Project Report
Class: ECE/CS 5745/6745 - Testing and Verification of Digital Circuits - Fall 2008

A C++ Implementation of an Efficient Algorithm for Labeled Transition System Minimization Based on Bisimulation Equivalence

Eliyah Kilada, Eliyah.Kilada@utah.edu

I. Introduction

In theoretical computer science a bisimulation is a binary relation between labeled transition systems, associating systems which behave in the same way in the sense that one system simulates the other and vice-versa [4]. Bisimulation find applications in the formal verification of concurrent systems. For example, to check the equivalence of an implementation of a certain system with respect to its specification model. Usually, the process of finding a bisimulation equivalence between two labeled transistion systems includes two main steps. First, to minimize each one of them to its canonical form, and, then, perform the comparison between the canonical forms [1]. This project was concerned with the first step, which is, reducing a labeled transition system to a canonical form.

II. Labeled Transition System (LTS) Minimization

A. LTS Definition

A labeled transition system S is defined as S = (Q, A, T, q_0) where Q is a set of states, A is a finite set of actions, $T \subseteq Q \times A \times Q$ is the transition relation and q_0 is the initial state.

B. Bisimulation Definition

According to [1], bisimulation could be formally defined as follows: Given a labeled transition system S = (Q, A, T, q_0), a binary relation $\rho \subseteq Q \times Q$ is a bisimulation if and only if:

$$\forall (p_1, p_2) \in \rho . \forall a \in A.$$
$$\forall r_1 . \left(p_1 \xrightarrow{a} r_1 \Rightarrow \exists r_2 . \left(p_2 \xrightarrow{a} r_2 \wedge (r_1, r_2) \in \rho\right)\right) \wedge$$
$$\forall r_2 . \left(p_2 \xrightarrow{a} r_2 \Rightarrow \exists r_1 . \left(p_1 \xrightarrow{a} r_1 \wedge (r_1, r_2) \in \rho\right)\right)$$

C. LTS Minimization Algorithm Overview

This task takes an LTS and divides it into its coarsest blocks (i.e., least number of blocks) so that each block consists of a set of bisimualtion-equivalent states.

D. Algorithm Example

An example of an input to the LTS Minimization task [1] is shown in Fig. 1(a). The algorithm will divide this LTS into its coarsest blocks as shown in Fig.1(b)., where each block consists of a set of bisimualtion-equivalent states. Then, the algorithm should produce the minimized graph as in Fig.1(c).

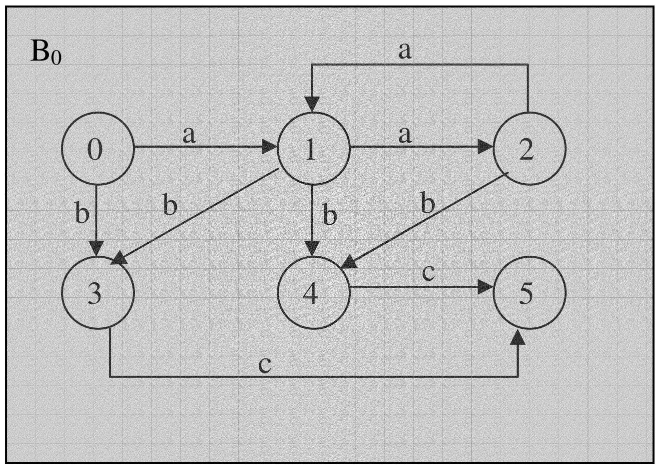

Figure 1(a): An example of an input LTS to the minimzation task.

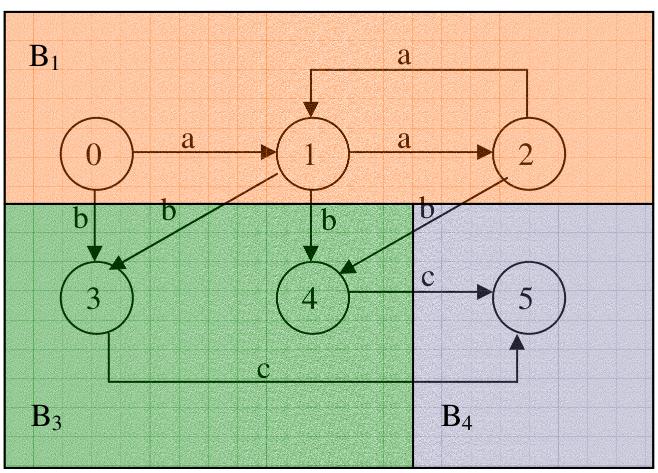

Figure 1(b): The LTS coarsest blocks.

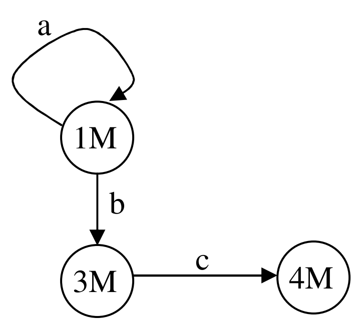

Figure 1(c): The minimized LTS.

Figure 1. Example of bisimulation-based minimization of an LTS

III. Some Definitions

A Block is a set of states. For example, in Fig.1, B_3={3,4}.

A Partition is a set of Blocks. The Blocks that constitute the Partition are mutually exclusive (i.e., don't have states in common) and their union constitutes the graph universe (i.e., set Q). For example, in Fig.1, P={B_1, B_3, B_4}, Q={0,1,2,3,4,5}, $Q = B_1 \cup B_3 \cup B_4$.

To define graph transitions we use the following terminology: $T_\alpha[p] = \{q\}$ means an α-transition from state p to state q. For example, in Fig.1, $T_a[0] = \{1\}$ and $T_b[1] = \{3,4\}$. Similarly, $T_a^{-1}[q] = p$ means an inverse α-transition. For example, in Fig.1, $T_b^{-1}[4] = \{1\}$. We also define an inverse transition for Block B and action α as follows: $T_a^{-1}[B] = \cup\{T_a^{-1}[q] \mid q \in B\}$ Back to the example in Fig.1, $T_b^{-1}[B_3] = \{0,1,2\}$

A set of set called Splitters, W, is used to contains those blocks that are going to be used to split the Partition. W should become more clear in the next two Sections.

An Info map is defined for a Block B, state p and action a as follows:

$\text{Info}_{\text{B}}(a, p) = |T_a[p] \cap B|$

$\text{Info}_{\text{B}3}(b,1) = 2$

IV. Pseudo Code of the Algorithm

In this section we introduce a pseudo code of the algorithm we used to do the LTS bisimulation-based minimization. It's almost the same as that described in [1].

```
Choose & remove any splitter (S) in W
    Case S is a simple splitter composed only of B
        For each α∈A
            Construct I = {X₁ | ∃X ∈ P ∧ X₁ = X ∩ T_α⁻¹[B] ≠ φ}
            //i.e., I is a set of those blocks in P that are going to
            be splitted by B
            For each block X in I:
                Compute X₁ = X ∩ T_α⁻¹[B]
                Compute X₂ = X − T_α⁻¹[B]
                Compute map info_B
                Check (X,X₁,X₂) & Update P and W (if required):
                    If ((X₁==0) || (X₂==0))
                        Do nothing! //No split
                    If ((X₁>0) && (X₂>0))
                        Remove X from P.
```

```
                Add X_1 to P.
                Add X_2 to P.
                Add (X,X_1,X_2) to W.
```

Case S is a compound splitter composed of (B,B_i,B_{ii}) // $B = B_i \cup B_{ii}$, assume $|B_i|<|B_{ii}|$

```
    For each α ∈ A
        Construct I = {X | X ∈ P ∧ X ⊆ T_α^{-1}[B]}
        //i.e., I is a set of those blocks in P that are going to
        be splitted by S
        // NB: P is already refined w.r.t. B.
        Calculate map info_Bi
        For each block X in I:
            Compute (X_1,X_2,X_3)  as follows:
                X_1: Set of states in X that goes to B_i but not
            to B_ii with action α.
                X_2: Set of states in X that goes to B_ii but not
            to B_i with action α.
                X_3: Set of states in X that goes to both B_i and
            B_ii.
                // This 3 subblocks are calculated directly from
            info_B and info_B1 as follows:
                For each state s in block X:
                    If (info_Bi[s][α]== info_B[s][α])
                        Add s to X_1
                        info_Bii[s][α]=0
                    If (info_Bi[s][α]== 0)
                        Add s to X_2
                        info_Bii[s][α]=info_B[s][α]
                    If ((info_Bi[s][α]> 0) &&( info_Bi[s][α]<
                info_B[s][α] ))
                        Add s to X_3
                        info_Bii[s][α]=info_B[s][α]- info_Bi[s][α]

            Check (X_1,X_2,X_3) & Update P and W (if required):
                If ((X==X_1) || (X==X_2)|| (X==X_3))
                    Do nothing! //No split
                Else
                    Replace X in P by non-null X_1,X_2,X_3
                    Add nun-null X_1,X_2,X_3 to W in the same way
                    as in the simple splitter case, except
                    that if all X_1,X_2,X_3 are non-null add the
                    following to W: (X,X_1,X_23),(X_23,X_2,X_3)
```

V. C++ Implementation and Minimization Example

The Pseudo code has been implemented in C++. Following is a detailed LTS minimization example:

An input LTS shown in Fig.1(a) is passed to the minimization algorithm. Initially, both Partition, *P*, and Splitters, *W*, will contain the graph universe, B_0.

Partition(P)={B_0}; Splitters(W)={B_0};

B_0={0,1,2,3,4,5}

A={a,b,c}

```
Choose & remove any splitter (S) in W
```
W={}, working with Splitter B_0

```
    Case S is a simple splitter composed only of B:
```
B_0 is a Simple Splitter

```
        For each
```
$\alpha \in A$ Start with action a, $T_a^{-1}[B_0] = \{0,1,2\}$

```
            Construct
```
$I = \{X_1 \mid \exists X \in P \wedge X_1 = X \cap T_\alpha^{-1}[B] \neq \varphi\}$

```
            //i.e., I is a set of those blocks in P that are going to be splitted
            by B
```
I={B_0}

```
            For each block X in I:
```
X=B_0

```
                Compute
```
$X_1 = X \cap T_\alpha^{-1}[B]$ X_1={0,1,2}

```
                Compute
```
$X_2 = X - T_\alpha^{-1}[B]$ X_2={3,4,5}

```
                Compute map info_B
```
infoB_0[0][a]=1, infoB_0[1][a]=1, infoB_0[2][a]=1

infoB_0[3][a]=0, infoB_0[4][a]=0, infoB_0[5][a]=0

```
                Check (X,X1,X2) & Update P and W (if required):
                    If ((X1==0) || (X2==0))
```
FALSE

```
                        Do nothing! //No split
                    If ((X1>0) && (X2>0))
```
TRUE

```
                        Remove X from P.
                        Add X1to P.
                        Add X2to P.
                        Add (X, X1, X2) to W.
```
P={B_1,B_2}; W={(B_0,B_1,B_2)}

Where B_1={0,1,2}, B_2={3,4,5}

After refinement with respect to B_0 and action a, partition (P) will look like the following

P={B_1,B_2}

W={(B_0,B_1,B_2)}

At this point, the LTS Partition, P, should look like Fig 2.

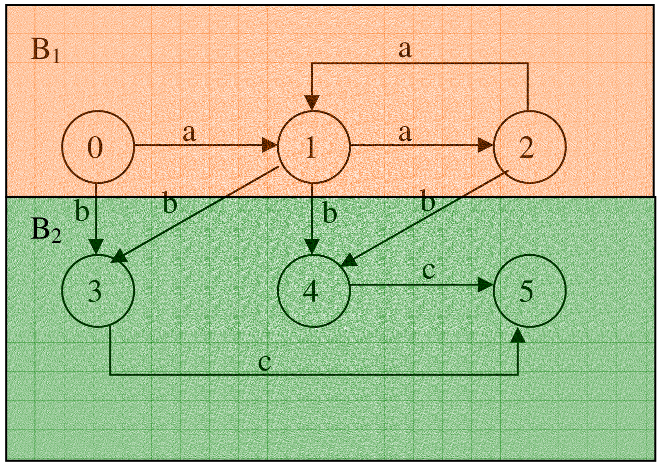

Figure 2: The LTS Partition after refinement with respect to Block B_0, action *a*.

Choose & remove any splitter (*S*) in *W* W={(B_0,B_1,B_2)} , Still working with Splitter B_0

Case *S* is a simple splitter composed only of *B* : Still B_0 is a Simple Splitter

For each $\alpha \in A$ Next is action b, $T_b^{-1}[B_0] = \{0,1,2\}$

Construct $I = \{X_1 \mid \exists X \in P \wedge X_1 = X \cap T_\alpha^{-1}[B] \neq \varphi\}$

//i.e., *I* is a set of those blocks in *P* that are going to be splitted by *B*

I={B_1}

For each block *X* in *I*: X=B_1

Compute $X_1 = X \cap T_\alpha^{-1}[B]$ X_1={0,1,2}

Compute $X_2 = X - T_\alpha^{-1}[B]$ X_2={ }

Compute map info$_B$

infoB_0[0][b]=1, infoB_0[1][b]=2, infoB_0[2][b]=1

infoB_0[3][b]=0, infoB_0[4][b]=0, infoB_0[5][b]=0

Check (*X*,X_1,X_2) & Update *P* and *W* (if required):

If ((X_1==0) || (X_2==0)) TRUE

Do nothing! //No split

If ((X_1>0) && (X_2>0))

Remove *X* from *P*.

Add X_1 to *P*.

Add X_2 to *P*.

Add (*X*,X_1,X_2) to *W*.

After refinement with respect to B_0 and action b, P and W will not change.

P={B_1,B_2}

W={(B_0,B_1,B_2)}

Choose & remove any splitter (S) in W W={(B_0,B_1,B_2)} , Still working with Splitter B_0

Case S is a simple splitter composed only of B: Still B_0 is a Simple Splitter

For each $\alpha \in A$ Next is action c, $T_c^{-1}[B_0] = \{3,4\}$

Construct $I = \{X_1 \mid \exists X \in P \wedge X_1 = X \cap T_\alpha^{-1}[B] \neq \varphi\}$

//i.e., I is a set of those blocks in P that are going to be splitted by B

I={B_2}

For each block X in I: X=B_2

Compute $X_1 = X \cap T_\alpha^{-1}[B]$ X_1={3,4}

Compute $X_2 = X - T_\alpha^{-1}[B]$ X_2={5}

Compute map $info_B$

$infoB_0[0][c]=0$, $infoB_0[1][c]=0$, $infoB_0[2][c]=0$

$infoB_0[3][c]=1$, $infoB_0[4][c]=1$, $infoB_0[5][c]=0$

Check (X,X_1,X_2) & Update P and W (if required):

If ((X_1==0) || (X_2==0)) FALSE

Do nothing! //No split

If ((X_1>0) && (X_2>0)) TRUE

Remove X from P.

Add X_1 to P.

Add X_2 to P.

Add (X,X_1,X_2) to W.

P={B_1,B_3, B_4}; W={(B_0,B_1,B_2),(B_2,B_3, B_4)}

Where B_1={0,1,2}, B_2={3,4,5}, B_3={3,4}, B_4={5}

After refinement with respect to B_0 and action c, partition (P) will look like the following
P={B_1,B_3, B_4};
W={(B_0,B_1,B_2),(B_2,B_3, B_4)}
Where B_1={0,1,2}, B_2={3,4,5}, B_3={3,4}, B_4={5}
At this point, the LTS Partition, P, should look like Fig. 1(b).

Choose & remove any splitter (S) in W W={(B_2,B_3, B_4)} // working with Splitter (B_0,B_1,B_2)

Case S is a compound splitter composed of (B,B_i,B_{ii}) // $B = B_i \cup B_{ii}$, assume $|B_i|<|B_{ii}|$

Splitter (B_0,B_1,B_2) is a compound splitter. B=B_0, B_i=B_1, B_{ii}=B_2

For each $\alpha \in A$ Start with action a, $T_a^{-1}[B_0] = \{0,1,2\}$

Construct $I = \{X \mid X \in P \wedge X \subseteq T_\alpha^{-1}[B]\}$

//i.e., I is a set of those blocks in P that are going to be splitted by S

// NB: P is already refined w.r.t. B.

I={B_1}

Calculate map $info_{Bi}$

$infoB_1[0][a]=1$, $infoB_1[1][a]=1$, $infoB_1[2][a]=1$

$infoB_1[3][a]=0$, $infoB_1[4][a]=0$, $infoB_1[5][a]=0$

For each block X in I: only B_1

```
Compute (X1,X2,X3)  as follows:
       X1: Set of states in X that goes to Bi but not to Bii with
action α.
       X2: Set of states in X that goes to Bii but not to Bi with
action α.
       X3: Set of states in X that goes to both Bi and Bii.
       // This 3 subblocks are calculated directly from infoB and
infoB1 as follows:
       For each state s in block X: // Let’s take s=0 as an example
              If (infoBi[s][α]== infoB[s][α]) TRUE: infoB1[0][a] == infoB0[0][a]=1
                     Add s to X1 Add s=0 to X1
                     infoBii[s][α]=0 infoB2[0][a]=0
              If (infoBi[s][α]== 0) FALSE:  infoB1[0][a]= 1
                     Add s to X2
                     infoBii[s][α]=infoB[s][α]
              If ((infoBi[s][α]> 0) &&( infoBi[s][α]< infoB[s][α]
              ))  FALSE: ((infoB1[0][a]> 0)  but (infoB1[0][a] == infoB0[0][a] ))
                     Add s to X3
                     infoBii[s][α]=infoB[s][α]- infoBi[s][α]
```

Applying the algorithm to all states in B_1, we get the following:

$X1=\{0,1,2\}=B_1$

$X2=\{\}$

$X3=\{\}$

```
Check (X1,X2,X3) & Update P and W (if required):
       If ((X==X1) || (X==X2)|| (X==X3)) TRUE
              Do nothing! //No split
       Else
              Replace X in P by non-null X1,X2,X3
              Add nun-null X1,X2,X3 to W in the same way as in the
              simple splitter case, except that if all X1,X2,X3
              are non-null add the following to W:
              (X,X1,X23),(X23,X2,X3)
```

After refinement with respect to (B_0,B_1,B_2) and action a, P and W will not change.

$P=\{B_1,B_3, B_4\}$;

$W=\{(B_2,B_3, B_4)\}$

Where $B_1=\{0,1,2\}$, $B_2=\{3,4,5\}$, $B_3=\{3,4\}$, $B_4=\{5\}$

Choose & remove any splitter (S) in W $W=\{(B_2,B_3, B_4)\}$ // Still working with Splitter (B_0,B_1,B_2)

Case S is a compound splitter composed of (B,B_i,B_{ii}) // $B = B_i \cup B_{ii}$, assume $|B_i|<|B_{ii}|$

Still: Splitter (B_0,B_1,B_2) is a compound splitter. $B=B_0$, $B_i=B_1$, $B_{ii}=B_2$

For each $\alpha \in A$ Next with action b, $T_b^{-1}[B_0] = \{0,1,2\}$

Construct $I = \{X \mid X \in P \wedge X \subseteq T_\alpha^{-1}[B]\}$

//i.e., I is a set of those blocks in P that are going to be splitted by S
// NB: P is already refined w.r.t. B.
I={B_1}
Calculate map $info_{Bi}$
$infoB_1[0][b]=0$, $infoB_1[1][b]=0$, $infoB_1[2][b]=0$
$infoB_1[3][b]=0$, $infoB_1[4][b]=0$, $infoB_1[5][b]=0$
For each block X in I: only B_1

Compute (X_1,X_2,X_3) as follows:

X_1: Set of states in X that goes to B_i but not to B_{ii} with action α.
X_2: Set of states in X that goes to B_{ii} but not to B_i with action α.
X_3: Set of states in X that goes to both B_i and B_{ii}.
// This 3 subblocks are calculated directly from $info_B$ and $info_{B1}$ as follows:
For each state s in block X:

If $(info_{Bi}[s][\alpha]== info_B[s][\alpha])$
    Add s to X_1
    $info_{Bii}[s][\alpha]=0$
If $(info_{Bi}[s][\alpha]== 0)$
    Add s to X_2
    $info_{Bii}[s][\alpha]=info_B[s][\alpha]$
If $((info_{Bi}[s][\alpha]> 0)$ &&$(info_{Bi}[s][\alpha]< info_B[s][\alpha]))$
    Add s to X_3
    $info_{Bii}[s][\alpha]=info_B[s][\alpha]- info_{Bi}[s][\alpha]$

Applying the algorithm to all states in B_1, we get the following:
X1={}
X2={0,1,2}=B_1
X3={}

Check (X_1,X_2,X_3) & Update P and W (if required):

If $((X==X_1) || (X==X_2)|| (X==X_3))$ TRUE
    Do nothing! //No split
Else
    Replace X in P by non-null X_1,X_2,X_3
    Add nun-null X_1,X_2,X_3 to W in the same way as in the simple splitter case, except that if all X_1,X_2,X_3 are non-null add the following to W:
    $(X,X_1,X_{23}),(X_{23},X_2,X_3)$

After refinement with respect to (B_0,B_1,B_2) and action a, P and W will not change.
P={B_1,B_3, B_4};
W={(B_2,B_3, B_4)}
Where B_1={0,1,2}, B_2={3,4,5}, B_3={3,4}, B_4={5}

Choose & remove any splitter (S) in W W={(B$_2$,B$_3$, B$_4$)} // Still working with Splitter (B$_0$,B$_1$,B$_2$)

Case S is a compound splitter composed of (B,B_i,B_{ii}) // $B = B_i \cup B_{ii}$, assume $|B_i|<|B_{ii}|$

Still: Splitter (B$_0$,B$_1$,B$_2$) is a compound splitter. B=B$_0$, B$_i$=B$_1$, B$_{ii}$=B$_2$

For each $\alpha \in A$ Next with action c, $T_c^{-1}[B_0] = \{3,4\}$

Construct $I = \{X \mid X \in P \wedge X \subseteq T_\alpha^{-1}[B]\}$

//i.e., I is a set of those blocks in P that are going to be splitted by S

// NB: P is already refined w.r.t. B.

I={B$_3$}

Calculate map $info_{Bi}$

$infoB_1[0][c]=0$, $infoB_1[1][c]=0$, $infoB_1[2][c]=0$

$infoB_1[3][c]=0$, $infoB_1[4][c]=0$, $infoB_1[5][c]=0$

For each block X in I: only B$_3$

Compute (X_1,X_2,X_3) as follows:

X_1: Set of states in X that goes to B_i but not to B_{ii} with action α.

X_2: Set of states in X that goes to B_{ii} but not to B_i with action α.

X_3: Set of states in X that goes to both B_i and B_{ii}.

// This 3 subblocks are calculated directly from $info_B$ and $info_{B1}$ as follows:

```
For each state s in block X:
      If (info_Bi[s][α]== info_B[s][α])
            Add s to X1
            info_Bii[s][α]=0
      If (info_Bi[s][α]== 0)
            Add s to X2
            info_Bii[s][α]=info_B[s][α]
      If ((info_Bi[s][α]> 0) &&( info_Bi[s][α]< info_B[s][α]
      ))
            Add s to X3
            info_Bii[s][α]=info_B[s][α]- info_Bi[s][α]
```

Applying the algorithm to all states in B$_3$, we get the following:

X1={ }

X2={3,4}=B$_3$

X3={ }

Check (X_1,X_2,X_3) & Update P and W (if required):

If $((X==X_1) \,||\, (X==X_2) \,||\, (X==X_3))$ TRUE

Do nothing! //No split

Else

Replace X in P by non-null X_1,X_2,X_3

Add nun-null X_1,X_2,X_3 to W in the same way as in the simple splitter case, except that if all X_1,X_2,X_3 are non-null add the following to W:

$(X,X_1,X_{23}),(X_{23},X_2,X_3)$

After refinement with respect to (B$_0$,B$_1$,B$_2$) and action a, P and W will not change.

P={B$_1$,B$_3$, B$_4$};

W={(B$_2$,B$_3$, B$_4$)}

Where B$_1$={0,1,2}, B$_2$={3,4,5}, B$_3$={3,4}, B$_4$={5}

Finally:

`Choose & remove any splitter (S) in W` W={} // Next working with Splitter (B_2,B_3, B_4)

We can easily see that, after refinement with respect to (B_2,B_3,B_4) and all actions a,b, and c, P and W will not change.

P={B_1,B_3, B_4};

W={(B_2,B_3, B_4)}

Where B_1={0,1,2}, B_2={3,4,5}, B_3={3,4}, B_4={5}

At this point, the LTS Partition, P, should look like Fig. 1(b). Proceeding from Fig. 1(b), the algorithm will generate a minimized LTS as in Fig. 1(c). This is done by replacing each block by a new state in the minimized graph.

References:

[1] J.-C. Fernandez, "An Implementation of an Efficient Algorithm for Bisimulation Equivalence," Science of Computer Programming, vol. 13, pp. 219-236, 1989/90.

[2] R. Paige and R.E. Tarjan, "Three Partition Refinement Algorithms," SIAM J. Computing, vol. 16, no. 6, pp. 973-989, 1987.

[3] Milner, Robin (1989). Communication and Concurrency.. Prentice Hall. ISBN 0-13-114984-9.

[4] "http://en.wikipedia.org/wiki/Bisimulation"