# EXAMPLE PROGRAM TEMPORAL REASONING

## 1. Requirements

Develop a *temporal reasoning system which when given (1) the temporal relationship* that exists between two points `A` and `B` and (2) the temporal relationship that exists between the point `B` and a further point `C`, determines the temporal relationship that exists between the points `A` and `C`.

Note that there are only three temporal relations that can exist between any two points located along a *time line*: "before", "equals" and "after". Thus there are three relations that can link the points `A` and `B`, and three relations that can link the points `B` and `C`.

To determine the relation between `A` and `C` we can use a technique known as a *transitivity table*. This comprises a table with the possible relations between `A` and `B` represented by the horizontal axis, and the possible relations between `B` and `C` represented by the vertical axis. At the intersections are given the relations that must exist between `A` and `C` given particular coordinate relations. Thus:

| Relation between `B` and `C` | Relation between `A` and `B` | | |
|---|---|---|---|
| | Before | Equals | After |
| Before | Before | Before | Before, Equals or After |
| Equals | Before | Equals | After |
| After | Before, Equals or After | After | After |

## 2. Design

A top down analysis of the problem is given in Table 1.

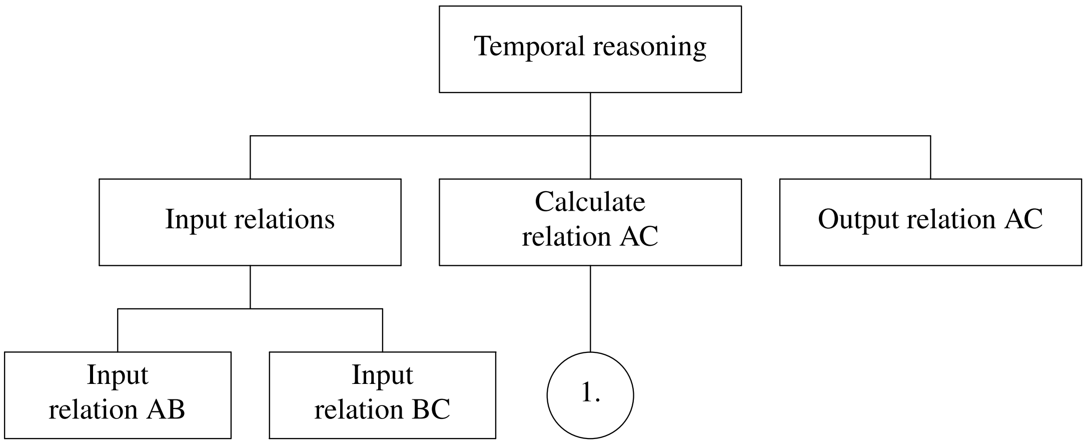


**Table 1:** *Top down analysis*

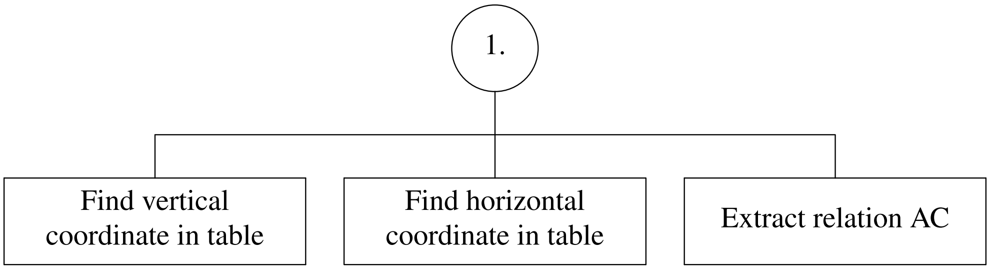


**Table 1 continued**

The operations given in Table 1 may be implemented using the following procedures:

1) `TEMPORAL_REASONING` (Top level procedure): Define enumerated type for permitted relations. Read in relation AB and relation BC. Call `RELATION_AC` function.

| Name | Description | Type | Range |
|---|---|---|---|
| RELATION_AB | Input variable | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |
| RELATION_BC | Input variable | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |

2) `RELATION_AC` (Level 2 procedure): Case statement for relation AB. Calls to `BEFORE_AB`, `AFTER_AB` and `EQUALS_AB` as appropriate.

| Name | Description | Type | Range |
|---|---|---|---|
| REL_AB | Formal parameter | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |
| REL_BC | Formal parameter | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |

3) `BEFORE_AB` (Level 3 procedure): Case statement for relation BC. Appropriate output for relation AC.

| Name | Description | Type | Range |
|---|---|---|---|
| REL_BC | Formal parameter | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |

4) `EQUALS_AB` (Level 3 procedure): Case statement for relation BC. Appropriate output for relation AC.

| Name | Description | Type | Range |
|---|---|---|---|
| REL_BC | Formal parameter | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |

5) `AFTER_AB` (Level 3 procedure): Case statement for relation BC. Appropriate output for relation AC.

| Name | Description | Type | Range |
|---|---|---|---|
| REL_BC | Formal parameter | RELATION (Enumerated type) | BEFORE, EQUALS, AFTER |

Appropriate Nassi-Shneiderman design is given in Table 2.

| |
|---|
| Input RELATION_AB |
| Input RELATION_BC |
| RELATION_AC(RELATION_AB,RELATION_BC) |

TEMPORAL_REASONING
Top level procedure

| Case statement on RELATION_AB | | |
|---|---|---|
| before | equals | after |
| BEFORE_AB (REL_BC) | EQUALS_AB (REL_BC) | AFTER_AB (REL_BC) |

RELATION_AC
Level 2 procedure

| Case statement on RELATION_BC | | |
|---|---|---|
| before | equals | after |
| output ‘before’ | output ‘before’ | output ‘before’ equals or after’ |

BEFORE_AB
Level 3 procedure

| Case statement on RELATION_BC | | |
|---|---|---|
| before | equals | after |
| output ‘before’ | output ‘equals’ | output ‘after’ |

EQUALS_AB
Level 3 procedure

| Case statement on RELATION_BC | | |
|---|---|---|
| before | equals | after |
| output ‘before’ equals or after’ | output ‘after’ | output ‘after’ |

AFTER_AB
Level 3 procedure

**Table 2**: *Nassi-Shneiderman chart*

A flow chart indicating the overall flow of control through the proposed software is given in Table 3.

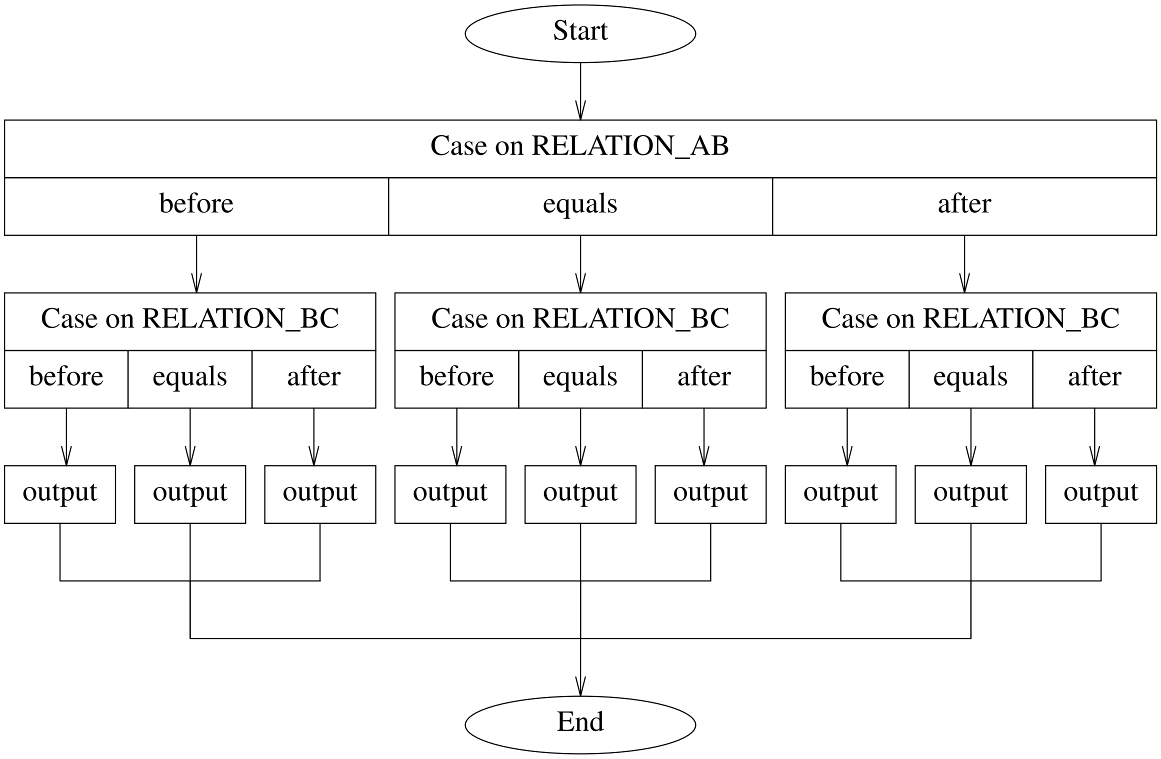


**Table 3**: *Flow chart*

### 3. Implementation

The encoding of the above design is given in Table 4.

```
-- TEMPORAL REASONING
-- 1 October 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO;
use TEXT_IO;
procedure TEMPORAL_REASONING is
    type RELATION is (before, equals, after);
    package RELATION_INOUT is new ENUMERATION_IO(RELATION);
    use RELATION_INOUT;
    RELATION_AB, RELATION_BC : RELATION;
```

**Table 4:** *Implementation*

```
----------------------------------------------------------------------
-- Identify horizontal element in transitivity table
procedure RELATION_AC(REL_AB, REL_BC : RELATION) is
    ------------------------------------------------------------------
    -- BEFORE AB
    procedure BEFORE_AB(REL_BC : RELATION) is
    begin
        case REL_BC is
            when before | equals =>
                PUT_LINE("A before C");
            when after =>
                PUT_LINE("A before, equals or after C");
        end case;
    end BEFORE_AB;
    ------------------------------------------------------------------
    -- EQUALS AB
    procedure EQUALS_AB(REL_BC : RELATION) is
    begin
        case REL_BC is
            when before =>
                PUT_LINE("A before C");
            when equals =>
                PUT_LINE("A equals C");
            when after =>
                PUT_LINE("A after C");
        end case;
    end EQUALS_AB;
    ------------------------------------------------------------------
    -- AFTER AB
    procedure AFTER_AB(REL_BC : RELATION) is
    begin
        case REL_BC is
            when before =>
                PUT_LINE("A before, equals or after C");
            when others =>
                PUT_LINE("A after C");
        end case;
    end AFTER_AB;
    ------------------------------------------------------------------

begin
    case RELATION_AB is
        when before =>
            BEFORE_AB(REL_BC);
        when equals =>
            EQUALS_AB(REL_BC);
        when after =>
            AFTER_AB(REL_BC);
    end case;
end RELATION_AC;
----------------------------------------------------------------------
```

**Table 4 continued**

```
begin
-- Input data
    PUT_LINE("Input relation A and B (before, equals or after)");
    GET(RELATION_AB);
    PUT_LINE("Input relation B and C (before, equals or after)");
    GET(RELATION_BC);
-- Determine relation linking points A and C
    RELATION_AC(RELATION_AB,RELATION_BC);
    NEW_LINE;
end TEMPORAL_REASONING;
```

**Table 4 continued**

### 4. Testing

The simplest way to test the above code is to test every combination of input. There are only nine of these as shown in Table 5. We should also test an input for each relation which is not included in the enumeration. These tests will ensure that all paths through the system are followed.

| TEST CASE NATURAL_NUMBER | EXPECTED RESULT OUTPUT |
|---|---|
| before | before |
| before | equals |
| before | after |
| equals | before |
| equals | equals |
| equals | after |
| after | before |
| after | equals |
| after | after |
| between | * |
| before | between |

**Table 5:** Test Cases

Results from running them above tests are presented in Table 6.

```
kuban-259 $ temporal_reasoning
Input relation A and B (before, equals or after)
before
Input relation B and C (before, equals or after)
before
A before C

kuban-260 $ temporal_reasoning
Input relation A and B (before, equals or after)
before
Input relation B and C (before, equals or after)
equals
A before C

kuban-261 $ temporal_reasoning
Input relation A and B (before, equals or after)
before
Input relation B and C (before, equals or after)
after
A before, equals or after C

kuban-262 $ temporal_reasoning
Input relation A and B (before, equals or after)
equals
Input relation B and C (before, equals or after)
before
A before C

kuban-263 $ temporal_reasoning
Input relation A and B (before, equals or after)
equals
Input relation B and C (before, equals or after)
equals
A equals C

kuban-264 $ temporal_reasoning
Input relation A and B (before, equals or after)
equals
Input relation B and C (before, equals or after)
after
A after C

kuban-265 $ temporal_reasoning
Input relation A and B (before, equals or after)
after
Input relation B and C (before, equals or after)
before
A before, equals or after C
```

**Table 6:** Testing results

```
kuban-266 $ temporal_reasoning
Input relation A and B (before, equals or after)
after
Input relation B and C (before, equals or after)
equals
A after C

kuban-267 $ temporal_reasoning
Input relation A and B (before, equals or after)
after
Input relation B and C (before, equals or after)
after
A after C

kuban-268 $ temporal_reasoning
Input relation A and B (before, equals or after)
between
Ada-runtime: Exception DATA_ERROR propagated out of main.

kuban-269 $ temporal_reasoning
Input relation A and B (before, equals or after)
before
Input relation B and C (before, equals or after)
between
Ada-runtime: Exception DATA_ERROR propagated out of main.
```

**Table 6 continued**