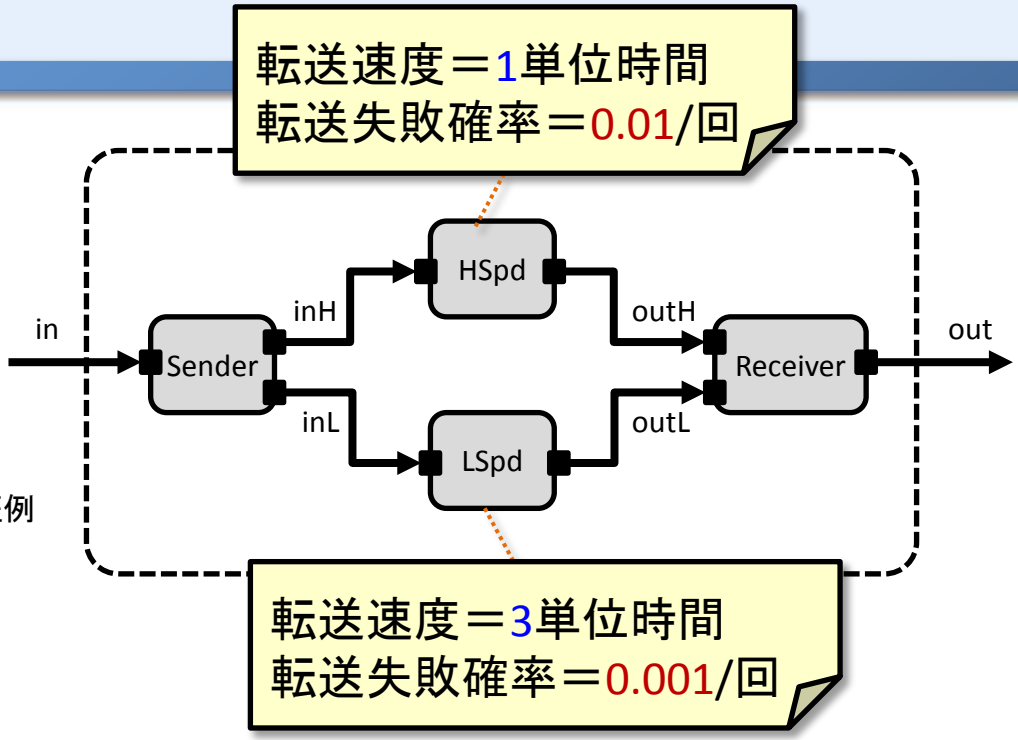
確率時間プロセス代数に基づくモデル検査器の紹介

産業技術総合研究所
情報技術研究部門
磯部祥尚

第8回CSP研究会（2011年11月26日）

発表内容

- **確率CSP（PAT）**
  - 最小確率と最大確率（Test1）
  - 確率の積（Test2）
  - 確率の和（Test3）
- **確率時間CSP（PAT）**
  - 制限時間と確率（Test4）
  - 確率と時間を考慮した並行システムの検証例
- **関連ツール**
  - 確率時間モデル検査器 PRISM



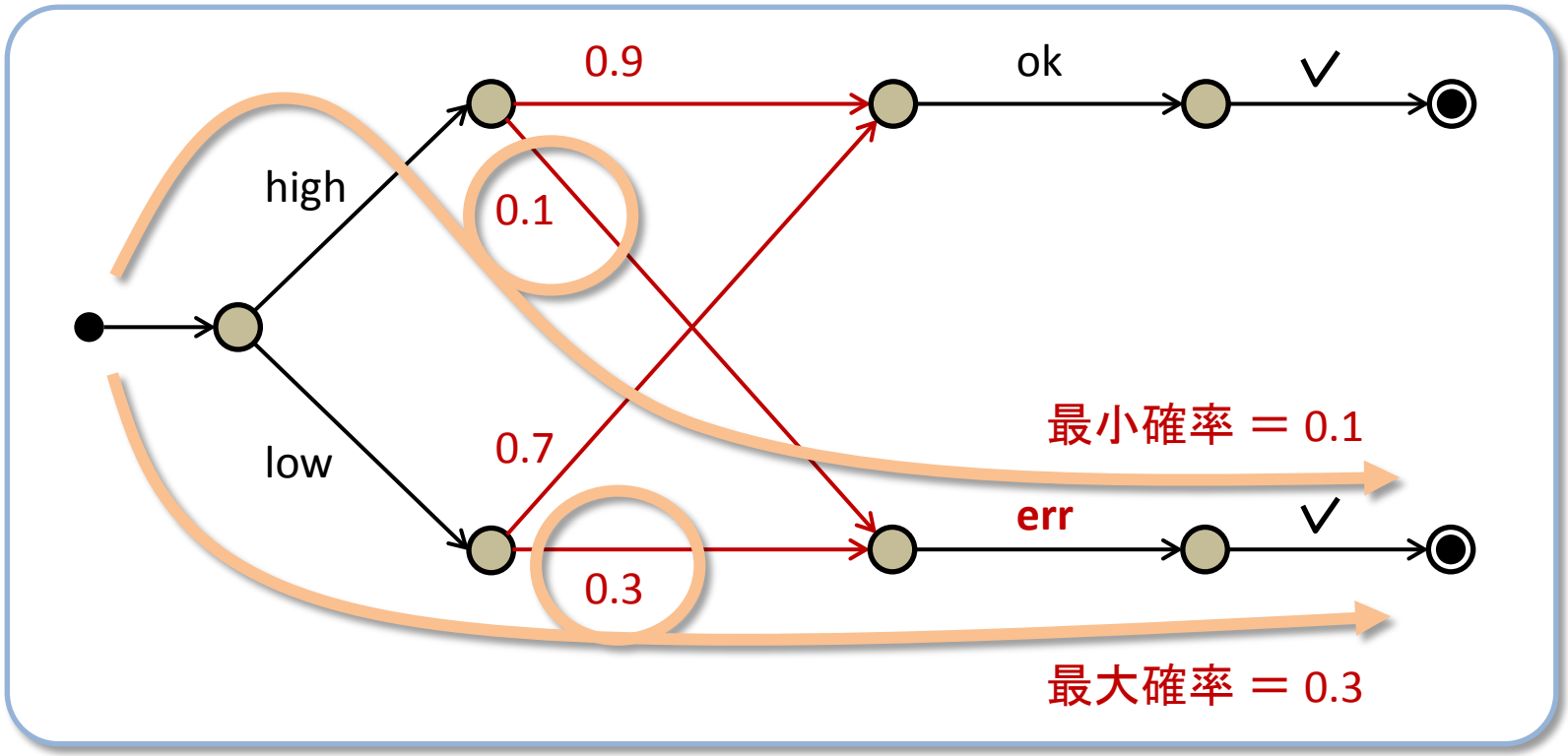
- PAT（Process Analysis Toolkit）：
  シンガポール大学が開発しているモデル検査器
  http://www.comp.nus.edu.sg/~pat/
- PRISM（Probabilistic model checker）：
  現在はオクスフォード大学が中心になって開発しているモデル検査器
  http://www.prismmodelchecker.org

確率CSP（PAT）

- 最小確率と最大確率（Test1）
- 確率の積（Test2）
- 確率の和（Test3）

最小確率と最大確率(例: Test1)

- 確率状態遷移システム:その遷移が実行される確率を指定することができる



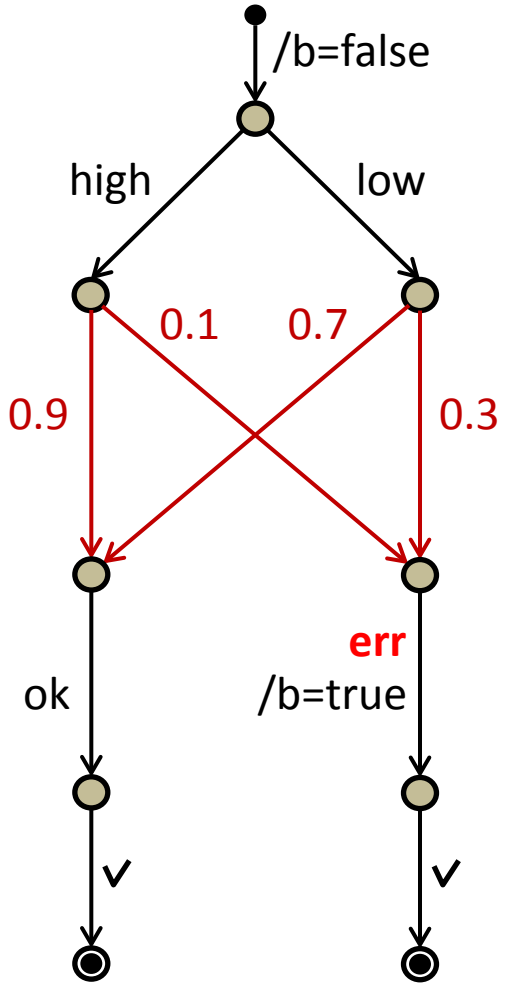
- いつか err になる確率は?
  - 最小確率 = 0.1
  - 最大確率 = 0.3

確率CSPによる記述（例：Test1）

■ 確率CSP：分岐確率を指定できるプロセス代数CSP

```
var b=false;

Test1() = high -> High() [] low -> Low();

 High() = pcase{[0.9] : OK() [0.1] : ERR()};
  Low() = pcase{[0.7] : OK() [0.3] : ERR()};

   OK() = ok -> Skip;
  ERR() = err{b=true} -> Skip;

#define Err (b==true);
#assert Test1() |= <>Err with prob;
```

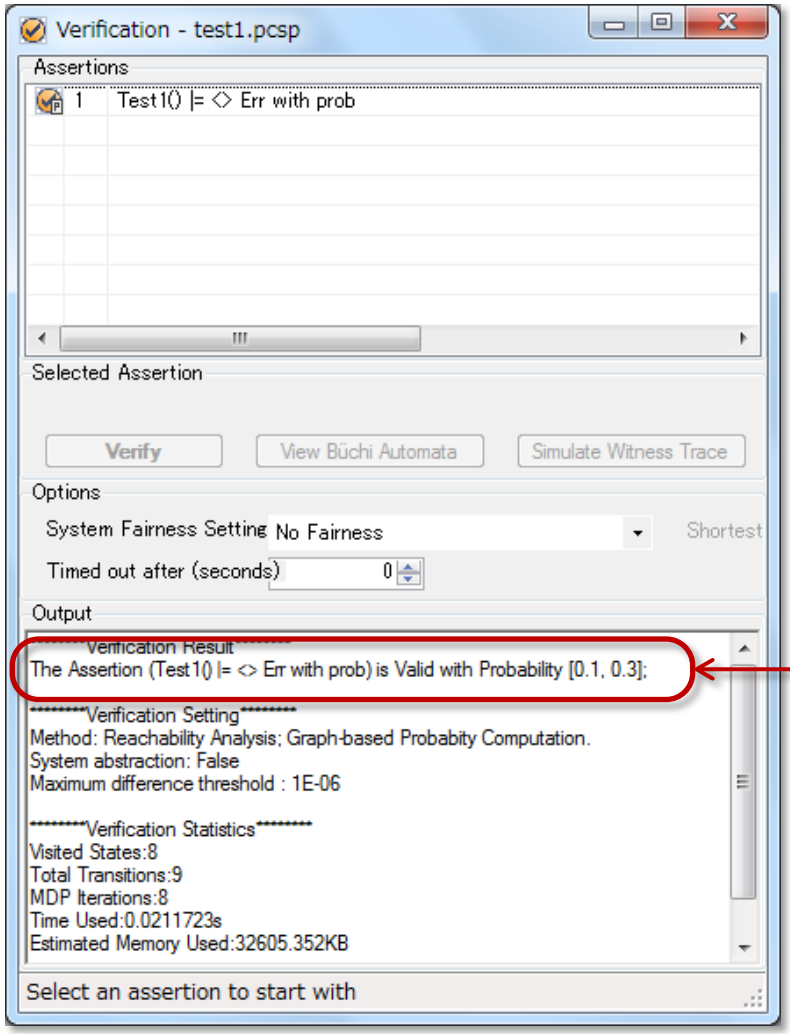
（モデル検査器PATの構文に従う）



いつかerrになる最小確率と最大確率を調べるための時相論理式

モデル検査器PATによる検証結果（例：Test1）

- PAT：確率CSPを検証できるモデル検査器



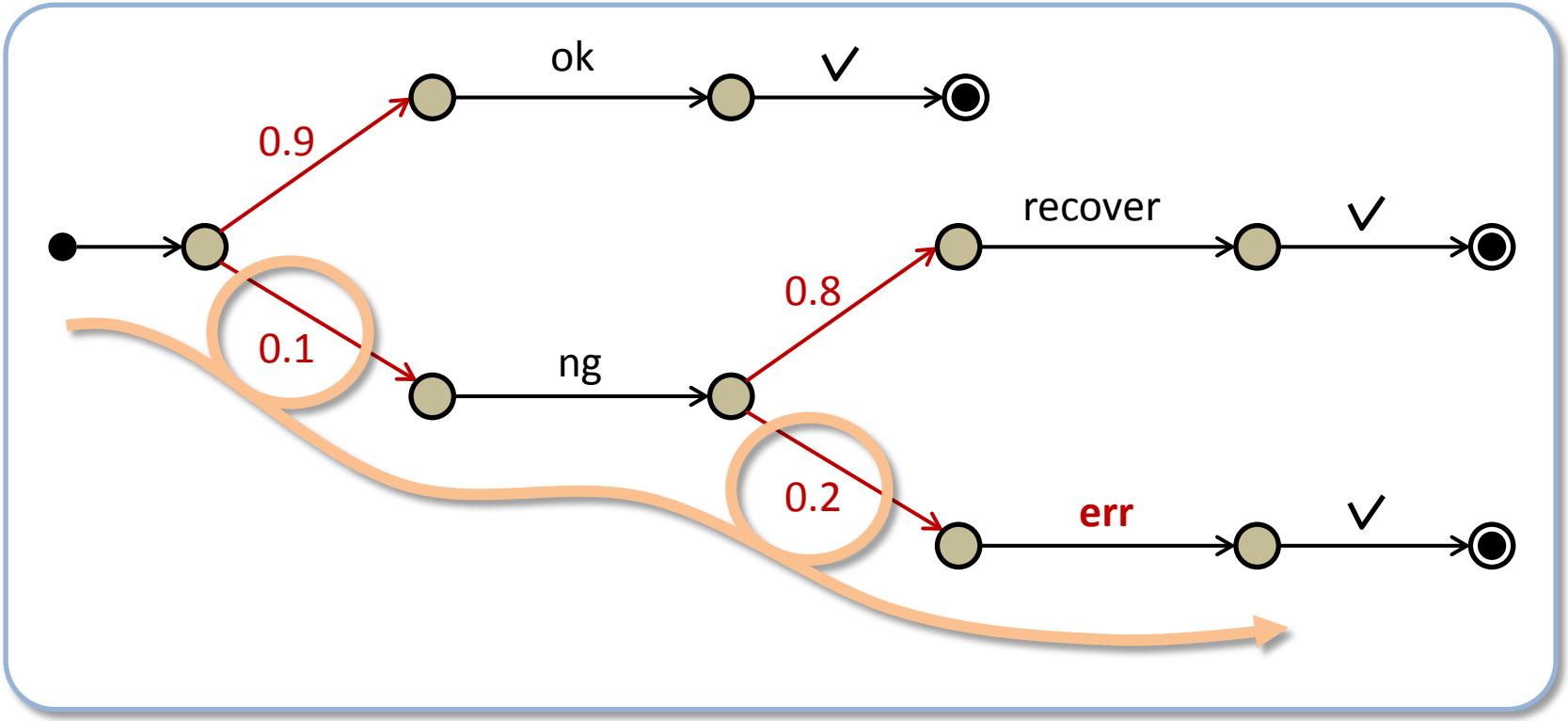

```
var b=false;

Test1() = high -> High() [] low -> Low();

 High() = pcase{[0.9] : OK() [0.1] : ERR()};
  Low() = pcase{[0.7] : OK() [0.3] : ERR()};

   OK() = ok -> Skip;
  ERR() = err{b=true} -> Skip;

#define Err (b==true);
#assert Test1() |= <>Err with prob;
```

The Assertion (Test1() |= <> Err with prob)
is Valid with Probability [0.1, 0.3];

[最小確率, 最大確率]

確率の積（例：Test2）

- Prob(a) × Prob(b): イベントaが起こり、その後イベントbが起きる確率

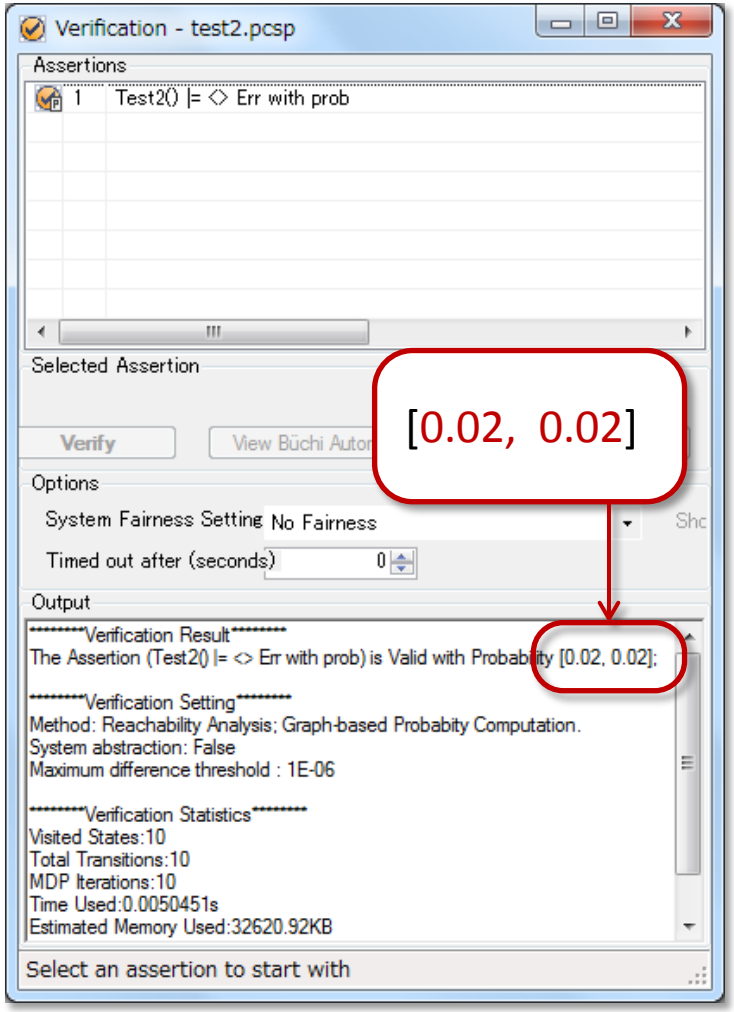


- いつか err になる確率は？
  - 最小確率　＝　最大確率　＝　0.1 × 0.2 ＝ 0.02

モデル検査器PATによる検証結果（例：Test2）

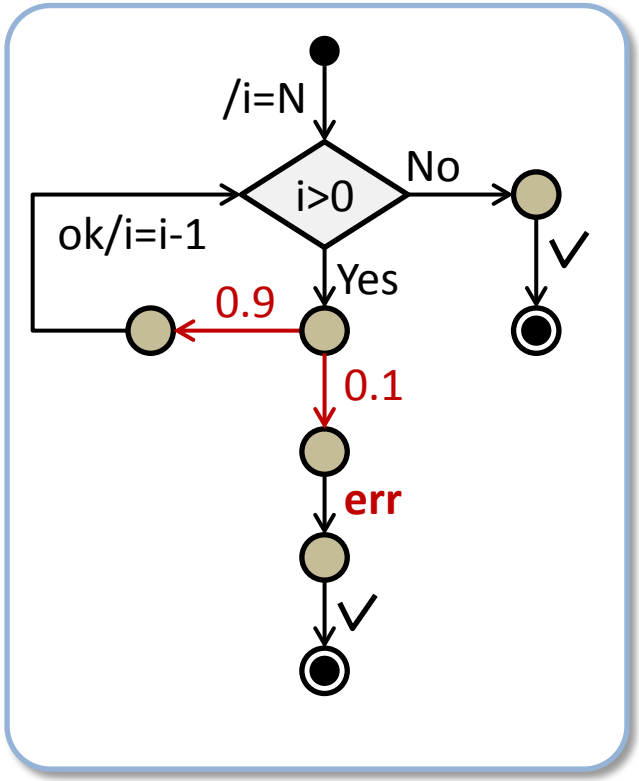
■ Test2の確率CSP記述（PAT）と検証結果

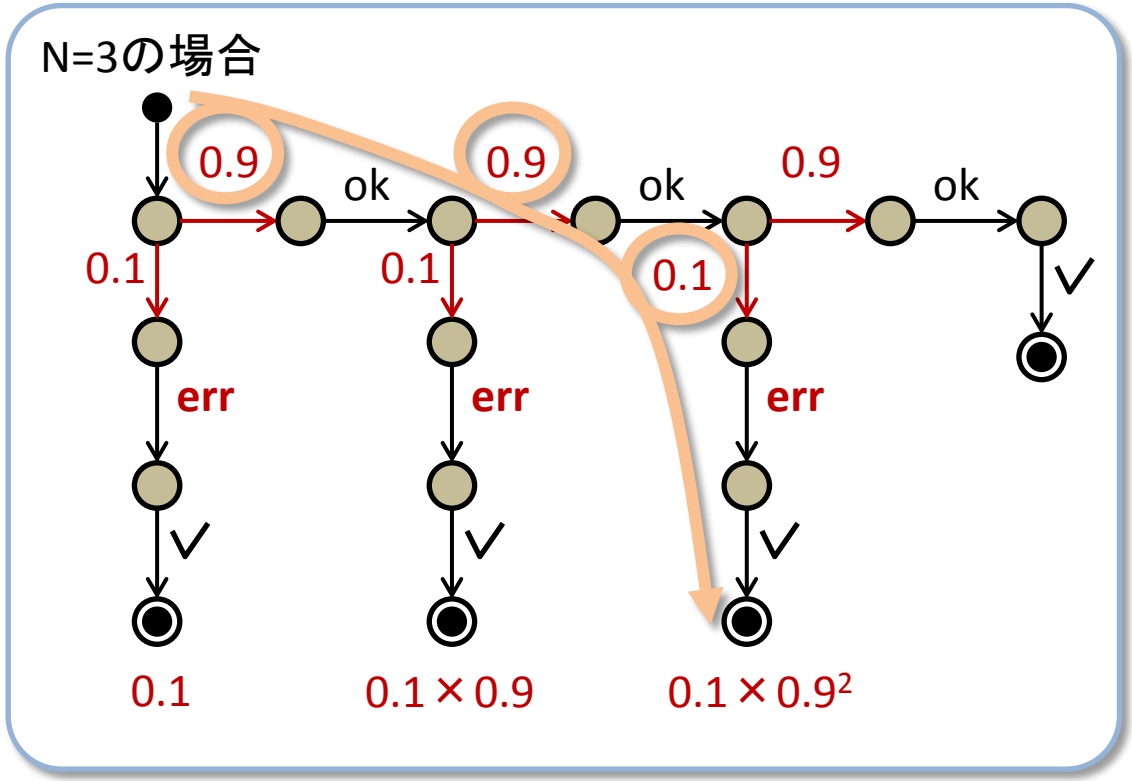
```
var b=false;

Test2() = pcase{
      [0.9] : ok -> Skip
      [0.1] : ng -> NG()
      };

NG() = pcase{
      [0.8] : recover -> Skip
      [0.2] : err{b=true} -> Skip
      };

#define Err (b==true);
#assert Test2() |= <>Err with prob;
```



確率の和（例：Test3）

■ Prob(a)+Prob(b): イベントaかbが起きる確率





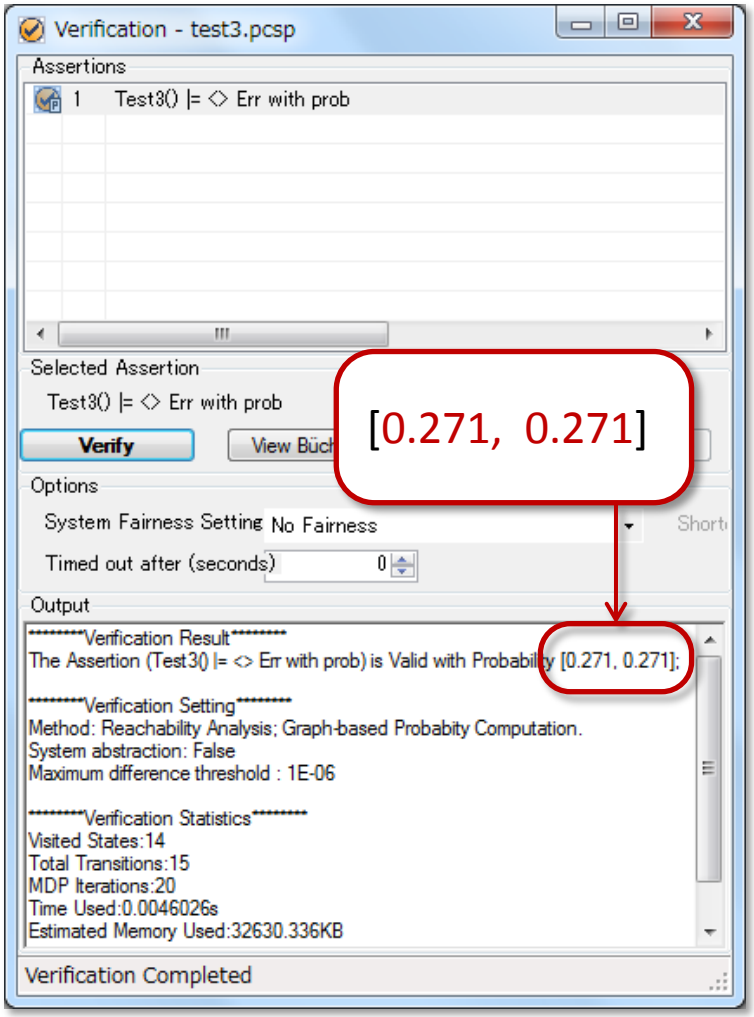
■ いつか err になる確率は？

- （N=3の場合）最小確率 ＝ 最大確率 ＝ $0.1 + 0.1 \times 0.9 + 0.1 \times 0.9^2 = 0.271$
- 最小確率 ＝ 最大確率 ＝ $0.1 \times \frac{1-0.9^N}{1-0.9} = 1-0.9^N$

モデル検査器PATによる検証結果（例：Test3）

■ Test3の確率CSP記述（PAT）と検証結果

```
var b=false;

Test3() = Loop(3);

Loop(i) =
  if (i>0) {
    pcase{
      [0.9] : ok -> Loop(i-1)
      [0.1] : err{b=true} -> Skip
    }
  }
  else{
    Skip
  };

#define Err (b==true);
#assert Test3() |= <>Err with prob;
```

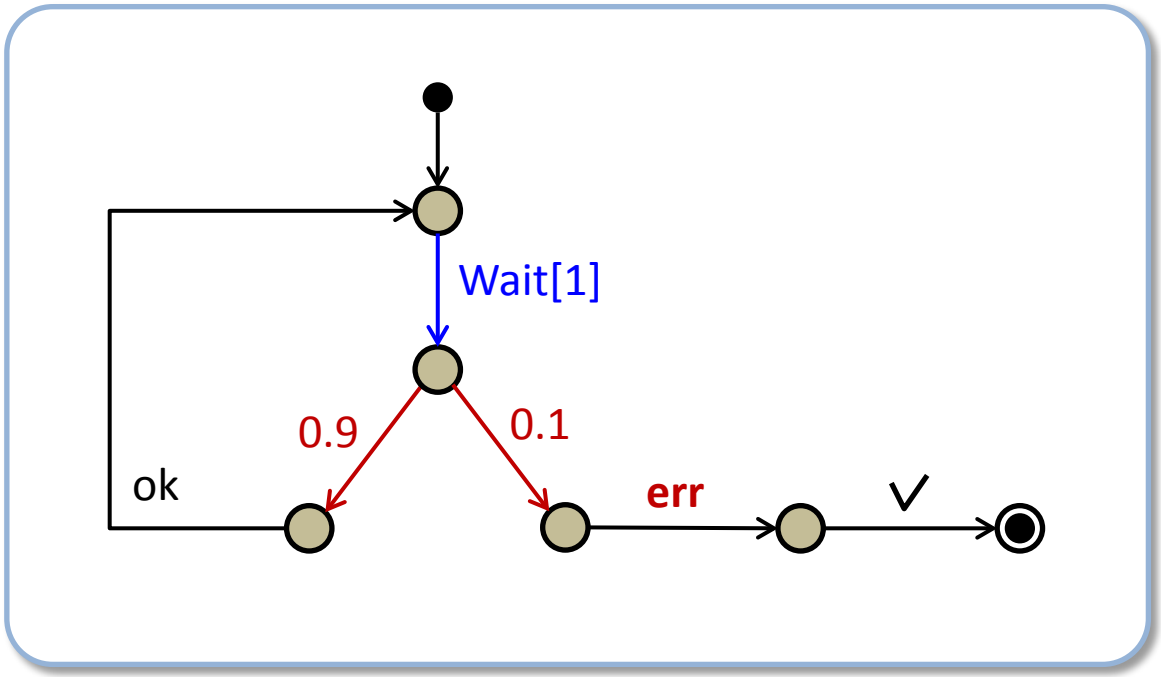


確率時間CSP（PAT）

- 制限時間と確率（Test4）
- 確率と時間を考慮した並行システムの検証例（Net）

確率時間CSP（例：Test4）

- Wait[t]: t 単位時間待ち

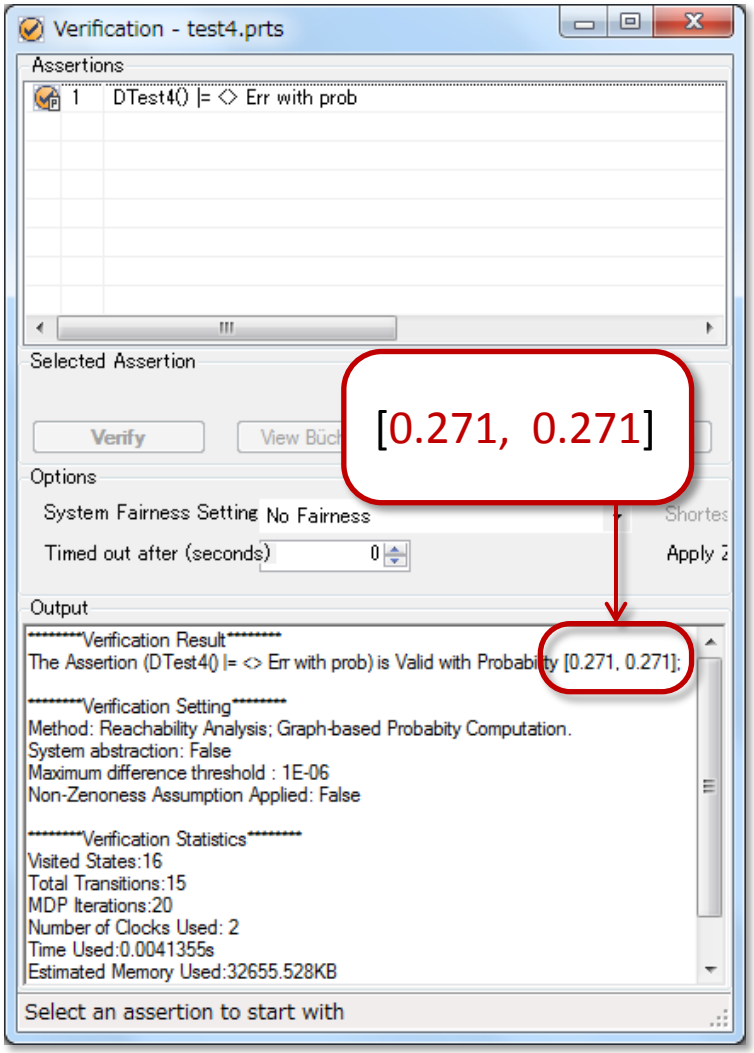


- T単位時間以内に err になる確率は？
  - （T=3の場合）最小確率 ＝ 最大確率 ＝ $0.1 + 0.1 \times 0.9 + 0.1 \times 0.9^2 = 0.271$
  - 最小確率 ＝ 最大確率 ＝ $0.1 \times \frac{1-0.9^T}{1-0.9} = 1-0.9^T$

モデル検査器PATによる検証結果（例：Test4）

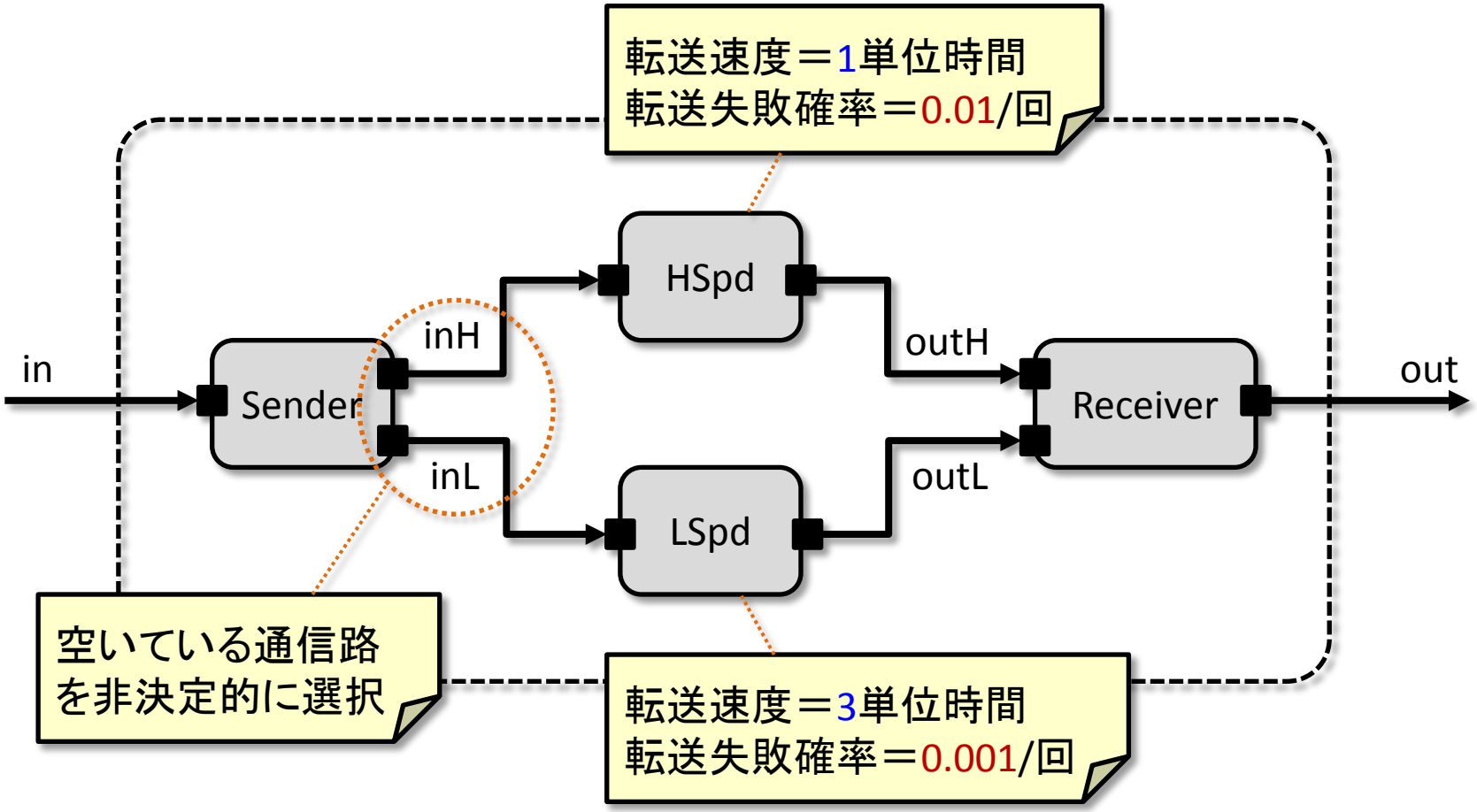
■ Test4の確率時間CSP記述（PAT）と検証結果

```
var b=false;

Test4() = Loop();

Loop() = Wait[1];
    pcase{
      [0.9] : ok -> Loop
      [0.1] : err{b=true} -> Skip
    };

DTest4() = Test4() deadline[3];

#define Err (b==true);
#assert DTest4() |= <>Err with prob;
```



時間と確率をもつ並行システムの例

- Net: 低信頼高速通信路HSpdと高信頼低速通信路LSpdをもつ転送システム



- 10単位以内に転送に失敗する最小確率と最大確率は？

確率時間CSPによる記述（例：Net）

■ Netの確率時間CSP記述（PAT）

```
var b=false;

Net() = (Sender() || HSpd() || LSpd() || Receiver())
        \{inH, inL, outH, outL};

  Sender() = in -> ToNet();
   ToNet() = inH -> Sender() [] inL -> Sender();

Receiver() = outH -> FromNet() [] outL -> FromNet();
 FromNet() = out -> Receiver();

HSpd() = inH -> Wait[1]; HFw();
 HFw() = pcase{ [0.99] : outH -> HSpd()
                [0.01] : err{b=true} -> Stop };

LSpd() = inL -> Wait[3]; LFw();
 LFw() = pcase{ [0.999] : outL -> LSpd()
                [0.001] : err{b=true} -> Stop };

DNet() = Net() deadline[10];
#define Err (b==true);
#assert DNet() |= <>Err with prob;
```

空いている通信路
を非決定的に選択

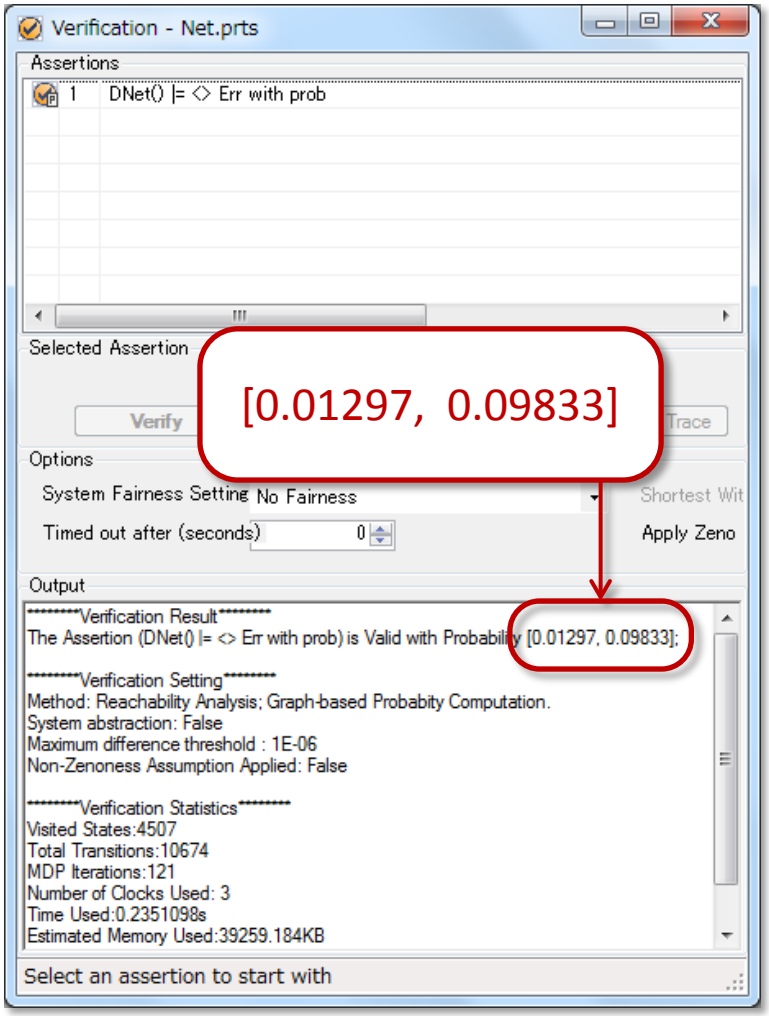
転送速度＝1単位時間
転送失敗確率＝0.01/回

転送速度＝3単位時間
転送失敗確率＝0.001/回

PATによる検証結果（例：Net）

- PATによるNetの検証結果

```
var b=false;

Net() = (Sender() || HSpd() || LSpd() || Receiver())
        \{inH, inL, outH, outL};

  Sender() = in -> ToNet();
   ToNet() = inH -> Sender() [] inL -> Sender();

Receiver() = outH -> FromNet() [] outL -> FromNet();
 FromNet() = out -> Receiver();

HSpd() = inH -> Wait[1]; HFw();
 HFw() = pcase{ [0.99] : outH -> HSpd()
                [0.01] : err{b=true} -> Stop };

LSpd() = inL -> Wait[3]; LFw();
 LFw() = pcase{ [0.999] : outL -> LSpd()
                [0.001] : err{b=true} -> Stop };

DNet() = Net() deadline[10];
#define Err (b==true);
#assert DNet() |= <>Err with prob;
```

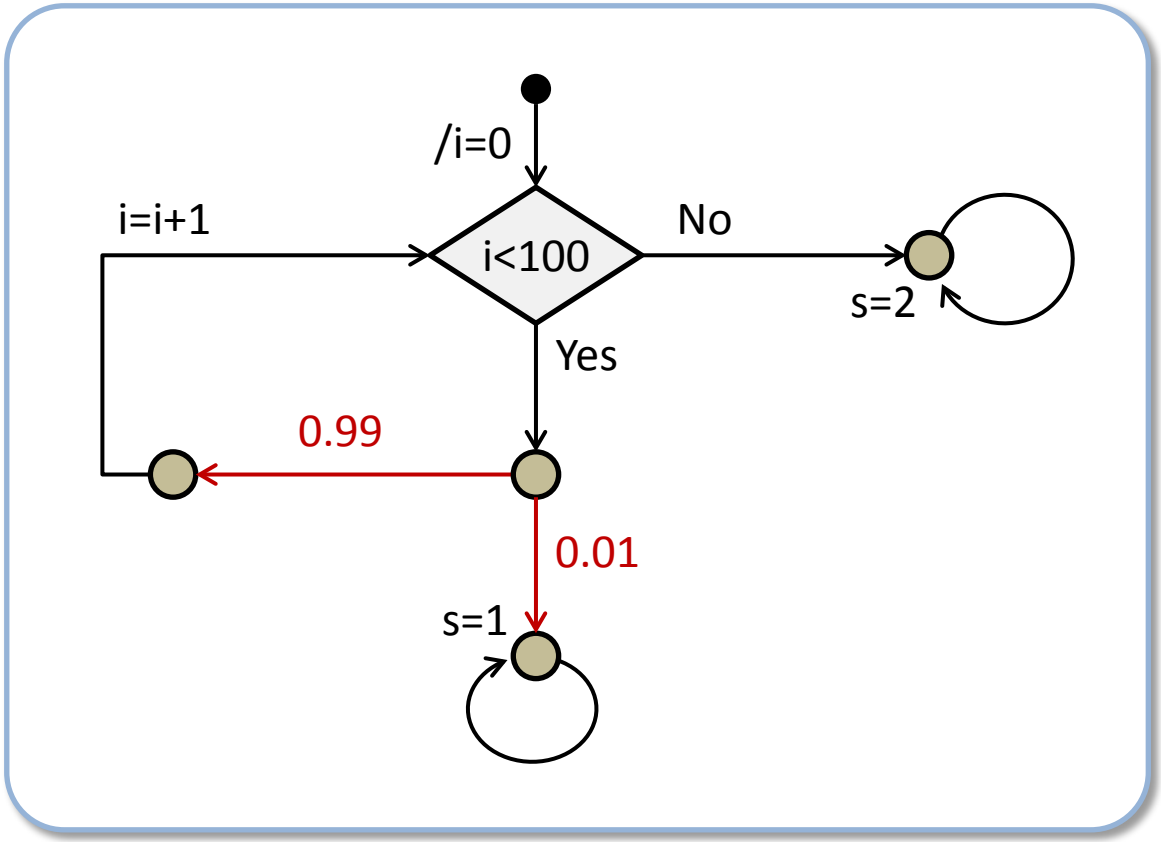


関連ツール

- モデル検査器 PRISM

モデル検査器PRISMによる検証例（Test001）

■ 1回あたり0.01の確率で失敗する動作（繰り返し回数の上限＝100回）



PRISMによる記述例

- PRISMによるTest001の記述例

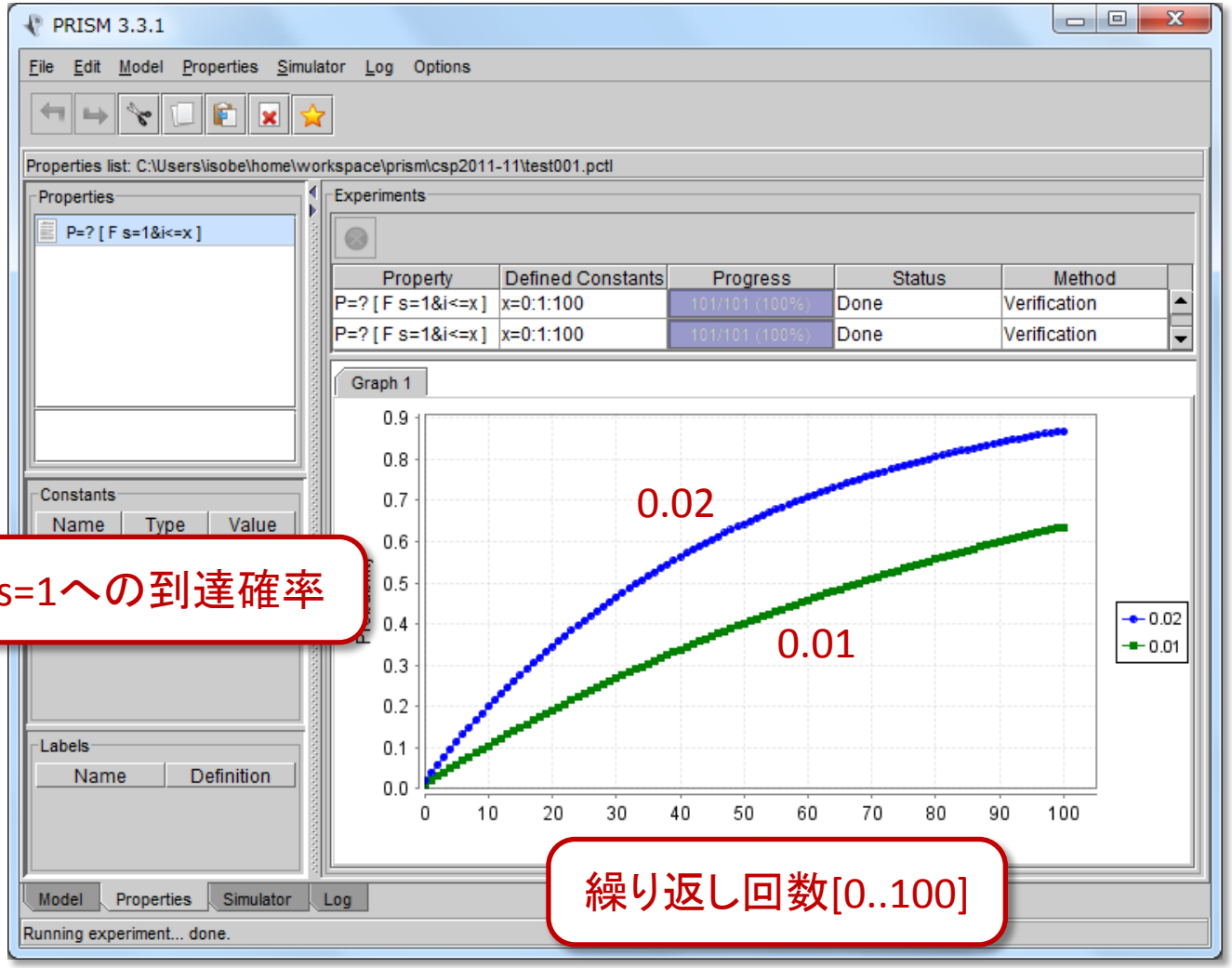
```
dtmc

module test001
  s : [0..2] init 0;
  i : [0..100] init 0;

  [] s=0 & i<100 -> 0.99 : (s'=0) & (i'= i+1) + 0.01 : (s'=1);
  [] s=0 & i=100 -> (s'=2);
  [] s=1 -> (s'=1);
  [] s=2 -> (s'=2);
endmodule
```

- 検証式「繰り返し回数 i が x 回以下で状態 s=1 に到達する確率は？」の記述例

```
const int x;

P=? [ F s=1 & i <= x ]
```

PRISMによる検証例

- PRISMによるTest001の検証例（1回あたりの失敗確率 0.01 と 0.02 の場合）



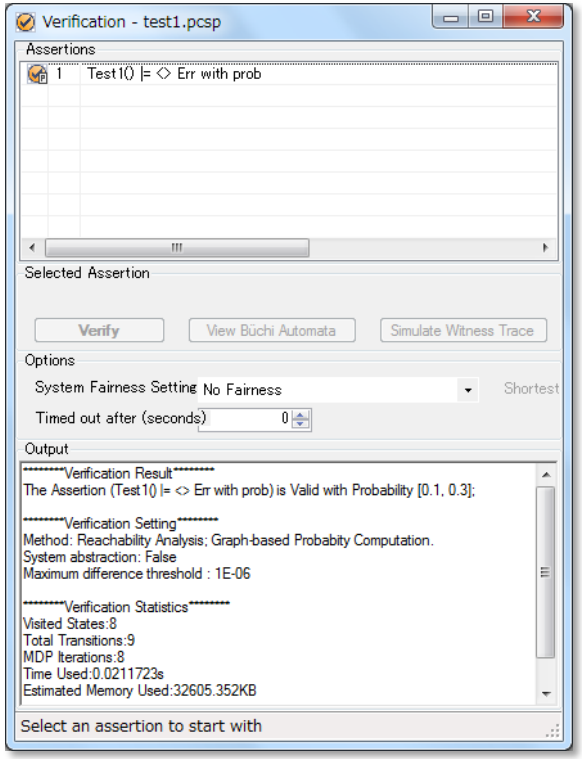
まとめ

- 確率時間モデル検査

まとめ

- 時間と確率を考慮した並行システムの確率時間CSPによる記述例と、確率時間モデル検査器による検証例を紹介した
- 単位時間当たりの故障確率の見積りなどに利用できる

PAT



PRISM