Source code transformation rules for Strong Mobility of BPEL processes: Technical Report

Soumaya Marzouk and Mohamed Jmaiel

ReDCAD Laboratory
National School of Engineers of Sfax
BP1173, 3038 Sfax, Tunisia
Soumaya.Marzouk@redcad.org, Mohamed.Jmaiel@enis.rnu.tn

December 2011

Contents

1 Introduction **2**

2 Strong Mobility Architecture **2**
2.1 The Web Service Invocation Manager (WSIM) . . . 2
2.2 The Web Service Checkpoint Manager (WSCM) . . . 2
2.3 The checkpointing, result, mobility, and recovery aspects . . . 3
2.4 The platform . . . 3

3 Source Code Transformation Rules **4**
3.1 Definition of an orchestration process state . . . 4
3.2 Generic Source code transformation rules . . . 5
3.2.1 Simple instructions . . . 5
3.2.2 Conditional structures . . . 6
3.2.3 Loops structures . . . 7
3.3 Specific Source code transformation rules . . . 7

4 Proof of correctness of Generic Source Code transformation rules **9**
4.1 Step 1: $migration = false$ . . . 9
4.1.1 Case of Simple Instructions . . . 9
4.1.2 Case of Conditional structures . . . 10
4.1.3 Case of Loop structures . . . 13
4.2 Step 2: $migration = true$ . . . 15
4.2.1 Case of simple instructions . . . 15
4.2.2 Case of conditional structures . . . 16
4.2.3 Case of loop structures . . . 19

5 Proof of correctness of Specific Source Code transformation rules **24**
5.1 Step 1: $migration = false$ . . . 25
5.2 Step 2: $migration = true$ . . . 27

6 Conclusion **31**

1 Introduction

Strong mobility of an orchestration process corresponds to the migration of all or a subset of its running instances from a host to another and their resumption, on the destination host, starting from their last captured checkpoints. Strong mobility may occur in case of failure of the orchestration process hosting node and it consists, in this case, in recovering all interrupted instances on a more reliable node. Strong mobility may occur also in case of performance degradation of the hosting node which necessitates the migration of only a subset of the running instances in order to handle the node overload.

Our Strong mobility solution relies mainly on source code transformation, AOP, and additional management services like the Web Service Checkpoint Manager (WSCM) and the Web Service Invocation Manager (WSIM).

Source code transformation enables the preparation and the maintenance of the orchestration process state in order to allow checkpointing when required. Aspects ensure dynamic checkpointing, recovery, and migration of any orchestration process belonging to a service oriented application even if its partners are also orchestration processes. The additional management services ensure checkpoint availability and mobility transparency.

In this technical report, we begin with presenting an overview on the architecture of our strong mobility solution as well as the different involved entities. Then, we will detail the employed source code transformation rules and we will present the proof verifying that these latter preserve the semantics of the original process.

2 Strong Mobility Architecture

The proposed architecture enabling strong mobility of orchestration processes is described in Figure 1. This architecture includes two main management services respectively called Web Service Invocation Manager (WSIM) and Web Service Checkpoint Manager (WSCM). It also includes four kinds of aspects: checkpointing, results, mobility, and recovery aspects. In the following, we will detail the components of this architecture as well as the used implementation platform.

2.1 The Web Service Invocation Manager (WSIM)

is a service deployed between the client and the main orchestration process[1]. It deals with routing messages between the main orchestration process and its clients. This service is essential in case of main orchestration process migration, since it ensures the routing of the response resulting from the new main orchestration process host towards the initial client. In spite of this centralized architecture which makes all invocations pass through the WSIM, the latter does not constitute a bottleneck. In fact, the WSIM does not make any treatment apart from invoking the main orchestration process, waiting for its response, and routing it to the client.

2.2 The Web Service Checkpoint Manager (WSCM)

is a service responsible for managing the checkpoints of the orchestration process instances as well as their results. Indeed, when the checkpointing aspect is deployed, every concerned instance calls the WSCM, through the checkpointing aspect code, for saving a copy of its execution state (checkpoint). Thus, the WSCM saves all captured checkpoints on a remote database in order to enable their possible use for resuming interrupted instances. Moreover, when an orchestration process reaches its end, the result aspect is executed to save the returned result on the WSCM.

[1]We use the term "main orchestration process" to refer to the BPEL process which is directly invoked by the client and not invoked by an other BPEL process.

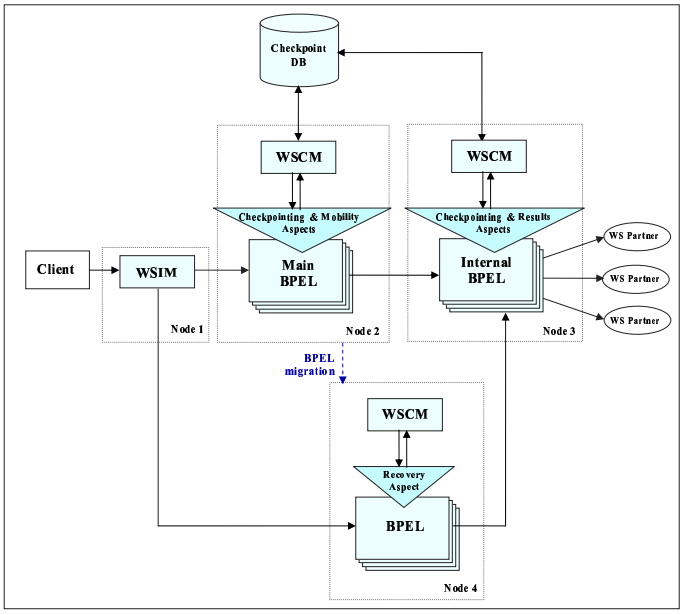

Figure 1: Architecture supporting strong mobility of orchestration processes.

2.3 The checkpointing, result, mobility, and recovery aspects

are essential parts of the strong mobility architecture. In fact, they ensure the flexibility of this architecture since aspects may be deployed and un-deployed according to the execution context. They encapsulate codes enabling the enforcement of checkpointing and mobility. In particular, the result aspect is responsible of saving the process result. While mobility aspect enforces process instances mobility, the recovery aspect ensures execution resumption after mobility.
Finally, the checkpointing aspect is used to capture a checkpoint. For that, we suggest different aspects implementing a variety of checkpointing techniques such as checkpointing at the next natural synchronization barrier, forced checkpoint, immediate checkpoint and migration. Checkpointing at the natural synchronization barrier corresponds to waiting until reaching a sequential execution (i.e. execution with no parallel branches) to enact the checkpoint. In fact, at such position, capturing the checkpoint is simple and requires only sending the current state to the WSCM after updating the activity counter. Contrary, forced checkpointing involves synchronizing all parallel branches, before capturing the execution state. This will guarantee a consistent checkpoint (see [2] for more details about synchronization). The checkpoint at migration is similar to the forced checkpointing scenario, but it additionally requires forcing the migration by throwing a notification to the WSIM and stopping the orchestration process execution just after checkpoint capture. These aspects will be deployed at runtime which makes it possible to dynamically select not only the adequate time point to make the checkpoint but also the appropriate technique to enact it. Once the checkpointing aspect is deployed, it saves the orchestration process execution state and sends it to the WSCM. Technical details about our aspect-oriented solution may be found in [1].

2.4 The platform

enabling the deployment of such architecture necessitates an AO4BPEL engine deployed on each node hosting an orchestration process. Moreover, for high availability, dedicated nodes should be used for hosting the WSIM and the database. The WSCM may be deployed on the same node as the orchestration process or on a different one. A unique WSCM may be employed for more than one orchestration process, but one should take precaution to avoid the risk of its overload. For instance, in Figure 1, the WSIM is deployed

on a dedicated host. For each BPEL process, we deploy a dedicated WSCM. Both BPEL process and its corresponding WSCM are deployed on the same host.

3 Source Code Transformation Rules

Source code transformation rules aim to maintain up to date the execution state of a BPEL instance, in order to enable its capture at any point of its execution. For that, the first step of source code transformation consists in adding a set of variables representing the BPEL instance execution state. The second step corresponds to the application of a set of source code transformation rules in order to maintain up to date these variables.

In the next sections, we will detail the structure of the execution state of a BPEL instance. Then, we will present the employed source code transformation rules. Here, we distinguish transformation rules targeting generic algorithmic structures and specific transformation rules which focus on BPEL specific activities.

3.1 Definition of an orchestration process state

The state of an orchestration process corresponds to a set of data describing its execution progress and which is sufficient for resuming the orchestration process execution in case of interruption.

```
<!-- Instance identifier -->
    <variable name="OPIdentifier" type="xsd:int"/>
<!-- Original process variables -->
    <variable name="getResponse" type="xsd:String"/>
    <variable name="sendPO" type="xsd:String"/>
    <variable name="getConfirmation" type="xsd:String"/>
<!-- Activity counter variables -->
        <variable name="Pos1" type="xsd:int"/>
        <variable name="Pos2" type="xsd:int"/>
<!-- Reception management variables -->
    <variable name="received1" type="xsd:int"/>
<!-- Links variables -->
    <variable name="link1" type="xsd:int"/>
    <variable name="link2" type="xsd:int"/>
<!-- Scope variables -->
    <variable name="getResponseCompensation" type="xsd:String"/>
<!-- Checkpoint and mobility management variables -->
        <variable name="CheckpointIndicator" type="xsd:int"/>
        <variable name="NbBranches" type="xsd:int"/>
        <variable name="NbSynch" type="xsd:int"/>
```

Figure 2: Variables for checkpoint data of a BPEL process

Those data will be saved in variables belonging to the transformed orchestration process. They include the original process variables as well as additional variables helping consistency and mobility management. These additional variables, written in bold in Figure 2, are: (1) the orchestration process instance identifier defining for which instance the checkpoint belongs, (2) the artificial activity counters which point to the actual running activities[2], (3) the reception management variables which ensure re-calling invocations for which no reply is received before migration, (4) links variables which ensure the preservation of the execution order of activities when links are employed, (5) scope variables which are responsible of compensation and fault handlers, and finally, (6) checkpoint and mobility management variables handling checkpoint capture and mobility launching. All these variables represent the checkpoint data of an orchestration process instance.

[2]Note that there are as many activity counters as branches (the number of activity counters is equal to the maximal number of parallel branches)

These variables are perpetually updated in order to maintain the execution state of the orchestration process instance up to date. For that, we define a set of rules that transform the process code by inserting instructions for updating these checkpoint variables. We distinguish specific rules for the basic activities (*invoke, receive, assign, wait ...*) and others for the structured activities (*while, switch ...*).

In the next subsection, we will detail how orchestration process state is prepared and maintained using source code transformation rules. In addition, we will present the proof verifying the correctness of these transformation rules.

3.2 Generic Source code transformation rules

Ensuring strong mobility of an orchestration process consists in integrating the capacity to capture its execution state (checkpoint), as well as the possibility of loading a checkpoint (re-establish) in order to resume the execution starting from it. In our approach, these functionalities are carried out based on aspects and pieces of code inserted into the orchestration process source code. Indeed, the code of the process is automatically instrumented in order to maintain perpetually an updated state of each running instance. Thus, aspects may be deployed at any execution time in order to capture this state, recover it, or launch instance mobility. In this section, we will present source code transformation rule which target common algorithmic structures such as conditional and loop structures.

In the following, we present our transformation rules for BPEL version 1.1 since we will deploy it on the AO4BPEL engine. However, they can also be applied for any orchestration process description language.

3.2.1 Simple instructions

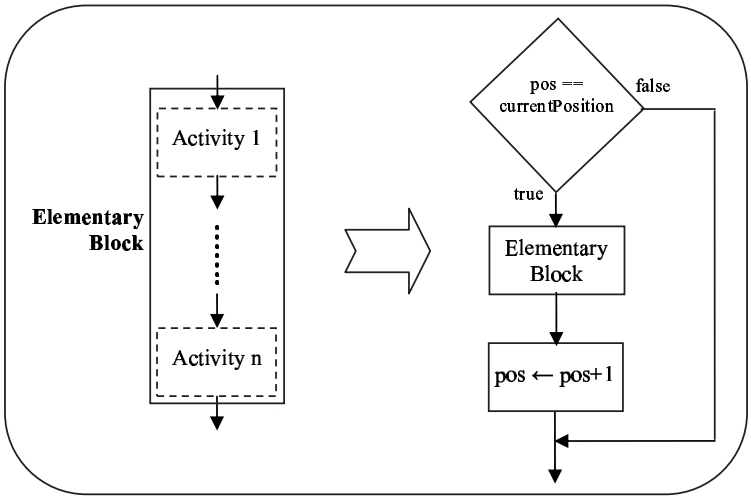

Figure 3: Transformation rule of basic activities.

The transformation rule for a block of basic activities labels these activities with a counter (see Figure 3). It consists in (1) updating the activity counter to the position of the next activity to be carried out and (2) testing if the current activity corresponds to the one which should be carried out. Thus, this rule will be applied to the *invoke, receive, reply, wait*, and *assign* activities. In this way, the transformed process will execute this block only if the current position is equal to its label. This will ensure that this block will not be re-executed after migration.

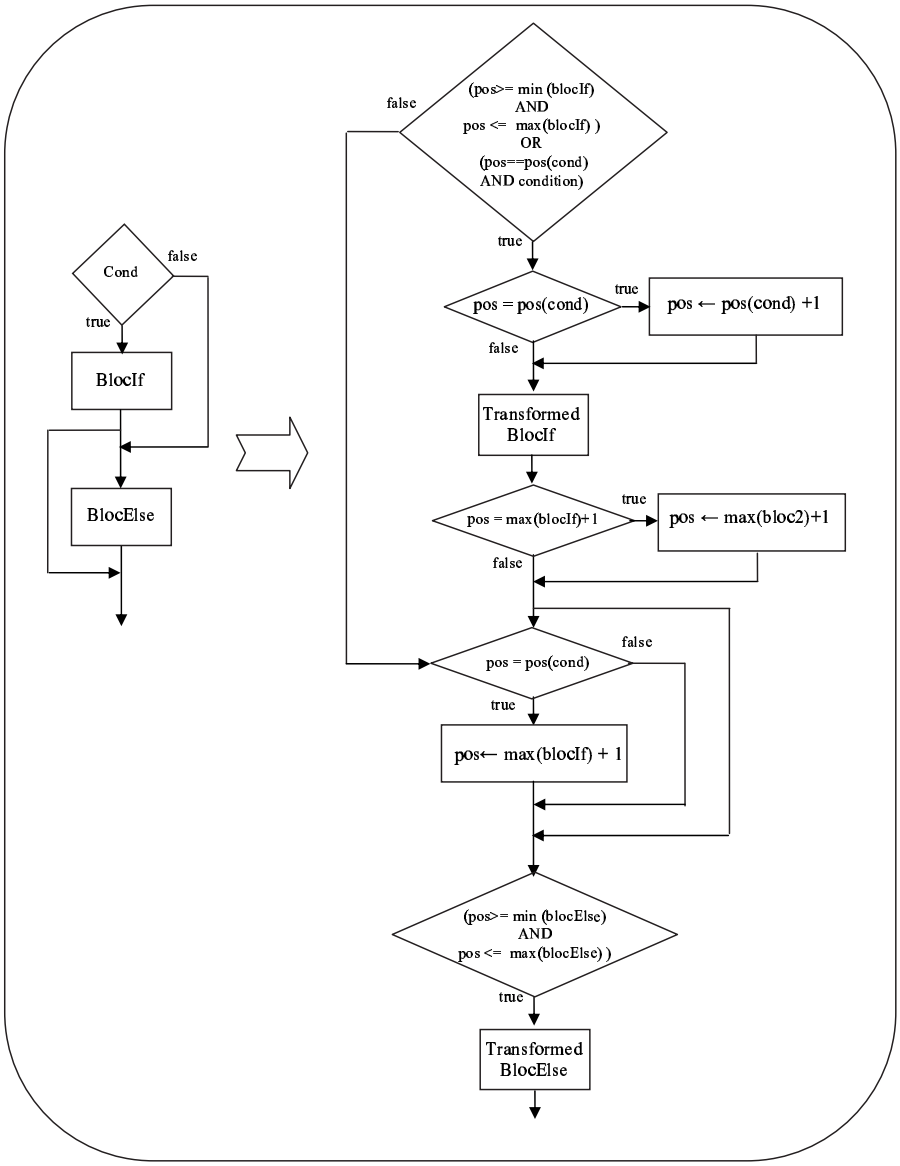

Figure 4: Transformation rule of an if-else structure.

3.2.2 Conditional structures

Figure 4 presents the transformation rule of an if-else structure. This transformation ensure that the if block is executed if both the activity counter *pos* corresponds to the first position of the conditional structure $pos(cond)$ and the condition is satisfied. Also, in case of interruption and when the last captured checkpoint was done during the execution of the if block, the execution of this latter after resumption necessitates that the current position is between the first position of the if block $min(if)$ and the last position of the if block $max(if)$. After the execution of the if block, the activity counter will point to the instruction located after the else block $max(else) + 1$. if the entry condition of the if block is not satisfied, the activity counter will point to the first position in the else block $min(else)$. The else block will be executed if the activity counter is located between the first position of the else block $min(else)$ and the last position of the else

block $max(else)$. Thus, if the last captured checkpoint occurred during the execution of the else block, the resumption will be correctly done starting from the interruption point.

3.2.3 Loops structures

Figure 5 presents the transformation rule of the *while* structure. This rule guarantees that the access to the while structure is made only if the activity counter value corresponds to one of the position of activities located within the loop. Moreover, this transformation ensures that at the loop end, the activity counter value corresponds to the position of the next activity. Analogously, we defined the transformation rules for the *RepeatUntil* structure.

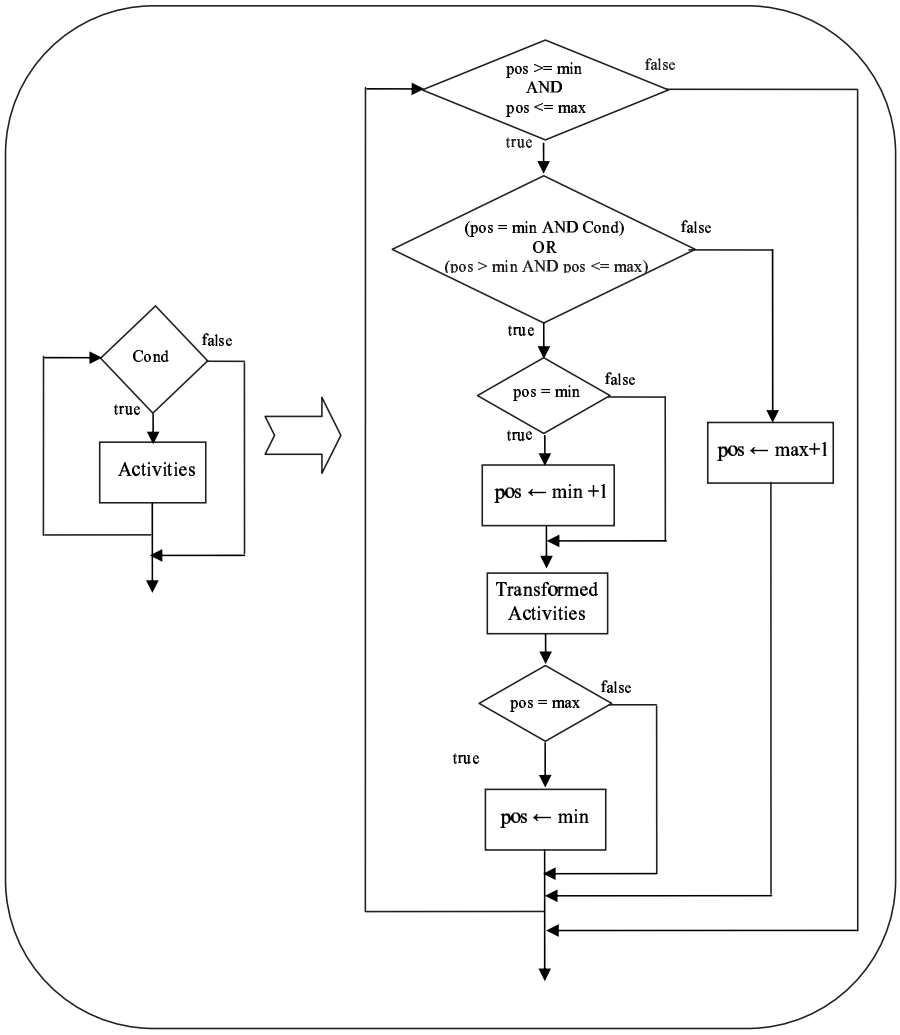

Figure 5: Transformation rule of a loop structure.

3.3 Specific Source code transformation rules

In this section, we will present specific transformation rules which corresponds to BPEL specific activities. We will focus on flow activity.

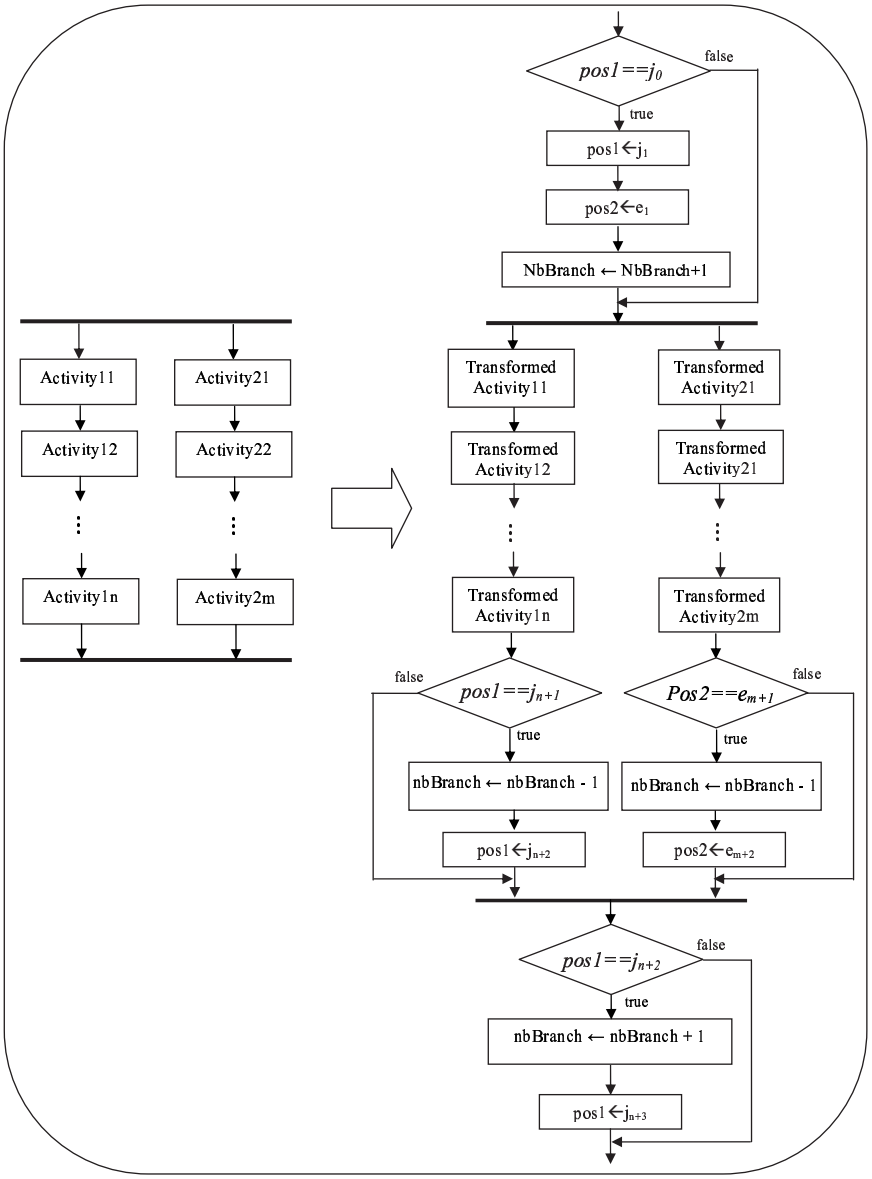

Figure 6: Transformation rule of a flow structure.

Flow structure enables the parallel execution of BPEL activities. The main transformation of this structure consists in updating the *nbBranch* variable which counts the current number of parallel branch. This value should be available at checkpointing time in order to manage the synchronization process (see [2] for more details).

As presented in Figure 6, the *nbBranch* value is updated by adding 1 at the *flow* structure beginning. Thereby, the number of parallel branches pass from 1 to 2. Moreover, it is decremented when a branch is terminated. So, at the end of the *flow* structure, the number of parallel branches becomes 0. Therefore, after the *flow* structure, the *nbBranch* value is incremented to become 1.

Making code transformation arises the need of ensuring that this latter preserves the semantics of initial

processes. For that, we present, in the next section, the proof verifying that the transformed process resulting from applying our source code transformation rules preserves the semantics of the original one.

4 Proof of correctness of Generic Source Code transformation rules

In order to prove that the original and the transformed codes have the same semantics, we make use of the Hoare logic []. Hence, we prove that when the original and transformed codes have the same pre-condition, the transformed code conserves the post-condition of the original one. In this proof, we consider that the original code may be written as follows:

$P = P(j_0, j_1 - 1); P(j_1, j_2 - 1); ...P(j_i, j_{i+1} - 1); ...P(j_n, j_{n+1} - 1)$

Where

- $P(j_i, j_{i+1} - 1)$ simple instruction or complex structure having $j_{i+1} - j_i$ as number of instructions
- $j_0 = 0$
- $\{Q_0\} \quad P \quad \{Qj_{n+1}\}$
- $\forall i \in \{0, ..., n\} \quad \{Qj_i\} \quad P(j_i, j_{i+1} - 1) \quad \{Qj_{i+1}\}$

We consider also that the transformed code corresponds to:

$Trans(P) = P_{Recovery}; Trans(P(j_0, j_1 - 1)); ...; Trans(P(j_n, j_{n+1} - 1))$

Where

```
P_Recovery : if (migration) then { pos = j_m; Q_0 = Qj_m;}
2            else  pos = 0 ;
```

Qj_m is the last saved post-condition before migration and j_m is the position of the next instruction to be executed after migration. Given these definitions, we should prove that $\{Q_0\} \quad Trans(P) \quad \{Qj_{n+1} \wedge pos = j_{n+1}\}$.

Our proof will be made in two steps: (1) the code is executed without any migration: $migration = false$, and (2) the code execution corresponds to a resumption after migration: $migration = true$.

4.1 Step 1: $migration = false$

In this case, we have : $\{Q_0\} \quad P_{Recovery} \quad \{Q_0 \wedge pos = 0\}$. So, we have to prove that :

$\forall i \in \{0, ..., n\}\{Qj_i \wedge pos = j_i\} \quad Trans(P(j_i, j_{i+1} - 1)) \quad \{Qj_{i+1} \wedge pos = j_{i+1}\}$

However, the result of the $Trans$ function depends on the structure of the original code. Therefore, a proof by structured induction will be performed. We start with simple instructions, then we make it for structured ones.

4.1.1 Case of Simple Instructions

We note the set of simple instructions S. For each I in S we have:

```
Trans(I): if (pos = k) then {I; pos = pos+1;}
```

Since instructions belonging to S are simple, we have $\forall i \in \{0, ..., n\} j_i = i$ and so$\{Q_i\} \quad P(i, i+1)) \quad \{Q_{i+1}\}$. That is, in order to achieve the proof for instructions belonging to S, we should prove that:

$\forall i \in \{0, ..., n\} \quad \{Q_i \wedge pos = i\} \quad Trans(P(i, i+1)) \quad \{Q_{i+1} \wedge pos = i + 1\}$

$$\dfrac{\dfrac{(A)}{\{Q_i \wedge pos = i \wedge pos = i\}P(i,i+1);pos = pos+1;\{Q_{i+1} \wedge pos = i+1\}} \quad \dfrac{false \rightarrow Q_{i+1} \wedge pos = i+1}{Q_i \wedge pos = i \wedge \neg pos = i \rightarrow Q_{i+1} \wedge pos = i+1}}{\dfrac{\{Q_i \wedge pos = i\}if(pos = i)then\{P(i,i+1);pos = pos+1;\}\{Q_{i+1} \wedge pos = i+1\}}{\{Q_i \wedge pos = i\}Trans(P(i,i+1))\{Q_{i+1} \wedge pos = i+1\}}}$$

$$\dfrac{\dfrac{\{Q_i\}P(i,i+1)\{Q_{i+1}\} \quad Q_i \wedge pos = i \rightarrow Q_i \quad Q_{i+1} \wedge pos = i \rightarrow Q_{i+1}}{\{Q_i \wedge pos = i\}P(i,i+1)\{Q_{i+1} \wedge pos = i\}} \quad \dfrac{\{pos = i\}pos = pos+1\{pos = i+1\}}{\{Q_{i+1}\neg pos = i\}pos = pos+1\{Q_{i+1}\neg pos = i+1\}}}{(A)}$$

This enables us to state the following results.

Result 1 :
$\forall i \in \{0,...,n\}$ and $P(i,i+1) \in S$,
$\{Q_i\}\ \ P(i,i+1)\ \ \{Q_{i+1}\}\ \ \Rightarrow\ \ \{Q_i \wedge pos = i\}\ \ Trans(P(i,i+1))\ \ \{Q_{i+1} \wedge pos = i+1\}$

Result 2 :
For any code P composed of instructions belonging to S:
$\{Q0\}\ \ P\ \ \{Qj_{n+1}\}\ \ \Rightarrow\ \ \{Q0 \wedge migration = false\}\ \ Trans(P)\ \ \{Qj_{n+1} \wedge pos = j_{n+1}\}$

Proof of Result2 Using structural induction:

- For $i = 0$, we have $P_0 = P(0,1)$ with $\{Q_0\}P(0,0)\{Q_1\}$. So, $Trans(P_0) = P_{Recovery};Trans(P(0,0))$. Moreover, we have $\{Q_0 \wedge migration = false\}P_{Recovery}\{Q_0 \wedge pos = 0\}$ and according to *Result1*, $\{Q_0 \wedge pos = 0\}Trans(P(0,0))\{Q_1 \wedge pos = 1\}$. Thus, we can conclude that $\{Q_0 \wedge migration = false\}Trans(P_0)\{Q_1 \wedge pos = 1\}$.
- Assume that $\{Q_0 \wedge migration = false\}Trans(P(0,n))\{Q_{n+1} \wedge pos = n+1\}$, let us prove that $\{Q_0 \wedge migration = false\}Trans(P(0,n+1))\{Q_{n+2} \wedge pos = n+2\}$.
  As $Trans(P(0,n+1)) = Trans(P(0,n));Trans(P(n+1,n+1))$; and $\{Q_{n+1} \wedge pos = n+1\}Trans(P(n+1,n+1))\{Q_{n+2} \wedge pos = n+2\}$, we can deduce that $\{Q_0 \wedge migration = false\}Trans(P(0,n+1))\{Q_{n+2} \wedge pos = n+2\}$

4.1.2 Case of Conditional structures

A conditional structure $P(j_i, j_{i+1} - 1)$ may be written as follows:

```
1 if (cond) then   P(j_{i+1},k);  else  P(k+1, j_{i+1}-1);
```

The transformation $Trans(P(j_i, j_{i+1} - 1)$ of this program corresponds to:

```
1 if((pos = j_i and cond) or (pos > j_i and pos <= k) ) then {
      if(pos = j_i) then pos = j_{i+1};
3     Trans(P(j_{i+1},k));
      if(pos = k+1) then  pos = j_{i+1}
5 } else if(pos = j_i) then pos=k+1;

7 if(pos > k and pos < j_{i+1}) then
      Trans(P(k+1, j_{i+1}-1));
```

We have :

- $\{Qj_i\}\ \ P(j_i, j_{i+1} - 1)\ \ \{Qj_{i+1}\}$,
- $\{Qj_i\}\ \ P(j_{i+1}, k)\ \ \{Qj_{i+1}\}$, and
- $\{Qj_i\}\ \ P(k+1, j_{i+1} - 1)\ \ \{Qj_{i+1}\}$

And we should prove that : $\{Qj_i \wedge pos = j_i\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$
For that, we should prove that:

- (i) $\{Qj_i \wedge pos = j_i \wedge cond\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$
- (ii) $\{Qj_i \wedge pos = j_i \wedge \neg cond\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

(i) Let us prove that $\{Qj_i \wedge pos = j_i \wedge cond\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

$$\dfrac{\dfrac{(A)\quad (B)}{\{Qj_i \wedge pos = j_i \wedge cond\} blocIf \{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad \dfrac{(F)}{\{Qj_{i+1} \wedge pos = j_{i+1}\} BlocElse \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_i \wedge pos = j_i \wedge cond\} Trans(P(j_i, j_{i+1} - 1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}$$

$$\dfrac{Qj_i \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k)) \rightarrow Qj_i \wedge pos = j_i \wedge cond \qquad (Y)}{\dfrac{\{Qj_i \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\} (X1) \{Qj_{i+1} \wedge pos = j_{i+1}\}}{(A)}}$$

Where (X1): if $(pos = j_i)$ then $pos = j_{i+1}; Trans(P(j_{i+1}, k))$; if $(pos = k + 1)$ then $pos = j_{i+1}$

$$\dfrac{(C)\quad (D)\quad (E)}{\{Q_i \wedge pos = j_i \wedge cond\} if\ (pos = j_i)\ then\ pos = j_{i+1};\ Trans(P(j_{i+1}, k));\ if\ (pos = k + 1)\ then\ pos = j_{i+1} \{Q_{i+1} \wedge pos = j_{i+1}\}}{(Y)}$$

$$\dfrac{\dfrac{\dfrac{(X2)\quad (X3)\quad (X4)}{\{Qj_i \wedge pos = j_i \wedge cond\} pos = j_{i+1} \{Qj_i \wedge pos = j_{i+1}\}}}{\{Qj_i \wedge pos = j_i \wedge cond \wedge pos = j_i\} pos = j_{i+1} \{Qj_i \wedge pos = j_{i+1}\}} \quad \dfrac{false \rightarrow Qj_i \wedge pos = j_{i+1}}{Qj_i \wedge pos = j_i \wedge cond \wedge \neg pos = j_i \rightarrow Qj_i \wedge pos = j_{i+1}}}{\dfrac{\{Qj_i \wedge pos = j_i \wedge cond\} if (pos = j_i) then pos = j_{i+1} \{Qj_i \wedge pos = j_{i+1}\}}{(C)}}$$

Where: $(X2) = \{pos = j_i\} pos = j_{i+1} \{pos = j_{i+1}\}$, $(X3) = Qj_i \wedge pos = j_i \wedge cond \rightarrow pos = j_i$, and $(X4) = Qj_i \wedge pos = j_{i+1} \rightarrow pos = j_{i+1}$.

$$\dfrac{\{Qj_i \wedge pos = j_i + 1\} Trans(P(j_i + 1, k)) \{Qj_{i+1} \wedge pos = k + 1\}}{(D)}\ \ (Already\ proved\ for\ P(j_i + 1, k) \in S)$$

$$\dfrac{\dfrac{\dfrac{\{pos = k + 1\} pos = j_{i+1} \{pos = j_{i+1}\}}{\{Qj_{i+1} \wedge pos = k + 1\} pos = j_{i+1} \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_{i+1} \wedge pos = k + 1 \wedge pos = k + 1\} pos = j_{i+1} \{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad \dfrac{false \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}{Qj_{i+1} \wedge pos = k + 1 \wedge \neg pos = k + 1 \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}}{\dfrac{\{Qj_{i+1} \wedge pos = k + 1\}\ if\ (pos = k + 1)\ then\ pos = j_{i+1} \{Qj_{i+1} \wedge pos = j_{i+1}\}}{(E)}}$$

$$\dfrac{\dfrac{\{false\} Trans(P(k + 1, j_{i+1} - 1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}{\{Qj_{i+1} \wedge pos = j_{i+1} \wedge pos >= k \wedge pos < j_{i+1}\} Trans(P(k + 1, j_{i+1} - 1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad (G)}{\dfrac{\{Qj_{i+1} \wedge pos = j_{i+1}\} if\ (pos >= k \wedge pos < j_{i+1})\ then\ Trans(P(k + 1, j_{i+1} - 1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}{(F)}}$$

$$\dfrac{\dfrac{pos = j_{i+1} \wedge (\neg pos >= k \vee \neg pos < j_{i+1}) \rightarrow pos = j_{i+1}}{Qj_{i+1} \wedge pos = j_{i+1} \wedge (\neg pos >= k \vee \neg pos < j_{i+1}) \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}}{\dfrac{Qj_{i+1} \wedge pos = j_{i+1} \wedge \neg (pos >= k \wedge pos < j_{i+1}) \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}{(G)}}$$

$$\dfrac{\dfrac{\{false\}\ if\ (pos = j_i) pos = k + 1; \{Qj_{i+1} \wedge pos = j_{i+1}\} \quad (pos = j_i \wedge cond) \wedge \neg (pos = j_i \wedge cond) \rightarrow false}{\{Qj_i \wedge (pos = j_i \wedge cond) \wedge \neg (pos = j_i \wedge cond) \wedge \neg (pos > j_i \wedge pos <= k)\}\ if\ (pos = j_i)\ then\ pos = k + 1; \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\dfrac{\{Qj_i \wedge pos = j_i \wedge cond \wedge \neg ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}\ if\ (pos = j_i)\ then\ pos = k + 1; \{Qj_{i+1} \wedge pos = j_{i+1}\}}{B}}$$

(ii) Let us prove that $\{Qj_i \wedge pos = j_i \wedge \neg cond\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

$$\frac{\dfrac{(A)\quad (B)}{\{Qj_i \wedge pos = j_i \wedge \neg cond\} blocIf \{Qj_i \wedge pos = k+1\}} \quad \dfrac{(C)}{\{Qj_i \wedge pos = k+1\} BlocElse \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_i \wedge pos = j_i \wedge \neg cond\} Trans(P(j_i, j_{i+1}-1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}$$

$$\frac{\{false\}\ if\ (pos = j_i)\ then\ pos = j_i + 1;\ Trans(P(j_i+1, k));\ if\ (pos = k+1)\ then\ pos = j_{i+1} \{Qj_i \wedge pos = k+1\} \quad (E)}{\dfrac{\{Qj_i \wedge pos = j_i \wedge \neg cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\} (X5) \{Qj_i \wedge pos = k+1\}}{(A)}}$$

Where $(X5) = if\ (pos = j_i)\ then\ pos = j_i + 1;\ Trans(P(j_i+1, k));\ if\ (pos = k+1)\ then\ pos = j_{i+1}$

$$\frac{\dfrac{\dfrac{\neg cond \wedge cond \rightarrow false}{pos = j_i \wedge \neg cond \wedge pos = j_i \wedge cond \rightarrow false} \quad \dfrac{pos = j_i \wedge pos > j_i \rightarrow false}{pos = j_i \wedge \neg cond \wedge pos > j_i \wedge pos <= k \rightarrow false}}{((pos = j_i \wedge \neg cond \wedge pos = j_i \wedge cond) \vee (pos = j_i \wedge \neg cond \wedge pos > j_i \wedge pos <= k)) \rightarrow false}}{\dfrac{pos = j_i \wedge \neg cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k)) \rightarrow false}{(E)}}$$

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\{pos = j_i\} pos = k+1; \{pos = k+1\}}{\{Qj_i \wedge pos = j_i \wedge \neg cond\} pos = k+1; \{Qj_i \wedge pos = k+1\}}}{\{Qj_i \wedge pos = j_i \wedge \neg cond \wedge pos = j_i\} pos = k+1; \{Qj_i \wedge pos = k+1\}} \quad \dfrac{false \rightarrow Qj_i \wedge pos = k+1}{Qj_i \wedge pos = j_i \wedge \neg cond \wedge \neg pos = j_i \wedge Qj_i \wedge pos = k+1}}{\{Qj_i \wedge (pos = j_i \wedge \neg cond)\}\ if\ (pos = j_i)\ then\ pos = k+1; \{Qj_i \wedge pos = k+1\}} \quad (F)}{\{Qj_i \wedge (pos = j_i \wedge \neg cond) \wedge (\neg pos = j_i \vee \neg cond) \wedge (\neg pos > j_i \vee \neg pos < k))\}\ if\ (pos = j_i)\ then\ pos = k+1; \{Qj_i \wedge pos = k+1\}}}{\{Qj_i \wedge (pos = j_i \wedge \neg cond) \wedge \neg(pos = j_i \wedge cond) \wedge \neg(pos > j_i \wedge pos < k)\}\ if\ (pos = j_i)\ then\ pos = k+1; \{Qj_i \wedge pos = k+1\}}}{\{Qj_i \wedge pos = j_i \wedge \neg cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < k))\}\ if\ (pos = j_i)\ then\ pos = k+1; \{Qj_i \wedge pos = k+1\}}}{(B)}$$

$$\frac{pos = j_i \wedge (\neg pos > j_i \vee \neg pos < k) \wedge pos = j_i}{Qj_i \wedge (pos = j_i \wedge \neg cond) \wedge (\neg pos = j_i \vee \neg cond) \wedge (\neg pos > j_i \vee \neg pos < k) \rightarrow Qj_i \wedge pos = j_i \wedge \neg cond}$$
$$(F)$$

$$\frac{\dfrac{\dfrac{pos = k+1 \wedge pos > k \wedge pos < j_{i+1} \rightarrow pos = k+1 \quad \{Qj_i \wedge pos = k+1\} Trans(P(k+1, j_{i+1}-1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}{\{Qj_i \wedge pos = k+1 \wedge (pos > k \wedge pos < j_{i+1})\} Trans(P(k+1, j_{i+1}-1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad (H)}{\{Qj_i \wedge pos = k+1\}\ if\ (pos > k \wedge pos < j_{i+1})\ then\ Trans(P(k+1, j_{i+1}-1)) \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{(C)}$$

$$\frac{false \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}{Qj_i \wedge pos = k+1 \wedge \neg(pos > k \wedge pos < j_{i+1}) \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}$$
$$(H)$$

This enables us to state the following results:

Result 3 :
For $P(j_i, j_{i+1}-1)$ a conditional structure satisfying:
$\{Qj_i \wedge pos = j_i + 1\}\ \ Trans(P(j_i+1, k))\ \ \{Qj_{i+1} \wedge pos = k+1\}$
$\{Qj_i \wedge pos = k+1\}\ \ Trans(P(k+1, j_{i+1}-1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$
We have:
$\{Qj_i\}\ \ P(j_i, j_{i+1}-1)\ \ \{Qj_{i+1}\}\ \ \Rightarrow\ \ \{Qj_i \wedge pos = j_i\}\ \ Trans(P(j_i, j_{i+1}-1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\} \quad (T)$

According to *Result1*, the two conditions of *Result3* are *true* for simple instructions. Using structural induction we can prove that this result remains correct for nested conditional structures.

Result 4 :
For $P_n(j_i, j_{i+1}-1)$ a conditional structure having n nestings:
$\{Qj_i\}\ \ P_n(j_i, j_{i+1}-1)\ \ \{Qj_{i+1}\}\ \ \Rightarrow\ \ \{Qj_i \wedge pos = j_i\}\ \ Trans(P_n(j_i, j_{i+1}-1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

Proof of Result4 Using structural induction:

- for $n = 0$, P_n has no nestings which means that $P(j_i + 1, k)$ and $P(k + 1, j_{i+1} - 1)$ are composed of simple instructions. In this case (T) is correct.
- Assume that (T) is correct for P_n and let us prove that (T) is correct for P_{n+1} which may be written as follow:

```
if (cond) then   P_n(j_{i+1},k);   else   P_n(k+1, j_{i+1}-1);
```

According to *Result3*, and as $P_n(j_{i+1}, k)$ and $P_n(k+1, j_{i+1} - 1)$ verify (T) according to the induction assumption, P_{n+1} verifies (T) and so *Result4* is proved.

4.1.3 Case of Loop structures

A loop structure $P(j_i, j_{i+1} + 1)$ may be written as follows:

```
1 while (cond) do P(j_{i}+1,j_{i+1}-1);
```

Its transformation according to our transformation rules corresponds to:

```
1 while (pos >= j_i and pos < j_{i+1}) do {
    if((pos = j_i and cond) or (pos > j_i and pos < j_{i+1})) then{
3       if(pos = j_i) then pos = j_{i+1};
        Trans(P(j_{i}+1,j_{i+1}-1));
5       if (pos = j_{i+1}) then  pos = j_i;
    }else   pos = j_{i+1};
7 }
```

Consider that nb is the iteration number of the loop. So, $P(j_i, j_{i+1} - 1)$ corresponds to the sequential execution of $P(j_i + 1, j_{i+1} - 1)$ nb times. So, we have :

- $\{Qj_i\}P(j_i, j_{i+1} - 1)\{Qj_{i+1}\}$
- $\forall k \in \{1...nb - 1\}$, we have $\{Q^k j_i \wedge cond\}P(j_i + 1, j_{i+1} - 1)\{Q^{k+1} j_i \wedge cond\}$
- $\{Q^{nb} j_i \wedge cond\}P(j_i + 1, j_{i+1} - 1)\{Q^{nb+1} j_i \wedge \neg cond\}$
- $Qj_i = Q^1 j_i$
- $Q^{nb+1} j_i \wedge \neg cond = Qj_{i+1}$
- $\forall k \in \{1...nb\}Q^k j_i \rightarrow Q^k j_i \wedge cond$ and $Qnb + 1ji \rightarrow Qnb + 1ji \wedge \neg cond$

We should prove that: $\{Qj_i \wedge pos = j_i\}Trans(P(j_i, j_{i+1} - 1))\{Qj_{i+1} \wedge pos = j_{i+1}\}$
For that, we should prove that:

- *(i)* $\forall k \in \{1...nb - 1\}$ we have $\{Q^k j_i \wedge pos = j_i \wedge cond\}$ $blocWhile$ $\{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}$ $((pos >= j_i \wedge pos < j_{i+1}) = true)$
- *(ii)* $\{Q^{nb} j_i \wedge pos = j_i \wedge cond\}$ $blocWhile$ $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}$ $((pos >= j_i \wedge pos < j_{i+1}) = true)$
- *(iii)* $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}$ $blocWhile$ $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\}$ which results in the termination of the loop as $(pos >= j_i \wedge pos < j_{i+1}) = false$.

(i) Let us prove that $\forall k \in \{1...nb-1\}$ $\{Q^k j_i \wedge pos = j_i \wedge cond\}$ $blocWhile$ $\{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}$

$$\frac{(A) \quad (B)}{\{Q^k j_i \wedge pos = j_i \wedge cond\}(X6)\{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}}$$

Where: $(X6)$ = if $((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\{$ if $(pos = j_i)$ then $pos = j_{i+1}$; $Trans(P(j_i + 1, j_{i+1} - 1))$; if $(pos = j_{i+1})$ then $pos = j_i\}$ else $pos = j_{i+1}$

$$\dfrac{\dfrac{\{Q^k j_i \wedge pos = j_i \wedge cond\} pos = j_{i+1}; \{Q^k j_i \wedge pos = j_{i+1} \wedge cond\}}{\{Q^k j_i \wedge pos = j_i \wedge cond\} \ if \ (pos = j_i) \ then \ pos = j_{i+1}; \{Q^k j_i \wedge pos = j_{i+1} \wedge cond\}} \ (C) \quad (D)}{\{Q^k j_i \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}(X7)\{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}} \ (A)$$

Where $(X7) =$ **if** $(pos = j_i)$ **then** $pos = j_i + 1$; $Trans(P(j_i + 1, j_{i+1} - 1))$; **if** $(pos = j_{i+1})$ **then** $pos = j_i$

$$\dfrac{\dfrac{\{Q^k j_i\} P(j_i + 1, j_{i+1} - 1) \{Q^{k+1} j_i\}}{\{Q^k j_i \wedge pos = j_i\} Trans(P(j_i + 1, j_{i+1} - 1)) \{Q^{k+1} j_i \wedge pos = j_{i+1}\}} \ (Already \ proved \ for \ simple \ and \ structured \ instructions)}{\{Q^k j_i \wedge pos = j_{i+1} \wedge cond\} Trans(P(j_i + 1, j_{i+1} - 1)) \{Q^{k+1} j_i \wedge pos = j_{i+1} \wedge cond\}} \ (C)$$

$$\dfrac{\dfrac{\{pos = j_{i+1}\} pos = j_i \{pos = j_i\}}{\{Q^{k+1} j_i \wedge pos = j_{i+1} \wedge cond \wedge pos = j_{i+1}\} pos = j_i \{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}} \ (E)}{\{Q^{k+1} j_i \wedge pos = j_{i+1} \wedge cond\} \ if \ (pos = j_{i+1}) \ then \ pos = j_i \{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}} \ (D)$$

$$\dfrac{false \rightarrow Q^{k+1} j_i \wedge pos = j_i \wedge cond}{Q^{k+1} j_i \wedge pos = j_{i+1} \wedge cond \wedge \neg pos = j_{i+1} \rightarrow Q^{k+1} j_i \wedge pos = j_i \wedge cond} \ (E)$$

$$\dfrac{\{false\} \ if \ (pos = j_i) \ then \ pos = j_i + 1; Trans(P(j_i + 1, j_{i+1} - 1)); \ if \ (pos = j_{i+1}) \ then \ pos = j_i \{Q^{k+1} j_i \wedge pos = j_{i+1} \wedge cond\}}{\{Q^k j_i \wedge pos = j_i \wedge cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\} \ (X8) \ \{Q^{k+1} j_i \wedge pos = j_i \wedge cond\}} \ (B)$$

Where $(X8) =$ **if** $(pos = j_i)$ **then** $pos = j_{i+1}$; $Trans(P(j_i + 1, j_{i+1} - 1))$; **if** $(pos = j_{i+1})$ **then** $pos = j_i$;

(ii) Let us prove that $\{Q^{nb} j_i \wedge pos = j_i \wedge cond\}$ *blocWhile* $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}$

$$\dfrac{(A) \quad (B)}{\{Q_i^{nbj} \wedge pos = j_i \wedge cond\}(X9)\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}}$$

Where $(X9) =$ **if** $((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))$ **if** $(pos = j_i)$ **then** $pos = j_i + 1$; $Trans(P(j_i + 1, j_{i+1} - 1))$; **if** $(pos = j_{i+1})$ **then** $pos = j_i$ **else** $pos = j_{i+1}$

$$\dfrac{\dfrac{\{Q^{nb} j_i \wedge pos = j_i \wedge cond\} \ \ pos = j_i + 1 \ \ \{Q^{nb} j_i \wedge cond \wedge pos = j_i + 1\}}{\{Q^{nb} j_i \wedge pos = j_i \wedge cond\} \ if \ (pos = j_i) \ then \ pos = j_i + 1 \{Q^{nb} j_i \wedge cond \wedge pos = j_i + 1\}} \ (C) \quad (D)}{\{Q^{nb} j_i \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}(X10)\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}} \ (A)$$

Where $(X10) =$ **if** $(pos = j_i)$ **then** $pos = j_i + 1; Trans(P(j_i + 1, j_{i+1} - 1))$; **if** $(pos = j_{i+1})$ **then** $pos = ji$

$$\dfrac{\dfrac{\{Q^{nb} j_i \wedge cond\} \ \ P(j_i + 1, j_{i+1} - 1) \ \ \{Q^{nb+1} j_i \wedge \neg cond\}}{\{Q^{nb} j_i \wedge pos = j_i \wedge cond\} \ \ Trans(P(j_i + 1, j_{i+1} - 1)) \ \ \{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\}} \ (Proved \ for \ simple \ and \ conditional \ structures)}{\{Q^{nb} j_i \wedge cond \wedge pos = j_i + 1\} \ \ Trans(P(j_i + 1, j_{i+1} - 1)) \ \ \{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\}} \ (C)$$

$$\dfrac{\dfrac{\{pos = j_{i+1}\} \ pos = j_i \ \{pos = j_i\}}{\{Q^{nb+1} j_i \wedge pos = j_{i+1} \wedge \neg cond \wedge pos = j_{i+1}\} \ pos = j_i \ \{Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond\}} \ (E)}{\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\} \ if \ (pos = j_{i+1}) \ then \ pos = j_i \ \{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}} \ (D)$$

$$\frac{false \rightarrow Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond}{Q^{nb+1} j_i \wedge pos = j_{i+1} \wedge \neg cond \wedge \neg pos = j_{i+1} \rightarrow Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond} \quad (E)$$

$$\frac{\{false\}\ pos = j_{i+1}\ \{Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond\}}{\{Q^{nb+1} j_i \wedge pos = j_i \wedge cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ pos = j_{i+1}\ \{Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond\}} \quad (B)$$

(iii) Let us prove that $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}$ *blocWhile* $\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\}$

$$\frac{(A) \quad (B)}{\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_i\}(X11)\{Q^{nb+1} j_i \wedge \neg cond \wedge pos = j_{i+1}\}}$$

$(X11) =$ **if** $((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))$ **if** $(pos = j_i)$ **then** $pos = j_i + 1;\ Trans(P(j_i + 1, j_{i+1} - 1));$ **if** $(pos = j_{i+1})$ *then* $pos = j_i$ **else** $pos = j_{i+1}$

$$\frac{\{false\}\ if\ (pos = j_i)\ then\ pos = j_i + 1;\ Trans(P(j_i + 1, j_{i+1} - 1));\ if\ (pos = j_{i+1})\ then\ pos = j_i \{Q^{nb+1} j_i \wedge pos = j_{i+1} \wedge \neg cond\}}{\{Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}(X12)\{Q^{nb+1} j_i \wedge pos = j_{i+1} \wedge \neg cond\}} \quad (A)$$

Where $(X12) =$ **if** $(pos = j_i)$ **then** $pos = j_i + 1;\ Trans(P(j_i + 1, j_{i+1} - 1));$ **if** $(pos = j_{i+1})$ **then** $pos = ji$

$$\frac{\{pos = j_i\} pos = j_{i+1} \{pos = j_{i+1}\}}{\{Q^{nb+1} j_i \wedge pos = j_i \wedge \neg cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\} pos = j_{i+1} \{Q^{nb+1} j_i \wedge pos = j_{i+1} \wedge \neg cond\}} \quad (B)$$

As proofs made for Result 2 and 4 we can prove the following result:

Result 5 :
For any loop $P_n(j_i, j_{i+1} - 1)$ having n nestings

$$\{Qj_i\}\ \ P_n(j_i, j_{i+1} - 1)\ \ \{Qj_{i+1}\} \Rightarrow \{Qj_i \wedge pos = j_i\}\ \ Trans(P_n(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$$

Result 6 :
For any program composed of instructions belonging to S, nested conditional structures, and nested loops:

$$\{Q0\}\ \ P\ \ \{Qj_{n+1}\} \Rightarrow \{Q0 \wedge migration = false\}\ \ Trans(P)\ \ \{Qj_{n+1} \wedge pos = j_{n+1}\}$$

4.2 Step 2: $migration = true$

In this case $\{Q0\}\ \ P_{Recovery}\ \ \{Qj_m \wedge pos = j_m\}$. So, we should prove that all instructions occurring before j_m will not be executed, and all instructions occurring after j_m will be executed and will preserve the post-conditions of the original instructions. Therefore, we should prove that:

- (i) $\forall i \backslash j_{i+1} \leq pos,\ \ \{Qj_m \wedge pos = j_m\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_m \wedge pos = j_m\}$
- (ii) $\forall i \backslash j_i \leq pos \wedge j_{i+1} > j_m,\ \ \{Qj_m \wedge pos = j_m\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Q_j i + 1 \wedge pos = j_{i+1}\}$
- (iii) $\forall i \backslash j_i > pos,\ \ \{Qj_i \wedge pos = j_i\}\ \ Trans(P(j_i, j_{i+1} - 1))\ \ \{Qj_{i+1} \wedge pos = j_{i+1}\}$ (true according to Result 1)

The last item is true according to *Result1.* But for the two first ones, we should make the proof according to the structure of $P(j_i, j_{i+1} - 1)$.

4.2.1 Case of simple instructions

- (i) Let's prove that $\forall i$ such as $j_{i+1} <= pos$. $\{Qj_m \wedge pos = jm\}\ Trans(P(j_i, j_{i+1} - 1))\{Qj_m \wedge pos = j_m\}$

$$\frac{\dfrac{\dfrac{\{false\}P(j_i, j_{i+1}-1)\{Q_m \wedge pos = j_m\}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos = j_i\}\ P(j_i, j_{i+1}-1);\ pos = pos+1\{Qj_m \wedge pos = j_m\}}\ (A)}{\{Qj_m \wedge pos = j_m \wedge j_i < pos\}\ if\ (pos = j_i)\ then\ P(j_i, j_{i+1}-1);\ pos = pos+1;\ \{Qj_m \wedge pos = j_m\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos\}P(j_i, j_{i+1}-1)\{Qj_m \wedge pos = j_m\}}$$

$$\frac{(Qj_m \wedge pos = j_m \wedge j_i < j_m) \rightarrow Qj_m \wedge pos = j_m}{(Qj_m \wedge pos = j_m \wedge j_i < j_m \wedge \neg pos = j_i) \rightarrow Qj_m \wedge pos = j_m}\ (A)$$

- (ii) Let's prove that $\forall i$ such as $j_i <= pos \wedge j_{i+1} > pos$. $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

  As $j_{i+1} - j_i = 1$, we should prove that: $\forall i$ such as $j_i = j_m$. $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$ which is true according to *Result 1.*

Thus, we proved the following results.

Result 7 :
$\forall i \in 0, ..., n$ and $P(j_i, j_{i+1}-1) \in S$ such as $\{Qj_i\}\ P(j_i, j_{i+1}-1)\{Qj_{i+1}\}$

- $\forall i$ such as $j_{i+1} <= pos$. $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\{Qj_m \wedge pos = j_m\}$
- $\forall i$ such as $j_i <= pos \wedge j_{i+1} > j_m$. $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\}$
- $\forall i$ such as $j_i > pos$. $\{Qj_i \wedge pos = j_i\}\ Trans(P(j_i, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\}$

Result 8 :
For any program P composed of instructions belonging to S

$$\{Q0\}\ \ P\ \ \{Qj_{n+1}\}\ \ \Rightarrow\ \ \{Q0 \wedge migration = true\}\ \ Trans(P)\ \ \{Qj_{n+1} \wedge pos = j_{n+1}\}$$

Proof of Result 8
In order to prove that $\{Q_0 \wedge migration = true\}\ Trans(P)\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$, we should prove that $\{Q_0 \wedge migration = true\}\ P_{recovery};\ Trans(P(0, j_m - 1));\ Trans(P(j_m, j_n))\{Qj_n + 1 \wedge pos = j_{n+1}\}$. The latter is true as we have

- $\{Q_0 \wedge migration\}\ P_{recovery}\ \{Qj_m \wedge pos = j_m\}$
- $\{Qj_m \wedge pos = j_m\}\ Trans(P(0, j_m - 1))\ \{Qj_m \wedge pos = j_m\}$
- $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_m, Qj_{i+1}))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

4.2.2 Case of conditional structures

(i) Let us prove that $\forall i \backslash j_{i+1} <= pos$. $\{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_m \wedge pos = j_m\}$

$$\frac{\dfrac{(A)\quad (B)}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}blocIf\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= j_m\}}\quad \dfrac{(C)\quad (D)}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}BlocElse\{Qj_m \wedge pos = j_m\}}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_m \wedge pos = j_m\}}$$

$$\frac{\{false\}\ if\ (pos = j_i)\ then\ pos = j_i + 1;\ Trans(P(j_i + 1, k));\ if(pos = k+1)\ then\ pos = j_{i+1}\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= jm\}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}\ (X13)\ \{Qj_m \wedge pos = jm \wedge j_{i+1} <= j_m\}}\ (A)$$

$(X13)$ = **if** $(pos = j_i)$ **then** $pos = j_{i+1}$; $Trans(P(j_i + 1, k))$; **if** $(pos = k+1)$ **then** $pos = j_{i+1}$

$$\frac{\dfrac{\dfrac{(E)\quad (F)}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}\ if\ (pos = j_i)\ pos = k+1;\ \{Qj_m \wedge pos = j_m \wedge j_{i+1} <= j_m\}}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge \neg(pos = j_i \wedge cond) \wedge \neg(pos > j_i \wedge pos <= k)\}\ if\ (pos = j_i)\ pos = k+1;\ \{Qj_m \wedge pos = j_m \wedge j_{i+1} <= j_m\}}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}\ if\ (pos = j_i)\ pos = k+1;\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= j_m\}}\ (B)$$

$$\frac{\{false\}\ pos = k+1\ \{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge pos = j_i\}\ pos = k+1\ \{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos\}}\ (E)$$

$$\frac{(Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos) \rightarrow (Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos)}{(Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge \neg pos = j_i) \rightarrow (Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos)}\ (F)$$

$$\frac{\{false\}\ Trans(P(k+1, j_{i+1}-1))\ \{Qj_m \wedge pos = j_m\}}{\{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge pos >= k \wedge pos < j_{i+1}\}\ Trans(P(k+1, j_{i+1}-1))\ \{Qj_m \wedge pos = j_m\}}\ (C)$$

$$\frac{\dfrac{\dfrac{j_{i+1} <= pos \wedge (pos < k \vee pos >= j_{i+1}) \rightarrow j_{i+1} <= pos}{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge (pos < k \vee pos >= j_{i+1}) \rightarrow Qj_m \wedge pos = j_m}}{Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge (\neg pos >= k \vee \neg pos < j_{i+1}) \rightarrow Qj_m \wedge pos = j_m}}{(Qj_m \wedge pos = j_m \wedge j_{i+1} <= pos \wedge \neg(pos >= k \wedge pos < j_{i+1})) \rightarrow Qj_m \wedge pos = j_m}\ (D)$$

(ii) Let us prove that $\forall i \backslash j_i <= pos < j_{i+1}\ .\ \{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

We consider here three cases:

- $pos = j_i$ (already proved according to results 3, 4 et 5)
- $j_i < pos <= k$
- $k+1 <= pos < j_{i+1}$

Let's prove that $\forall i \backslash j_i < pos <= k\ .\ \{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

$$\frac{\dfrac{(A)\quad (B)}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge j_m <= k\} blocIf \{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad \dfrac{(C)\quad (D)}{\{Qj_{i+1} \wedge pos = j_{i+1}\} BlocElse \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}}$$

$$\frac{\dfrac{(E)\quad (F)\quad (H)}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}\ (X14)\ \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}\ (X14)\ \{Qj_{i+1} \wedge pos = j_{i+1}\}}\ (A)$$

$(X14) =$ **if** $(pos = j_i)$ **then** $pos = j_{i+1};\ Trans(P(j_i+1, k));$ **if** $(pos = k+1)$ **then** $pos = j_{i+1};$

$$\frac{(M)\quad (N)}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}\ if\ (pos = j_i)\ then\ pos = j_i + 1\ \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}}\ (E)$$

$$\frac{\dfrac{\dfrac{\dfrac{\{false\} pos = j_i + 1 \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}}{\{Qj_m \wedge pos = j_m \wedge pos <= k \wedge (false)\} pos = j_i + 1 \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}}}{\{Qj_m \wedge pos = j_m \wedge pos <= k \wedge (pos = j_i \wedge j_i < pos)\} pos = j_i + 1 \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k \wedge pos = j_i\} pos = j_i + 1 \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}}}{(M)}$$

$$\frac{\dfrac{\dfrac{\neg pos = j_i \wedge j_i < pos \rightarrow j_i < pos}{Qj_m \wedge pos = j_m \wedge pos <= k \wedge (\neg pos = j_i \wedge j_i < pos) \rightarrow Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k}}{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k \wedge \neg pos = j_i \rightarrow Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k}}{(N)}$$

$$\frac{\frac{(L)\quad (O)\quad (P)}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}Trans(P(j_i+1, j_m-1)); Trans(P(j_m, j_{m+1}-1)); Trans(P(j_{m+1}, k)); \{Qj_{i+1} \wedge pos = k+1\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}Trans(P(j_i+1, k))\{Qj_{i+1} \wedge pos = k+1\}}(F)$$

$$(L) = \{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}Trans(P(j_i+1, j_m-1))\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}$$

$$\frac{\{Qj_m \wedge pos = j_m\}Trans(P(j_m, j_{m+1}-1))\{Qj_{m+1} \wedge pos = j_{m+1}\}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k\}Trans(P(j_m, j_{m+1}-1))\{Qj_{m+1} \wedge pos = j_{m+1} \wedge j_i < pos \wedge pos <= k\}}(O)$$

$$\frac{\{Qj_{m+1}\}P(j_{m+1}, k)\{Qj_{i+1}\}}{\{Qj_{m+1} \wedge pos = j_{m+1}\}Trans(P(j_{m+1}, k))\{Qj_{i+1} \wedge pos = k+1\}}(P)$$

$$\frac{\frac{\{Qj_{i+1} \wedge pos = k+1\}pos = j_{i+1}\{Qj_{i+1} \wedge pos = j_{i+1}\}}{\{Qj_{i+1} \wedge pos = k+1 \wedge pos = k+1\}pos = j_{i+1}\{Qj_{i+1} \wedge pos = j_{i+1}\}} \quad \frac{false \rightarrow (Qj_{i+1} \wedge pos = j_{i+1}}{Qj_{i+1} \wedge pos = k+1 \wedge \neg pos = k+1 \rightarrow Qj_{i+1} \wedge pos = j_{i+1}}}{\{Qj_{i+1} \wedge pos = k+1\}if(pos = k+1)thenpos = j_{i+1}\{Qj_{i+1} \wedge pos = j_{i+1}\}}(H)$$

$$\frac{\frac{\{false\}if(pos = j_i)pos = k+1; \{Qj_{i+1} \wedge pos = j_{i+1}\} \quad (j_i < pos \wedge pos <= k) \wedge \neg(pos > j_i \wedge pos <= k) \rightarrow false}{\{Qj_m \wedge pos = j_m \wedge (j_i < pos \wedge pos <= k) \wedge \neg(pos = j_i \wedge cond) \wedge \neg(pos > j_i\ pos <= k)\}if(pos = j_i)pos = k+1; \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge j_i < pos \wedge pos <= k \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}if(pos = j_i)pos = k+1; \{Qj_{i+1} \wedge pos = j_{i+1}\}}(B)$$

$$\frac{\frac{\{false\}Trans(P(k+1, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\} \quad (pos = j_{i+1} \wedge pos < j_{i+1}) \rightarrow false}{\{Qj_{i+1} \wedge (pos = j_{i+1} \wedge pos < j_{i+1}) \wedge pos >= k\}Trans(P(k+1, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_{i+1} \wedge pos = j_{i+1} \wedge (pos >= k \wedge pos < j_{i+1})\}Trans(P(k+1, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}(C)$$

$$\frac{\frac{(Qj_{i+1} \wedge pos = j_{i+1} \wedge (pos < k \vee pos >= j_{i+1})) \rightarrow (Qj_{i+1} \wedge pos = j_{i+1})}{(Qj_{i+1} \wedge pos = j_{i+1} \wedge (\neg pos >= k \vee \neg pos < j_{i+1})) \rightarrow (Qj_{i+1} \wedge pos = j_{i+1})}}{(Qj_{i+1} \wedge pos = j_{i+1} \wedge \neg(pos >= k \wedge pos < j_{i+1})) \rightarrow (Qj_{i+1} \wedge pos = j_{i+1})}(D)$$

Let's prove that $\forall i \backslash k+1 <= pos < j_{i+1}\ .\ \{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1}-1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

$$\frac{\frac{(X)\quad (Y)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}Trans(P(j_i, j_{i+1}-1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}}{}$$

$$\frac{\frac{(A)\quad (B)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}\ blocIf\ \{Qj_m \wedge pos = j_m \wedge k <= j_m \wedge j_m < j_{i+1}\}}(X)}{}$$

$$\frac{\frac{(C)\quad (D)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}\ BlocElse\ \{Qj_{i+1} \wedge pos = j_{i+1}\}}(Y)}{}$$

$$\frac{\frac{\{false\}\ (X15)\ \{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}{\{(false) \vee (false)\}\ (X15)\ \{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}\ (X15)\ \{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}(A)$$

$(X15)$ =**if** $(pos = j_i)$ **then** $pos = j_{i+1}$; $Trans(P(j_i+1, k))$; **if** $(pos = k+1)$ **then** $pos = j_{i+1}$

$$\frac{\dfrac{\dfrac{(F)\quad (G)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge (\neg pos = j_i \vee \neg cond) \wedge (\neg pos > j_i \vee \neg pos <= k)\}(X16)\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge \neg(pos = j_i \wedge cond) \wedge \neg(pos > j_i \wedge pos <= k)\}(X16)\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos <= k))\}(X16)\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{(B)}$$

$(X16) =$ **if** $(pos = j_i)$ **then** $pos = k + 1$;

$$\frac{pos = j_m \wedge (\neg pos = j_i \vee \neg cond) \rightarrow pos = j_m \quad (\neg pos > j_i \vee \neg pos <= k) \wedge k < pos \rightarrow k < pos}{pos = j_m \wedge k < pos \wedge (\neg pos = j_i \vee \neg cond) \wedge (\neg pos > j_i \vee \neg pos <= k) \rightarrow pos = j_m \wedge k < pos}(F)$$

$$\frac{\dfrac{\dfrac{\dfrac{\{false\}pos = k+1\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\} \quad k < pos \wedge pos = j_i \rightarrow false}{\{Qj_m \wedge pos = j_m \wedge (k < pos \wedge pos = j_i) \wedge pos < j_{i+1}\}pos = k+1\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge pos = j_i\}pos = k+1\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}(H)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}(X16)\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}}}{(G)}$$

$$\frac{(Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge \neg pos = j_i) \rightarrow (Qj_m \wedge pos = j_m \wedge k < j_m \wedge j_m < j_{i+1})}{(H)}$$

$$\frac{\dfrac{\dfrac{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}Trans(P(k+1, j_m - 1)); \{Qj_m \wedge pos = j_m\} \quad (M)}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}Trans(P(k+1, j_m - 1)); Trans(P(j_m, j_{i+1} - 1)); \{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1}\}Trans(P(k+1, j_{i+1} - 1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}}{\{Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge (pos > k \wedge pos < j_{i+1})\}Trans(P(k+1, j_{i+1} - 1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}(C)$$

$$\frac{\{Qj_m \wedge pos = j_m\}Trans(P(j_m, j_{i+1} - 1))\{Qj_{i+1} \wedge pos = j_{i+1}\}}{(M)}$$

$$\frac{(false)) \rightarrow (Qj_{i+1} \wedge pos = j_{i+1}) \quad k < pos \wedge pos < j_{i+1} \wedge \neg(pos > k \wedge pos < j_{i+1}) \rightarrow false}{(Qj_m \wedge pos = j_m \wedge k < pos \wedge pos < j_{i+1} \wedge \neg(pos > k \wedge pos < j_{i+1})) \rightarrow (Qj_{i+1} \wedge pos = j_{i+1})}(D)$$

Thereby, we can prove by induction as made for result 4 the following result.

Result 9:
for $P(j_i, j_{i+1} - 1))$ a nested conditional structure, such as $\{Qj_i\}\ P(j_i, j_{i+1} - 1)\ \{Qj_{i+1}\}$, we have:

- $\forall i \backslash j_{i+1} <= pos\ .\ \{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_m \wedge pos = j_m\}$
- $\forall i \backslash j_i <= pos \wedge j_{i+1} > j_m\ .\ \{Qj_m \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$
- $\forall i \backslash j_i > pos\ .\ \{Qj_i \wedge pos = j_i\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

4.2.3 Case of loop structures

Here, we consider that the last captured state is located in the while structure. So, we should consider the number of the actual iteration h. So, we have:

$$\{Q_0\}\ P_{Recovery}\ \{Qj_m^h \wedge pos = j_m\}$$

Therefore, we should prove that:

- (i) $\forall i \backslash j_{i+1} <= pos\ .\ \{Qj_m^h \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_m^h \wedge pos = j_m\}$
- (ii) $\forall i \backslash ji <= pos \wedge j_{i+1} > j_m\ .\ \{Qj_m^h \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

- (iii) $\forall i \backslash j_i > pos$. $\{Qj_i \wedge pos = j_i\}$ $Trans(P(j_i, j_{i+1} - 1))$ $\{Qj_{i+1} \wedge pos = j_{i+1}\}$ (true, according to result 4)

(i) Let's prove that $\forall i \backslash j_{i+1} <= pos$. $\{Qj_m^h \wedge pos = j_m\}$ $Trans(P(j_i, j_{i+1} - 1))$ $\{Qj_m^h \wedge pos = j_m\}$
which corresponds to prove that:
$\{Qj_m^h \wedge pos = j_m \wedge j_{i+1} <= pos\}$ while$(pos >= j_i \wedge pos < j_{i+1})$ if$((pos = ji \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))$ if $(pos = j_i)$ then $pos = j_{i+1}$; $Trans(P(j_{i+1}, j_{i+1} - 1))$; if $(pos = j_{i+1})$ then $pos = j_i$; else $pos = j_{i+1}$ $\{Qj_m^h \wedge pos = j_m\}$
According to the pre-condition, the loop condition will never be satisfied. So, we should prove that: $Qj_m^h \wedge pos = j_m \wedge j_{i+1} <= pos \wedge Qj_m^h \wedge pos = j_m$. (which is true)

(ii) Let's prove that (ii) $\forall i \backslash ji <= pos \wedge j_{i+1} > j_m$. $\{Qj_m^h \wedge pos = j_m\}$ $Trans(P(j_i, j_{i+1} - 1))$ $\{Qj_{i+1} \wedge pos = j_{i+1}\}$
For that, we should prove that:
$\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ while $(\mathsf{pos} >= \mathsf{j_i} \wedge \mathsf{pos} < \mathsf{j_{i+1}})$ $\{\mathsf{intWhile}\}$ $\{Qj_{i+1} \wedge pos = j_{i+1}\}$
Where
$intWhile =$ if $((\mathsf{pos} = \mathsf{j_i} \wedge \mathsf{cond}) \vee (\mathsf{pos} > \mathsf{j_i} \wedge \mathsf{pos} < \mathsf{j_{i+1}}))$ {
if $(\mathsf{pos} = \mathsf{j_i})$ then $\mathsf{pos} = \mathsf{j_i} + 1$;
$\mathsf{Trans}(\mathsf{P}(\mathsf{j_i} + 1, \mathsf{j_{i+1}} - 1))$;
if $(\mathsf{pos} = \mathsf{j_{i+1}})$ then $\mathsf{pos} = \mathsf{j_i}$;
} else $\mathsf{pos} = \mathsf{j_{i+1}}$;
The loop condition is satisfied, so we can linearize the loop. Thereby, we have to prove that:

- if $h < nb$
  - (1.1) $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}$ (loop condition is satisfied $(pos >= j_i \wedge pos < j_{i+1}) = true$)
  - (1.2) $\forall k \in \{h + 1 ... nb - 1\} \{Qj_i^k \wedge pos = j_i \wedge cond\}$ $intWhile$ $\{Qj_i^{k+1} \wedge pos = j_i \wedge cond\}$ (loop condition is satisfied$(pos >= j_i \wedge pos < j_{i+1}) = true$) (already proved)
  - (1.3) $\{Qj_i^{nb} \wedge pos = j_i \wedge cond\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$ (loop condition is satisfied $(pos >= j_i \wedge pos < j_{i+1}) = true$) (already proved)
  - (1.4)$\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$ $intWhile \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_{i+1}\}$ (already proved) which results in the termination of the loop as $(pos >= j_i \wedge pos < j_{i+1}) = false$.
- If $h = nb$
  - (2) $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$ $((pos >= j_i \wedge pos < j_{i+1}) = true)$
- If $h = nb + 1$
  - (3) $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_{i+1}\}$ which results in the termination of the loop as $(pos >= j_i \wedge pos < j_{i+1}) = false$.

(1.1) Let's prove that for $h < nb$. $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}$
We consider here two cases: $pos = j_i$ and $pos > j_i$

- Case 1: $pos = j_i$

Let's prove that $\{Qj_i^h \wedge pos = j_i\}\ intWhile\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}$, in case of $h < nb$. The latter means that $Qj_i^h \rightarrow Qj_i^h \wedge cond$

$$\dfrac{\dfrac{(A)\quad (B)}{\{Qj_i^h \wedge pos = j_i \wedge cond\}\ intWhile\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{\dfrac{\dfrac{\{pos = j_i\}\ pos = j_i + 1\ \{pos = j_i + 1\}}{\{Qj_i^h \wedge pos = j_i \wedge cond\}\ if\ (pos = j_i)\ then\ pos = j_i + 1; \{Qj_i^h \wedge pos = j_i + 1 \wedge cond\}}\quad (C)\quad (D)}{\{Qj_i^h \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ X16\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(A)}$$

$X16 =$ if $(pos = j_i)$ then $pos = j_i + 1$; $Trans(P(j_i + 1, j_{i+1} - 1))$; if $(pos = j_{i+1})$ then $pos = j_i$;

$$\dfrac{\dfrac{\{Qj_i^h \wedge cond\}P(j_i + 1, j_{i+1} - 1)\{Qj_i^{h+1} \wedge cond\}}{\{Qj_i^h \wedge pos = j_i + 1 \wedge cond\}Trans(P(j_i + 1, j_{i+1} - 1))\{Qj_i^{h+1} \wedge pos = j_{i+1} \wedge cond\}}}{(C)}$$

$$\dfrac{\dfrac{\{pos = j_{i+1}\}pos = j_i\{pos = j_i\}}{\{Qj_i^{h+1} \wedge pos = j_{i+1} \wedge cond\}\ if\ (pos = j_{i+1})\ then\ pos = j_i\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(D)}$$

$$\dfrac{\dfrac{\{false\}pos = j_{i+1}\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}{\{Qj_i^h \wedge pos = j_i \wedge cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}pos = j_{i+1}\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(B)}$$

- Case 2: $pos > j_i$

Let's prove that for $h < nb$. $\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ intWhile\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}$

$$\dfrac{\dfrac{(A)\quad (B)}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ intWhile\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{\dfrac{\dfrac{(E)\quad (F)}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ if\ (pos = j_i)\ then\ pos = j_{i+1}\ \{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}}\quad (C)\quad (D)}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}(X16)\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(A)}$$

$$\dfrac{\dfrac{\{false\}\ pos = j_{i+1}\ \{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge pos = j_i\}\ pos = j_{i+1}\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}}}{(E)}$$

$$\dfrac{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge \neg pos = j_i \rightarrow Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}}{(F)}$$

$$\dfrac{\dfrac{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ Trans(P(j_i + 1, j_{i+1} - 1));\ \{Qj_i^{h+1} \wedge pos = j_{i+1}\}}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ Trans(P(j_i + 1, j_{i+1} - 1));\ \{Qj_i^{h+1} \wedge pos = j_{i+1} \wedge cond\}}}{(C)}$$

$$\dfrac{\dfrac{\{pos = j_{i+1}\}pos = j_i\{pos = j_i\}}{\{Qj_i^{h+1} \wedge pos = j_{i+1} \wedge cond\}\ if\ (pos = j_{i+1})\ then\ pos = j_i\ \{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(D}$$

$$\dfrac{\dfrac{\{false\}pos = j_{i+1}\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}{\{Qj_m^h \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}pos = j_{i+1}\{Qj_i^{h+1} \wedge pos = j_i \wedge cond\}}}{(B)}$$

(2) Let's prove that for $h = nb$. $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$

We consider here two cases: $pos = j_i$ and $pos > j_i$

- Case 1: $pos = j_i$

Let's prove that $\{Qj^n b_i \wedge pos = j_i\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$. In this case, we have $Qj^n b_i \rightarrow Qj^n b_i \wedge cond$

$$\frac{(A) \quad (B)}{\{Qj^n b_i \wedge pos = j_i\}\ intWhile\ \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}}$$

$$\frac{\dfrac{\{pos = j_i\}\ pos = j_{i+1}\ \{pos = j_{i+1}\}}{\{Qj_i^{nb} \wedge pos = j_i \wedge cond\}\ if (pos = j_i)\ then\ pos = j_{i+1}\ \{Qj_i^{nb} \wedge pos = j_i + 1 \wedge cond\}} \quad (C) \quad (D)}{\{Qj_i^{nb} \wedge pos = j_i \wedge cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}(X16)\{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}$$
$$(A)$$

$$\frac{\{Qj_i^{nb} \wedge cond\}\ P(j_i + 1, j_{i+1} - 1)\ \{Qj_i^{nb+1} \wedge \neg cond\}}{\{Qj_i^{nb} \wedge pos = j_i + 1 \wedge cond\}\ Trans(P(j_i + 1, j_{i+1} - 1))\ \{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}}$$
$$(C)$$

$$\frac{\{pos = j_{i+1}\}\ pos = j_i\ \{pos = j_i\}}{\{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}\ if\ (pos = j_{i+1})\ then\ pos = j_i\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge cond\}}$$
$$(D)$$

$$\frac{\{false\} pos = j_{i+1}\{Qj_i^{nb+1} \wedge pos = j_i \wedge cond\}}{\{Qj_i^{nb} \wedge pos = j_i \wedge cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ pos = j_{i+1}\{Qj_i^{nb+1} \wedge pos = j_i \wedge cond\}}$$
$$(B)$$

- Case 2: $pos > j_i$

Let's prove that $\{Qj^n b_i \wedge j_i < pos \wedge pos < j_{i+1}\}$ $intWhile$ $\{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}$.

$$\frac{(A) \quad (B)}{\{Qj^n b_i \wedge j_i < pos \wedge pos < j_{i+1}\}\ intWhile\ \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_i\}}$$

$$\frac{\dfrac{(E) \quad (F)}{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ if\ (pos = j_i)\ then\ pos = j_{i+1}\ \{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}} \quad (C) \quad (D)}{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ (X16)\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}$$
$$(A)$$

$$\frac{\{false\}\ pos = j_i + 1\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}}{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge pos = j_i\}\ pos = j_i + 1\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}}$$
$$(E)$$

$$\frac{}{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge \neg pos = j_i \rightarrow Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}}$$
$$(F)$$

$$\frac{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ Trans(P(j_i + 1, j_{i+1} - 1));\ \{Qj_i^{nb+1} \wedge pos = j_{i+1}\}}{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1}\}\ Trans(P(j_i + 1, j_{i+1} - 1));\ \{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}}$$
$$(C)$$

$$\frac{\{pos = j_{i+1}\}\ pos = j_i\ \{pos = j_i\}}{\{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}\ if\ (pos = j_{i+1})\ then\ pos = j_i\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}$$
$$(D)$$

$$\frac{\{false\}\ pos = j_{i+1}\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}{\{Qj_m^{nb} \wedge pos = j_m \wedge j_i < pos \wedge pos < j_{i+1} \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ pos = j_{i+1}\{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}\ (B)$$

(3) Let's prove that for $h = nb + 1$. $\{Qj_m^h \wedge pos = j_m \wedge j_i <= pos \wedge pos < j_{i+1}\}\ intWhile\ \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_{i+1}\}$

In this case, we have $j_m = j_i$ and $Qj_i^{nb+1} \rightarrow Qj_i^{nb+1} \wedge \neg cond$. So, let's prove that $\{Qj_m^h \wedge pos = j_i\}\ intWhile\ \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_{i+1}\}$

$$\frac{(A) \quad (B)}{\{Qj_m^h \wedge pos = j_i\}\ intWhile\ \{Qj_i^{nb+1} \wedge \neg cond \wedge pos = j_{i+1}\}}$$

$$\frac{\{false\}\ if\ (pos = j_i)\ then\ pos = j_{i+1};\ Trans(P(j_i + 1, j_{i+1} - 1));\ if\ (pos = j_{i+1})\ then\ pos = j_i\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}{\{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond \wedge ((pos = j_i \wedge cond) \vee (pos > j_i \vee pos < j_{i+1}))\}\ (X16)\ \{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}}\ (A)$$

$$\frac{\dfrac{\{pos = j_i\}\ pos = j_{i+1}\ \{pos = j_{i+1}\}}{\{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}\ pos = j_{i+1}\ \{Qj_i^{nb+1} \wedge pos = j_{i+1} \wedge \neg cond\}}}{\{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond \wedge \neg((pos = j_i \wedge cond) \vee (pos > j_i \wedge pos < j_{i+1}))\}\ pos = j_{i+1}\ \{Qj_i^{nb+1} \wedge pos = j_i \wedge \neg cond\}}\ B$$

Thus, we can deduce the following results

Result 10:

$P(j_i, j_{i+1} - 1))$ a nested loop and Qj_m^h the last saved state at position m of iteration h

if $\{Qj_i\}\ P(j_i, j_{i+1} - 1)\ \{Qj_{i+1}\}$ **then**

- $\forall i \backslash j_{i+1} <= pos$. $\{Qj_m^h \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\{Qj_m^h \wedge pos = j_m\}$
- $\forall i \backslash j_i <= pos \wedge j_{i+1} > j_m$. $\{Qj_m^h \wedge pos = j_m\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$
- $\forall i \backslash j_i > pos$. $\{Qj_i \wedge pos = j_i\}\ Trans(P(j_i, j_{i+1} - 1))\ \{Qj_{i+1} \wedge pos = j_{i+1}\}$

This results may be proved by induction for nested loop. Also, we can prove using structural induction the following result.

Result 11:

For any program P composed of instructions belonging to S, nested conditional structures, and nested loops:

$$\{Q_0\}\ P\ \{Qj_{n+1}\} \Rightarrow \{Q_0 \wedge migration = true\}\ Trans(P)\ \{Qj_{n+1} \wedge pos = j_{n+1}\}$$

Results 6 and 11 allows us to state that :

Result 12 :

For any program P composed of instructions belonging to S, nested conditional structures, and nested loops

$$\{Q_0\}\ P\ \{Qj_{n+1}\} \Rightarrow \{Q_0\}\ Trans(P)\ \{Qj_{n+1} \wedge pos = j_{n+1}\}$$

5 Proof of correctness of Specific Source Code transformation rules

For proving the correctness of our source code transformation rules which are specific for BPEL activities, we employ the Stirling compositional logic [3] which corresponds to an extension of the Hoare logic covering the case of parallel programs with shared variables.

The Stirling compositional logic is based on specifications of commands under the form:

$$(R_c, G_c) : \{P\}\ C\ \{Q\}$$

Where P is a precondition and Q is a postcondition whereas R_c defines the relationship which can be assumed to exist between the free variables of command C in states changed by other processes, and G_c is a commitment which must be respected by all state transformations of C.

In the next sections, we will use the Stirling compositional logic in order to prove the correctness of our transformation rules targeting *flow*, *link* and *wait* activities. In this proof, we consider that the original code may be written as follows:

$$P = [C1||C2]$$

Where
$C1 = C1(j_1, j_2 - 1); ...C1(j_i, j_{i+1} - 1); ...C1(j_n, j_{n+1} - 1)$ and
$C2 = C2(e_1, e_2 - 1); ...C2(e_i, e_{i+1} - 1); ...C2(e_m, e_{m+1} - 1)$
Where

- $C1(j_i, j_{i+1} - 1)$ and $C2(e_i, e_{i+1} - 1)$ simple instruction or complex structure or also link or wait activities having respectively $j_{i+1} - j_i$ and $e_{i+1} - e_i$ as number of instructions
- $j_0 = 0$ and $e_0 = 0$
- $(R1 \cup R2, G) : \{Q1_0 \wedge Q2_0\}\ P\ \{Q1j_{n+1} \wedge Q2e_{m+1}\}$
- $\forall i \in \{1, ..., n\}\quad \{Q1j_i\}\quad C1(j_i, j_{i+1} - 1)\quad \{Q1j_{i+1}\}$
- $\forall i \in \{1, ..., m\}\quad \{Q2e_i\}\quad C2(e_i, e_{i+1} - 1)\quad \{Q2e_{i+1}\}$

We consider also that the transformed code corresponds to:

```
P_Recvovery;
if(pos1 = j_0) then {
    NB_Branch= NB_Branch+1;
    pos1=j_1;
    pos2=e_1; }
Trans(C1)\\Trans(C2)
if(pos1 = j_n+2) then {
    NB_Branch= NB_Branch+1;
    pos1=j_n+3; }
```

Where $P_{Recovery}$ corresponds to:

```
if (migration) then {
    pos1 = j_k; pos2 = e_h;
    Q1_1 = Q1j_k; Q2_1 = Q2e_h;
    NB_Branch=nb;}
else {
    pos1 = j_0;
    pos2 = e_0;
    NB_Branch=1;}
```

and $Trans(C1)$ the transformation of $C1$ corresponds to:

```
Trans(C1(j_1 ,j_2 -1)) ;... Trans(C1(j_i ,j_{i+1}-1)) ;... Trans(C1(j_n ,j_{n+1}-1)) ;
if (pos1 = j_{n+1}) then {
    NB_Branch= NB_Branch-1;
    pos1=j_{n+2};}
```

and $Trans(C2)$ the transformation of $C2$ corresponds to:

```
Trans(C2(e_1 ,e_2 -1)) ;... Trans(C2(e_i ,e_{i+1}-1)) ;... Trans(C2(e_m ,e_{m+1}-1)) ;
if (pos2 = e_{m+1}) then {
    NB_Branch= NB_Branch-1;
    pos2=e_{m+2};}
```

$Q1j_k$ and $Q2e_h$ are the last saved post-condition before migration, j_k and e_h are the position of the next instruction to be executed after migration, and nb is the number of parallel branches saved before migration. Given these definitions, we should prove that

$$\{Q1j_1 \wedge Q2e_1\}\ Trans(P)\ \{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge NB_{Branch} = 1\}$$

.

In the same logic of the previous section, our proof will be made in two steps: (1) the code is executed without any migration: $migration = false$, and (2) the code execution corresponds to a resumption after migration: $migration = true$.

5.1 Step 1: $migration = false$

In this case, we have :

- $\{Q1j_1 \wedge Q2e_1\}\ P_{Recovery}\ \{Q1j_1 \wedge Q2e_1 \wedge pos1 = j_0 \wedge pos2 = e_0 \wedge Nb_{Branch} = 1\}$.
- $\{Q1j_1 \wedge Q2e_1 \wedge pos1 = j_0 \wedge pos2 = e_0\}$ **if** $(pos1 = j_0)$ **then** $\{\ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_1;\ pos2 = e_1;\}\ \{Q1j_1 \wedge Q2e_1 \wedge pos1 = j_1 \wedge pos2 = e_1 \wedge Nb_{Branch} = 2\}$
- $\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0\}$ **if** $(pos1 = j_{n+2})$ **then** $\{\ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_{n+3};\}\ \{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$

So, we have to prove that :
$\{Q1j_1 \wedge Q2e_1 \wedge pos1 = j_1 \wedge pos2 = e_1 \wedge Nb_{Branch} = 2\}\ Trans(C1)||Trans(C2)\ \ \{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0\}$
For that, we should prove that

$$(R1 \cup R2, G) : \{P\}[\{A1\}Trans(C1)\{A2\}]||[\{B1\}Trans(C2)\{B2\}]\{Q\}$$

where:
$P = A1 \wedge B1 = Q1j_1 \wedge pos1 = j_1 \wedge Q2e_1 \wedge pos2 = e_1 \wedge Nb_{Branch} = 2$
$Q = A2 \wedge B2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0$
$A1 = Q1j_1 \wedge pos1 = j_1 \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]$
$A2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]$
$B1 = Q2e_1 \wedge pos2 = e_1 \wedge [(Nb_{Branch} = 2 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 1 \wedge pos1 = j_{n+2})]$
$B2 = Q2e_{m+1} \wedge pos2 = e_{m+2} \wedge [(Nb_{Branch} = 1 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 0 \wedge pos1 = j_{n+2})]$
for that, we should prove:

1- $R1 \Rightarrow A2$

2- $(R1, R2 \cup G) : \{A1\}Trans(C1)\{A2\}$

3- $(R2, R1 \cup G) : \{B1\} Trans(C2) \{B2\}$

4- $R2 \Rightarrow B2$

The first and fourth points become correct if we pose $R1 = \{A1, A2, I_{12}, I_{13}, ..., I_{1n}\}$ and $R2 = \{B1, B2, I_{22}, I_{23}, ..., I_{2m}\}$.

Where, $I1i = Q1j_i \wedge pos1 = j_i \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]$ and $I2i = Q2e_i \wedge pos2 = e_i \wedge [(Nb_{Branch} = 2 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 1 \wedge pos1 = j_{n+2})]$.

The proofs of second and third points are similar. We present next the proof of the second point: $(R1, R2 \cup G) : \{A1\} Trans(C1) \{A2\}$, which corresponds to:

$(R1, R2 \cup G) : \{A1\}$
$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{A2\}$

For that we should prove that:

$-(a)-$ $(R1, R2 \cup G) : \{Q1j_1 \wedge pos1 = j_1 \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$
$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
$\{Q1j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

and

$-(b)-$ $(R1, R2 \cup G) : \{Q1j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$
if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$

For proving (a), we should prove that:
$\forall i \in \{1...n\}$, $\{Q1j_i \wedge pos1 = j_i \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$ $Trans(C1(j_i, j_{i+1} - 1))$ $\{Q1j_{i+1} \wedge pos1 = j_{i+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

As we know, $C1$ may be simple, conditional, or loop structure. For these structures, this proof is already made in the previous section. However, considering parallel programs, we should prove also that:
$\forall i \in \{1...n\}$, $R1 \Rightarrow \{Q1j_i \wedge pos1 = j_i \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$
Which is *true* considering the definition of $R1$. So, (a) is correct.
For proving (b), we should prove that:

- $R1 \Rightarrow \{Q1j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$ (Which is *true* considering the definition of $R1$).
- $(R1, R2 \cup G) : (R1, \{Q1j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})] \wedge pos1 = j_{n+1}\}$ $Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};$ $\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$ (Which is also *true*).

Thereby, we proved the following result.

Result 13 :

For any parallel program $P = C1||C2$ composed of instructions belonging to S, nested conditional structures, and nested loops

$$\{Q1j_1 \wedge Q2e_1 \wedge \neg migration\}\ Trans(P)\ \{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$$

5.2 Step 2: $migration = true$

In this case, we have:
$\{Q1j_1 \wedge Q2e_1\}\ P_{Recovery}\ \{Q1j_m1 \wedge Q2e_m2 \wedge pos1 = j_m1 \wedge pos2 = e_m2 \wedge Nb_{Branch} = m3\}$.

Where $Q1j_m1$ the last captured state of branch $C1$ and $m1$ its position, $Q2e_m2$ the last captured state of branch $C2$ and $m2$ its position, and $m3$ the number of parallel branches at these states.

We can distinguish five possible states:

1- $m1 = j_0 \wedge m2 = e_0 \wedge m3 = 1$ (equivalent to the case of $migration = false$)

2- $j_0 < m1 \leq j_{n+1} \wedge e_0 < m2 \leq j_{m+1} \wedge m3 = 2$

3- $m1 = j_{n+2} \wedge e_0 < m2 \leq j_{m+1} \wedge m3 = 1$

4- $j_0 < m1 \leq j_{n+1} \wedge m2 = j_{m+2} \wedge m3 = 1$

5- $m1 = j_{n+2} \wedge m2 = j_{m+2} \wedge m3 = 0$

For each state, we should prove that the post condition of the transformed program corresponds to : $\{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$

We begin by proving the second item. In such a case, we have:

$\{Q1j_m1 \wedge Q2e_m2 \wedge pos1 = j_m1 \wedge pos2 = e_m2 \wedge Nb_{Branch} = 2\}$ **if** $(pos1 = j_0)$ **then** $\{\ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_1;\ pos2 = e_1;\}$ $\{Q1j_m1 \wedge Q2e_m2 \wedge pos1 = j_m1 \wedge pos2 = e_m2 \wedge Nb_{Branch} = 2\}$ (since $pos1 > j_0$).

and

$\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0\}$ **if** $(pos1 = j_{n+2})$ **then** $\{\ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_{n+3};\}$ $\{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$

So, we should prove that:

We should prove that

$$(R1 \cup R2, G) : \{P\}[\{A1\}Trans(C1)\{A2\}]|||[\{B1\}Trans(C2)\{B2\}]\{Q\}$$

where:
$P = A1 \wedge B1 = Q1j_{m1} \wedge Q2e_{m2} \wedge pos1 = j_{m1} \wedge pos2 = e_{m2} \wedge Nb_{Branch} = 2$
$Q = A2 \wedge B2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0$
$A1 = Q1j_m1 \wedge pos1 = j_{m1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]$
$A2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]$
$B1 = Q2e_{m2} \wedge pos2 = e_{m2} \wedge [(Nb_{Branch} = 2 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 1 \wedge pos1 = j_{n+2})]$
$B2 = Q2e_{m+1} \wedge pos2 = e_{m+2} \wedge [(Nb_{Branch} = 1 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 0 \wedge pos1 = j_{n+2})]$
For that, we should prove the following points:

- $R1 \Rightarrow A2\ (true)$
- $(R1, R2 \cup G) : \{A1\}Trans(C1)\{A2\}$
- $(R2, R1 \cup G) : \{B1\}Trans(C2)\{B2\}$
- $R2 \Rightarrow B2\ (true)$

The proof of the second and the third points are the same. We will present the proof of the second point. which corresponds to:

$(R1, R2 \cup G) : \{A1\}$
$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$

$\{A2\}$

For that we should prove that:

$-(a)-$ $(R1, R2 \cup G) : \{Q1 j_m 1 \wedge pos1 = j_{m1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$

$\{Q1 j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge os2 = e_{m+2})]\}$

and

$-(b)-$ $(R1, R2 \cup G) : \{Q1 j_{n+1} \wedge pos1 = j_{n+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$

$\{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$

We already proved (b) in the previous section.

For proving (a), we should prove that:

- $\forall i < m1$, $\{Q1 j_{m1} \wedge pos1 = j_{m1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$ $Trans(C1(j_i, j_{i+1} - 1))$ $\{Q1 j_{m1} \wedge pos1 = j_{m1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$
- $\forall i \geq m1$, $\{Q1 j_i \wedge pos1 = j_i \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$ $Trans(C1(j_i, j_{i+1} - 1))$ $\{Q1 j_{i+1} \wedge pos1 = j_{i+1} \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

As we know, $C1$ may be simple, conditional, or loop structure. For these structures, this proof is already made in the previous section for the case of sequential programs. However, considering parallel programs, we should prove also that:

$\forall i \in \{1...n\}$, $R1 \Rightarrow \{Q1 j_i \wedge pos1 = j_i \wedge [(Nb_{Branch} = 2 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]\}$

Which is *true* considering the definition of $R1$. So, (a) is correct.

The proof of the fourth item is similar to the third item and will not be presented. Now, we present the proof of the third item corresponding to $m1 = j_{n+2} \wedge e_0 < m2 \leq j_{m+1} \wedge m3 = 1$.

In such a case, we have:

$\{Q1 j_{n+1} \wedge Q2 e_m 2 \wedge pos1 = j_{n+2} \wedge pos2 = e_m 2 \wedge Nb_{Branch} = 1\}$ **if** $(pos1 = j_0)$ **then** $\{ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_1;\ pos2 = e_1;\}$ $\{Q1 j_{n+1} \wedge Q2 e_m 2 \wedge pos1 = j_{n+2} \wedge pos2 = e_m 2 \wedge Nb_{Branch} = 1\}$ (since $pos1 > j_0$).

and

$\{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2 e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0\}$ **if** $(pos1 = j_{n+2})$ **then** $\{ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_{n+3};\}$ $\{Q1 j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2 e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$

So, we should prove that:

We should prove that

$$(R1 \cup R2, G) : \{P\}[\{A1\} Trans(C1) \{A2\}] || [\{B1\} Trans(C2) \{B2\}] \{Q\}$$

where:

$P = A1 \wedge B1 = Q1 j_{n+1} \wedge Q2 e_{m2} \wedge pos1 = j_{n+2} \wedge pos2 = e_{m2} \wedge Nb_{Branch} = 1$

$Q = A2 \wedge B2 = Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2 e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0$

$A1 = Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]$

$A2 = Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 1 \wedge pos2 = e_{m+2})]$

$B1 = Q2 e_{m2} \wedge pos2 = e_{m2} \wedge [(Nb_{Branch} = 2 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 1 \wedge pos1 = j_{n+2})]$

$B2 = Q2 e_{m+1} \wedge pos2 = e_{m+2} \wedge [(Nb_{Branch} = 1 \wedge pos1 < j_{n+2}) \vee (Nb_{Branch} = 0 \wedge pos1 = j_{n+2})]$

For that, we should prove the following points:

- $R1 \Rightarrow A2$ $(true)$
- $(R1, R2 \cup G) : \{A1\}Trans(C1)\{A2\}$
- $(R2, R1 \cup G) : \{B1\}Trans(C2)\{B2\}$
- $R2 \Rightarrow B2$ $(true)$

The proof of the third point is similar to the proof made for the previous case when $j_0 < m1 \leq j_{n+1}$. Next, we will present the proof of the second point for which $m1 = j_{n+2}$.

So, we have to prove that:

$(R1, R2 \cup G) : \{A1\}$
$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{A2\}$

For that we should prove that:

$-(a)-$ $(R1, R2 \cup G) : \{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$
$Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
$\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge os2 = e_{m+2})]\}$

and

$-(b)-$ $(R1, R2 \cup G) : \{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$
if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge [(Nb_{Branch} = 1 \wedge pos2 < e_{m+2}) \vee (Nb_{Branch} = 0 \wedge pos2 = e_{m+2})]\}$

We already proved (a) and (b) in the previous section.

Now, we will present the proof of the fifth case which corresponds to $m1 = j_{n+2} \wedge m2 = j_{m+2} \wedge m3 = 0$. In such a case:

$\{Q1j_{n+1} \wedge Q2e_{m+1} \wedge pos1 = j_{n+2} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0\}$ **if** $(pos1 = j_0)$ **then** $\{ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_1;\ pos2 = e_1;\}$ $\{Q1j_{n+1} \wedge Q2e_{m+1} \wedge pos1 = j_{n+2} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0\}$ (since $pos1 > j_0$).

and

$\{Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0\}$ **if** $(pos1 = j_{n+2})$ **then** $\{ Nb_{Branch} = Nb_{Branch} + 1;\ pos1 = j_{n+3};\}$ $\{Q1j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$

So, we should prove that:

We should prove that

$$(R1 \cup R2, G) : \{P\}[\{A1\}Trans(C1)\{A2\}]||[\{B1\}Trans(C2)\{B2\}]\{Q\}$$

where:

$P = A1 \wedge B1 = Q1j_{n+1} \wedge Q2e_{m+1} \wedge pos1 = j_{n+2} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0$
$Q = A2 \wedge B2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Q2e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 0$
$A1 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0$
$A2 = Q1j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0$
$B1 = Q2e_{m+1} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0$
$B2 = Q2e_{m+1} \wedge pos2 = e_{m+2} \wedge Nb_{Branch} = 0$

For that, we should prove the following points:

- $R1 \Rightarrow A2$ $(true)$
- $(R1, R2 \cup G) : \{A1\} Trans(C1) \{A2\}$
- $(R2, R1 \cup G) : \{B1\} Trans(C2) \{B2\}$
- $R2 \Rightarrow B2$ $(true)$

The proof of the second and third points are similar. Next, we will present the proof of the second point. So, we have to prove that:

$(R1, R2 \cup G) : \{A1\}$
$\quad Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
$\quad$ if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{A2\}$

For that we should prove that:

$-(a)-$ $(R1, R2 \cup G) : \{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0\}$
$\quad Trans(C1(j_1, j_2 - 1)); ...Trans(C1(j_i, j_{i+1} - 1)); ...Trans(C1(j_n, j_{n+1} - 1));$
$\{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0\}$

and

$-(b)-$ $(R1, R2 \cup G) : \{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0\}$
$\quad$ if $(pos1 = j_{n+1})$ then $\{Nb_{Branch} = Nb_{Branch} - 1; pos1 = j_{n+2};\}$
$\{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0\}$
(a) is correct according to result 11 and since we have $R1 \Rightarrow \{Q1 j_{n+1} \wedge pos1 = j_{n+2} \wedge Nb_{Branch} = 0\}$.
(b) is also correct since the condition is not satisfied.
Thereby, we proved the following result.

Result 14 :

For any parallel program $P = C1||C2$ composed of instructions belonging to S, nested conditional structures, and nested loops

$$\{Q1 j_1 \wedge Q2 e_1 \wedge migration\}\ Trans(P)\ \{Q1 j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2 e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$$

Based on results 13 and 14, we can deduce the following result:

Result 15 :

For any parallel program $P = C1||C2$ composed of instructions belonging to S, nested conditional structures, and nested loops

$$\{Q1 j_1 \wedge Q2 e_1\}\ Trans(P)\ \{Q1 j_{n+1} \wedge pos1 = j_{n+3} \wedge Q2 e_{m+1} \wedge pos2 = j_{m+2} \wedge Nb_{Branch} = 1\}$$

6 Conclusion

In this report, we presented the technical details of our strong mobility approach. We focus on the solution architecture as well as the employed transformation rules. In addition, we presented the proof of correctness of our transformation rules. In fact, we proved that the resulting code is semantically equivalent to the original one.

References

[1] Soumaya Marzouk, Afef Jmal Maâlej, and Mohamed Jmaiel. Aspect-oriented checkpointing approach of composed web services. In *Proceedings of the 10th international conference on Current trends in web engineering*, ICWE'10, pages 301–312, Berlin, Heidelberg, 2010. Springer-Verlag.

[2] Soumaya Marzouk, Afef Jmal Maâlej, Ismael Bouassida Rodriguez, and Mohamed Jmaiel. Periodic checkpointing for strong mobility of orchestrated web services. In *In Proceedings of the International Workshop on Self Healing Web Services (SHWS 2009) in conjunction of the 7th IEEE International Conference on Web Services (ICWS 2009)*, LosAngelos, California, USA, 2009. ACM.

[3] C. Stirling. A generalization of owicki-gries's hoare logic for a concurrent while language. *Theor. Comput. Sci.*, 58:347–359, June 1988.