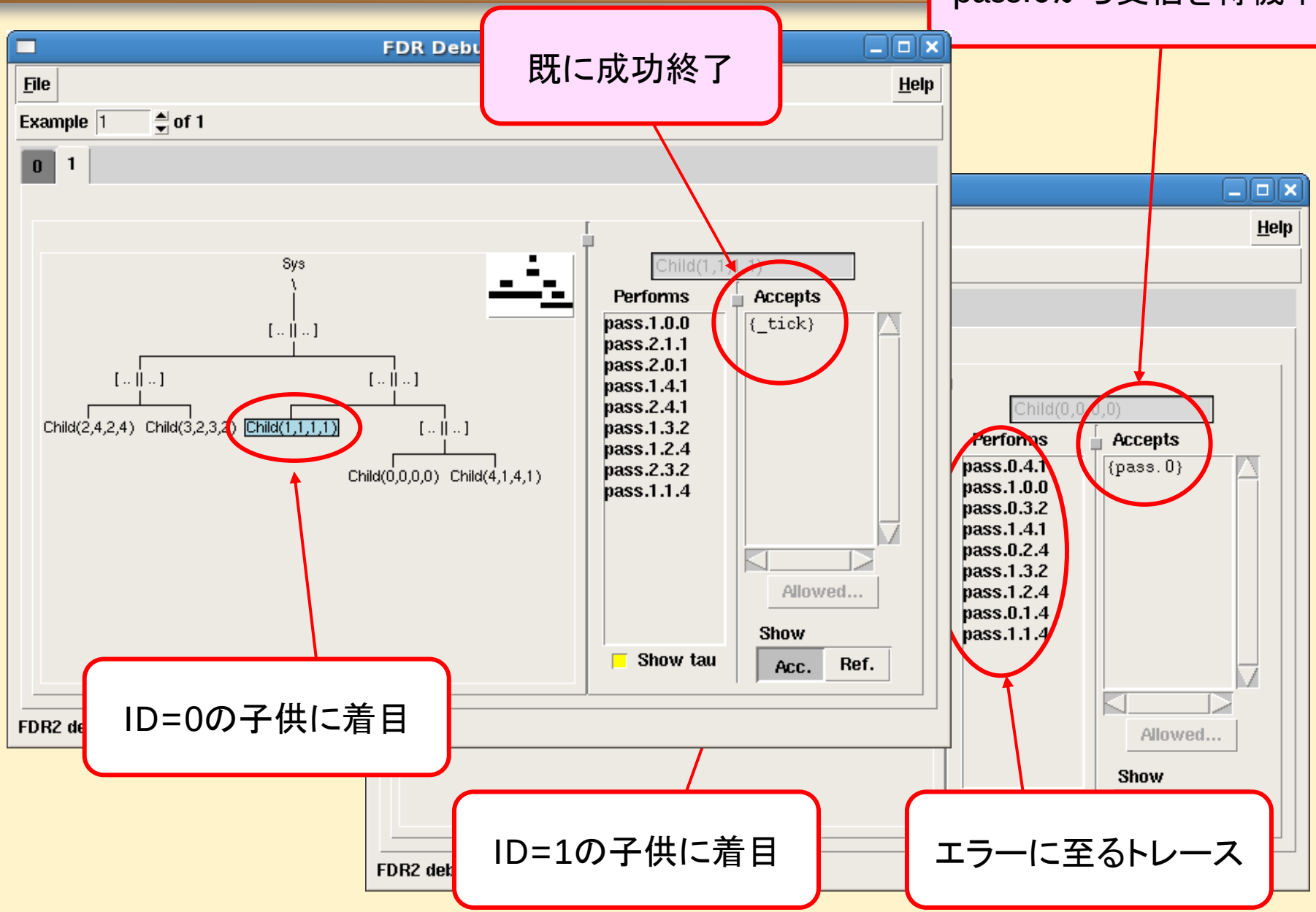
pass.0から受信を待機中



ID=1の子供に着目

エラーに至るトレース

デバッガの起動（デッドロック版）



実装

JCSPプログラムへの変換

```
-- 準備

 N = 5       -- 整数の最大値
 M = 5       -- 人数

Nat = {0..N}      -- 整数の集合
 Id = {0..(M-1)}  -- 個人ID

right(i) = (i+1)%M    -- 右隣のID

list = <0, 1, 4, 2, 1>  -- 初期値

-- read(s,n) 列sの番めを読み出す

read(s,0) = head(s)
read(s,n) = read(tail(s),n-1)

-- maxf(n,m) nかmの大きい方を返す

maxf(n,m) = if (n > m) then n else m

-- maxl(s) 列s中の最大値を返す

maxl(<>) = 0
maxl(<n>) = n
maxl(<n>^s) = if n >= maxl(s) then n else maxl(s)

-- channel 宣言

-- pass.i.k.n     左隣からiへId付値k.nを送受信
-- disp.k.n       最大値n(kは所有者のID)を表示

channel pass : Id.Id.Nat
channel disp  : Id.Nat
```

```
-- プロセス

Child(i,n,j,m) = pass.right(i)!j.maxf(n,m) -> pass.i?k.l -> Check(i,n,k,l) [] -- 先に左に送信
                 pass.i?k.l -> pass.right(i)!j.maxf(n,m) -> Check(i,n,k,l)    -- 先に右から受信

Check(i,n,j,m) = if (i==j) then End(i,n,m) else Child(i,n,j,m)    -- 自分のIdが戻ったら終了

End(i,n,m) = if (n==m) then disp.i.m -> SKIP else SKIP

-- プロセスの合成

 Ev(i) = {| pass.i, pass.right(i), disp.i |}
  Sys = (|| i:Id @ [Ev(i)] Child(i,read(list,i),i,read(list,i))) ¥ {|pass|}

-- 仕様:最大値がdispから順不同に出力される

Ev2(i) = {| disp.i |}

  Spec = || i:Id@[Ev2(i)]
          if (read(list,i) == maxl(list))
          then disp.i.read(list,i) -> SKIP else SKIP
```

```
-- 検証

assert Spec [FD= Sys
assert Sys [FD= Spec
```

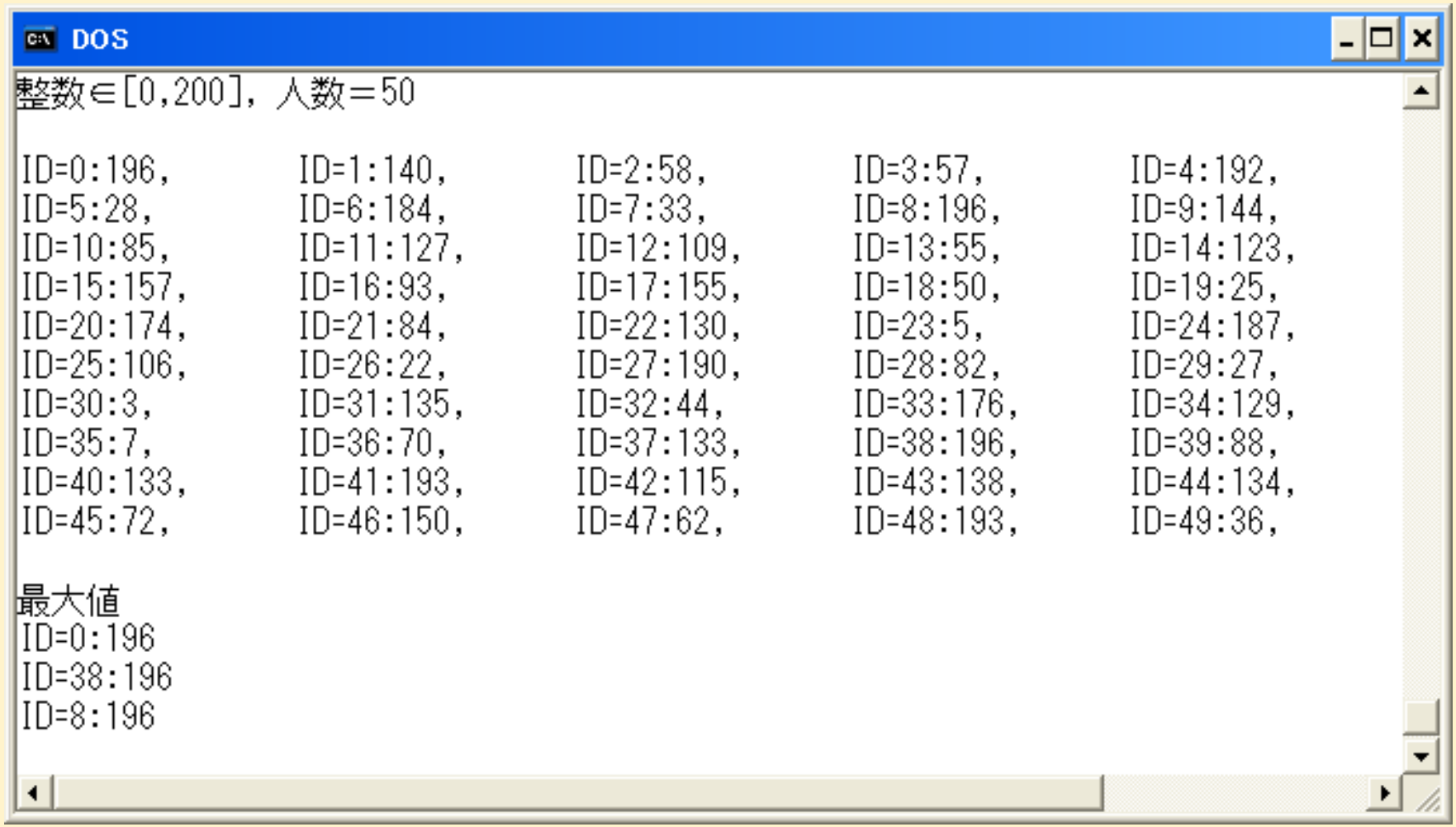
変換

Javaプログラム
（JCSPライブラリ使用）

この例でプログラミングは2時間程度

最初の実行でチャネル生成を忘れて
エラーが1つでたが、それを修正後は
他のエラーはなし。

JCSPプログラムの実行例(リーダー決定問題)



まとめ

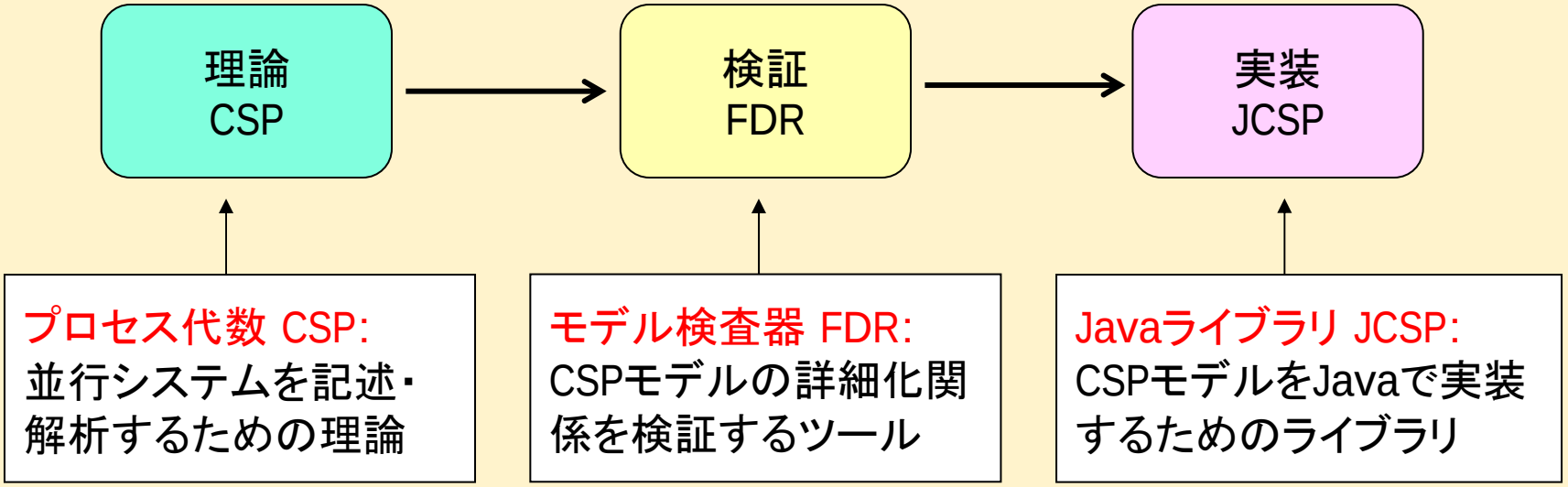
まとめ

(1) FDRでは並行システムも仕様もCSPモデルで表現する。

(2) CSPモデル間の詳細化関係をFDRで検証できる。
（デッドロック、ライブロック、決定性検証を含む）

(3) デバッガを使ってエラーに至るトレースとそのエラー状態を確認できる。

(4) FDRで検証してからJCSP等で実装すると致命的エラーを減らせる。



付録 （FDRスクリプト）

P.1

```
-- 準備

 N = 5        -- 整数の最大値
 M = 5        -- 人数

Nat = {0..N}      -- 整数の集合
 Id = {0..(M-1)}  -- 個人ID
right(i) = (i+1)%M    -- 右隣のID
list = <0, 1, 4, 2, 1>  -- 初期値

-- read(s,n) 列sの番めを読み出す

read(s,0) = head(s)
read(s,n) = read(tail(s),n-1)

-- maxf(n,m) nかmの大きい方を返す

maxf(n,m) = if (n > m) then n else m

-- maxl(s) 列s中の最大値を返す

maxl(<>) = 0
maxl(<n>) = n
maxl(<n>^s) = if n >= maxl(s) then n else maxl(s)

-- channel 宣言
-- pass.i.k.n     左隣からiへId付値k.nを送受信
-- disp.k.n       最大値n(kは所有者のID)を表示

channel pass : Id.Id.Nat
channel disp  : Id.Nat
```

P.2

```
-- プロセス

Child(i,n,j,m) =
  pass.right(i)!j.maxf(n,m) -> pass.i?k.l -> Check(i,n,k,l) []
  pass.i?k.l -> pass.right(i)!j.maxf(n,m) -> Check(i,n,k,l)

Check(i,n,j,m) = if (i==j) then End(i,n,m) else Child(i,n,j,m)

End(i,n,m) = if (n==m) then disp.i.m -> SKIP else SKIP

-- プロセスの合成

 Ev(i) = {| pass.i, pass.right(i), disp.i |}
  Sys = (|| i:Id @ [Ev(i)] Child(i,read(list,i),i,read(list,i)))
          ¥ {|pass|}

-- 仕様：最大値がdispから順不同に出力される

Ev2(i) = {| disp.i |}

  Spec = || i:Id@[Ev2(i)]
           if (read(list,i) == maxl(list))
           then disp.i.read(list,i) -> SKIP else SKIP

-- 検証

assert Spec [FD= Sys
assert Sys [FD= Spec
```