Chapter 17

Indeterminacy and the Endgame

17.1 Logical Analysis of Handover Failures

A *Predicate-Action Diagram* (PAD) is a form of 'state machine' definable in logic, which illustrates intuitively and clearly how a finite-state process defined in TLA works. The states of a PAD are defined by state predicates, and the transitions (arrows) between states by actions. A specification of a process conforming to this PAD would entail that whatever action (from the defined actions) was taken, it is guaranteed that one ends up in a state satisfying one (a precise one) of the defined state predicates.

17.1.1 Defining the PAD Rigorously

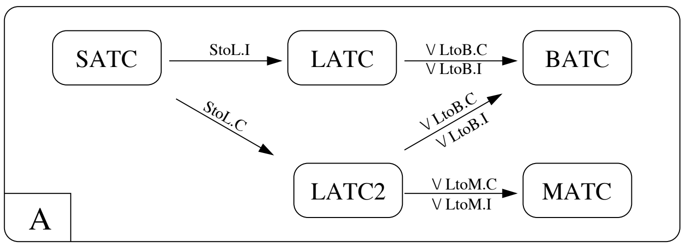

Figure 17.1: Predicate Action Diagram for the ATC Handoffs

The PAD that we wish to define is illustrated in Figure 17.1. Its definition in terms of the TLA+ modules in Section 16 is given in the TLA+ module in Figure 17.2. The intuitive meaning of the PAD is that SATC could hand over correctly or incorrectly to LATC. If incorrect, then no matter what LATC did, the flight continues (incorrectly) to Brussels. We do not consider the possibility that an error from SATC was cancelled out by another error from LATC, because we are not interested in odd cases in which everything turns out right in the end. We are interested first in the actual case, in which everything actually did turn out wrong, and second in the 'usual', correct, handoff sequence. If the handoff from SATC to LATC was correct, there are two possiblities shown: either the correct handoff from LATC, which would have been to Maastricht ATC, or an incorrent handoff to BATC. It should be emphasised that the graphical PAD is a helpful illustration, but that Figure 17.2 is the precise specification with which we work in the analysis.

17.1.2 Detecting possible sources of error

We reason from the TLA+ specifications to detect possible procedural errors. By way of example, let us assume that an error occurred on the communication between SATC and LATC. We aim to find out what this error could be. The correct communication is defined by the predicate

$$\neg StoL.C$$

By evaluating this expression using its specification given in $PAD.ATCcomm\_history$ and $ATCcomm\_history$, we obtain

$$\neg StoL.C \;\triangleq\; \neg StoL_{correct}$$

and

$$\begin{aligned}\neg StoL_{correct} \;\triangleq\;& \lor \neg StoL\\ & \lor \neg \text{UNCHANGED } persistent\_data\end{aligned}$$

(Notice that $(StoL \land \neg \text{UNCHANGED } persistent\_data)$ is equivalent to $StoL_{incorrect}$, since $\neg\text{UNCHANGED } persistent\_data$ implies $persistent\_data' \neq persistent\_data$.)

We analyse $\neg StoL$ further:

$$\begin{aligned}\neg StoL \;\triangleq\;& \lor \neg SATC.Upload(fid)\\ & \lor \neg LATC.Download\end{aligned}$$

Now we insert the specifications of $SATC.Upload(fid)$ and $LATC.Download$:

module *PAD.ATCcomm_history*

extends *Naturals*, *Sequences*, *ATCcomm*, *ATCcomm_history*

$$
\begin{array}{lll}
\mathit{SATC} & \triangleq & \exists\,!\mathit{msg} \in \mathit{SATCdata} : \begin{array}{l}\land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fid}] = \text{“}\mathit{NW052}\text{”}\\ \land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fdata}] = \text{“}\mathit{FRA}\text{”}\end{array}\\
\mathit{LATC} & \triangleq & \exists\,!\mathit{msg} \in \mathit{SATCdata} : \begin{array}{l}\land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fid}] = \text{“}\mathit{NW052}\text{”}\\ \land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fdata}] = \text{“}\mathit{FRA}\text{”}\end{array}\\
\mathit{LATC2} & \triangleq & \exists\,!\mathit{msg} \in \mathit{SATCdata} : \begin{array}{l}\land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fid}] = \text{“}\mathit{NW052}\text{”}\\ \land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fdata}] \neq \text{“}\mathit{FRA}\text{”}\end{array}\\
\mathit{BATC} & \triangleq & \exists\,!\mathit{msg} \in \mathit{SATCdata} : \begin{array}{l}\land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fid}] = \text{“}\mathit{NW052}\text{”}\\ \land\ \mathit{msg}[\mathit{persistent\_data}][\mathit{fdata}] = \text{“}\mathit{BRU}\text{”}\end{array}\\
\mathit{MATC} & \triangleq & \exists\,!\mathit{msg} \in \mathit{SATCdata} : \mathit{msg}[\mathit{persistent\_data}][\mathit{fid}] = \text{“}\mathit{NW052}\text{”}\\
\mathit{StoL.C} & \triangleq & \mathit{StoL}_{\mathit{correct}}\\
\mathit{StoL.I} & \triangleq & \mathit{StoL}_{\mathit{incorrect}}\\
\mathit{LtoB.C} & \triangleq & \mathit{LtoB}_{\mathit{correct}}\\
\mathit{LtoB.I} & \triangleq & \mathit{LtoB}_{\mathit{incorrect}}\\
\mathit{LtoM.C} & \triangleq & \mathit{LtoM}_{\mathit{correct}}\\
\mathit{LtoM.I} & \triangleq & \mathit{LtoM}_{\mathit{incorrect}}
\end{array}
$$

$$
\begin{array}{lll}
\mathit{SomeAction} & \triangleq & \begin{array}{l}\lor\ \mathit{StoL.C}\\ \lor\ \mathit{StoL.I}\\ \lor\ \mathit{LtoB.C}\\ \lor\ \mathit{LtoB.I}\\ \lor\ \mathit{LtoM.C}\\ \lor\ \mathit{LtoM.I}\end{array}\\
\mathit{Safety} & \triangleq & \begin{array}{l}\land\ \mathit{Init}\\ \land\ \Box[\mathit{SomeAction}]_{\mathit{persistent\_data}}\end{array}
\end{array}
$$

Figure 17.2: The Communications History Module in TLA+

$$
\begin{aligned}
\neg StoL \triangleq\ & \vee \neg \left(\exists\, !msg \in SATCdata : \begin{array}{l} \wedge\ msg[1] = fid \\ \wedge\ SATCdata' = SATCdata \setminus \{msg\} \\ \wedge\ SATC.Send(msg) \end{array} \right) \\
& \vee \neg \left(\begin{array}{l} \wedge\ |storage| < StorageSize \\ \wedge\ LATCdata' = LATCdata \cup \{Head(channel)\} \\ \wedge\ LATCtx.Receive \end{array} \right) \\
\triangleq\ & \vee\ \forall\, !msg \in SATCdata : \begin{array}{l} \vee\ msg[1] \neq fid \\ \vee\ SATCdata' \neq SATCdata \setminus \{msg\} \\ \vee\ \neg SATC.Send(msg) \end{array} \\
& \vee \left(\begin{array}{l} \vee\ |LATCdata| \geq StorageSize \\ \vee\ LATCdata' \neq LATCdata \cup \{Head(channel)\} \\ \vee\ \neg LATCtx.Receive \end{array} \right)
\end{aligned}
$$

Finally we replace $SATC.Send(msg)$ and $LATC.Receive$ by their specifications:

$$
\begin{aligned}
\neg StoL \triangleq\ & \vee\ \forall\, !msg \in SATCdata : \begin{array}{l} \vee\ msg[1] \neq fid \\ \vee\ SATCdata' \neq SATCdata \setminus \{msg\} \\ \vee\ msg \notin Messages \\ \vee\ Len(channel) \geq ChannelSize \\ \vee\ channel \neq channel \circ \langle msg \rangle \end{array} \\
& \vee\ |LATCdata| \geq StorageSize \\
& \vee\ LATCdata' \neq LATCdata \cup \{Head(channel)\} \\
& \vee\ channel = \langle\, \rangle \\
& \vee\ channel' \neq Tail(channel)
\end{aligned}
$$

$\neg StoL.C$ subsumes many actions, of course, that have nothing to do with any attempted handover from SATC to LATC. But assuming that a handover was attempted, but failed, we can use the table to tell us what the possible actions could have been. Assuming that error messages had been announced as expected (marked as $(*)$), we can identify four possible errors:

1. *loss/modification of SATCdata*
2. *erroneous data sent (a) or data lost (b)*
3. *channel data entered erroneously to LATCdata*
4. *erroneous transmission over channel*

Formula	Meaning
$msg[1] \neq fid$	no data for FID present (*)
$SATCdata' \neq SATCdata \setminus \{msg\}$	**loss/modification of SATCdata**
$msg \notin Messages$	invalid/corrupted data
$Len(channel) \geq ChannelSize$	channel overflow (*)
$channel \neq channel \circ \langle msg \rangle$	**erroneous data sent or data lost**
$\lvert LATCdata \rvert \geq StorageSize$	LATCdata overflow (*)
$LATCdata' \neq LATCdata \cup \{Head(channel)\}$	**channel data entered erroneously to LATCdata**
$channel = \langle\,\rangle$	no data transmitted (*)
$channel' \neq Tail(channel)$	**erroneous transmission over channel**

(*): Error message expected

Figure 17.3: The Meaning of Formulas in the Communication Analysis

Let us now assume a reliable (lossless, translation-error-free) channel (while keeping in mind that this assumption could indeed be false). Possible mistakes can therefore only happen in the ATCCs. So item 4 and alternative (b) of item 2 may be ignored. The possible errors left are then

- The data for Flight NW052 was in SATC database, but was either modified by mistake or completely lost (see error 1). Since we know from the story that data *was* sent to LATC, we can abandon the latter supposition (otherwise we have found a really serious problem with the SATC's computer system!...)

- The data sent by SATC was erroneous, either because it was already false in the database (see above) or because an error was made when entering the data during the transfer process to LATC (error 2(a)).

- Correct data reached LATC, but it was changed mistakenly when it was entered to LATC database (error 3).

Any of these mistakes would result in a state described by predicate $LATC2$. These represent the logically possible ways (assuming a reliable channel) that a failure in *StoL* could have occurred, provided that the ATC handovers satisfy the (very general, therefore probably applicable) TLA+ specifications in Section 16. The other handover or handling failures can be analysed similarly, and these analyses not only result in the PAD in Figure 17.1, but tell us very precisely what the possible errors could have been.

17.1.3 Putting all the Pieces Together

We can perform the analysis presented in Section 17.1.2 for the transmission of data from SATC to LATC (*StoL*) similarly for LtoB (and of course *LtoM* to cover the expected world), too. The result is a completely rigorous logical analysis of the failure possibilities in the ATC handover sequence. We emphasise that nothing has been left out of the logic – full specifications have been given and the definitions of the actions manipulated logically to enumerate the possible ways in which a handover can fail.

A summary of possible errors in the $StoL - and - LtoB$-world is given by the PAD in Figure 17.6. This may lead us to further field investigation to narrow the choice of possible errors down even further, maybe to the point at which we could definitively state which error occurred; or it may lead us to recommendations which would preclude – or at least considerable reduce the chance of – any of these possible errors occurring in the future. (*Recommendations*, along with *Findings*, and statements of *Probable Cause* and *Contributing Factors*, are a 'standard output' of an accident investigation.) A certain amount of indeterminacy may be expected – we should not always expect to be able to narrow down the choice of possible procedural errors to one definite error. In such a situation, the PAD expresses precisely what is known about the situation, showing the possible alternatives and their consequences in precise form. This may often be as good as we can get. In this case, we can go no further – we have no more information that enables us to determine which of the possible erroneous behaviors actually occurred in the incident.

17.2 The Endgame

The analysis appears to be nearly complete. There are two major points to consider. First, how we may constrain the hypotheses to analyse a possible or probable scenario, and how indeterminacy may be represented. We shall consider these points in Sections 17.2.1 and 17.2.2.

Second, how do we know that the analysis is correct? So far, we have been proceeding informally. Do we have reason to believe our analysis is correct according to the general principles of WBA? In fact, we don't. From our informal reasoning so far, we construct the 'final' WB-Graphs in Figures 17.4 (pictorial) and 17.2.1. However, in the course of constructing the formal proof according to the formal rules of EL in Part part:proof, we shall find that various additional facts and hypotheses need to be added in order to make the formal proof correct. Thus our intuition only takes us so far. Constructing the formal proof that an analysis suffices is generally necessary to find the gaps in the informal reasoning. But back now to finishing off the informal WBA.

17.2.1 Constraining the Hypotheses

We do not in fact know which of the state predicates given in the module in Figure 17.2 is true. However, the source texts contain such assertions or assumptions: for example, that $\{3a\}$ holds. We cannot say from the sources whether $\langle 3b \rangle$ or $\langle 3c \rangle$ or both are false. It is also not necessarily advisable to take assumptions made in sources for true assertions. They may just be unintentional, unanalysed, assumptions which an analyst should expose to explicit consideration.

To complete discussion of the example, we investigate just one of the possibilities. Suppose we assume $\langle 3c \rangle$[1]. We then can conclude that $\langle 3b \rangle$ is false, though either the data or the flight plan (or both) transmitted to aircraft and BATC by LATC was erroneous. Consequently, we would then need to determine whether

[221] London received false data from SATC
or
[222] data was falsified during processing in LATC.

Although both possibilities are given in [dWL95], we learn from [Lad95a] that [221] was asserted. So let us assume this also.

We need to analyse the relation "$\langle 4 \rangle \hookrightarrow \langle 3 \rangle$" to replace it by a causal explanation. To illustrate this last step, let us take it here as true that [221]. Since we know about the procedures performed by ATC instances now, we should include four more nodes in the graph:

$\langle 31 \rangle$ SATC procedures are implemented correctly
$\langle 32 \rangle$ FI is correct at SATC
$\langle 33 \rangle$ AC position and direction of flight is consistent with FI at SATC
[*StoL*] FI from SATC imply next ATC is LATC

This leads to the graph, totally based on causality, without any further $\hookrightarrow$ arrows, that appears in Figure 17.4. The textual form appears in Figure 17.2.1.

17.2.2 Indeterminacy

Is that all we can do? As far as the facts we can directly derive from our textual sources are concerned, it seems so, since there aren't any more facts in the source text! Further analysis will be based mainly on assumptions, although it is not absolutely speculation. There are two ways in which we can do more. One is to write the formal proof, which we consider in Section 17.3.

The other is as follows. As we discovered in our analysis, under the assumption that wrong data were sent by SATC, this sending of incorrect data would be one

[1]We also could assume $\langle 3b \rangle$ at this stage. This is just to define a starting point for further investigations.

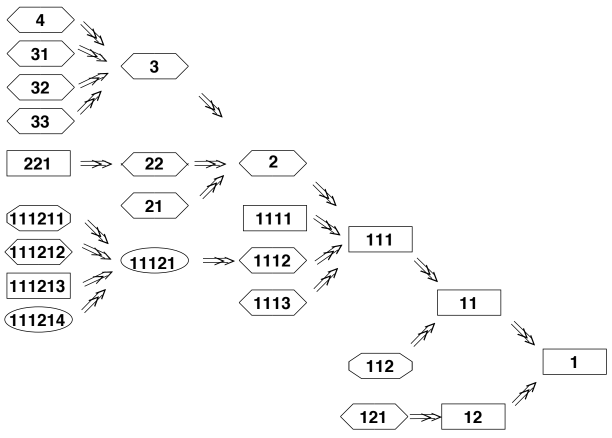

Figure 17.4: The Supposedly 'Final' WB-Graph

of the fundamental causes of the incident. We need a method then of handling assumptions in the analysis, since finding why erroneous data was entered to certain computers might supply hints on what to change to prevent such incidents in future;

To argue with assumptions, we introduce more notation to differentiate assumed nodes from those based on facts. We denote states by "⟪**A**⟫", events by "⟦**A**⟧", processes by "⦃**A**⦄" and non-events by "⦅**A**⦆".

Let us assume a solution, then, to the earlier non-determinacy in Section 16, namely that SATC sent the wrong data to LATCC. Under this assumption, why did SATC send wrong flight data?
We assume that

⟪**2211**⟫ correct information was provided to SATC,
⟦**2212**⟧ information is recognized by air traffic controller
and
⟪**2213**⟫ communication of data between SATC and its adjacent ATC
is error-free,

are true – this need not necessarily be so, it is just *one* way to base an argument. To argue e.g. that

⟦**2214**⟧ ATC controller entered erroneous data into computer

we can use different reasoning, depending on what else we assume. Let *Controller* stand for the Shannon controller. Saying that

⟪**2215**⟫ Controller correctly recognized information displayed
and
⟦**2216**⟧ Controller made typing error

```
[1] /* AC lands at Brussels RWY 25 */
  [-.1] /* Crew (CRW) realizes they are landing at the wrong airport */
  [-.2]/* CRW opts to continue landing */

  [1.1] /\[-.1] /* CRW gets visual contact to Brussels airport */
        /\{-.2} /* CRW notices that Brussels' airport layout is different
                   from Frankfurt's */

    [1.1.1] /\[-.1] /* AC breaks out under clouds */
            /\<-.2> /* AC near Brussels Airport */
            /\<-.3> /* CRW procedures */
            /\ <2>  /* AC in BATC area */

     <1.1.1.2> (-.1) /* CRW did not realize that they were on wrong course,
                        UNTIL:[111] */

       (1.1.1.2.1) /\{-.1} /* CRW addresses BATC controller as
                              ''Frankfurt'' several times,  */
                   /\<-.2> /* ILS has different frequency for Frankfurt. */
                   /\[-.3] /* CRW asks for the Bruno VOR's frequency. */
                   /\(-.4) /* Brussels did not question the addressing error
                              although it happened more than once */
    <2> /\<-.1> /* LATC procedures */
        /\<-.2> /* LATC uses false flight data for NW052*/
        /\ <3>  /* AC in LATC area */

     <2.2> [-.1] /* London received false data from SATC */

     <3> /\<-.1> /* SATC handoff procedures under this flightplan
                    are to LATC */
         /\<-.2> /* FI is correct at SATC */
         /\<-.3> /* AC-PD is sufficiently consistent with FI at SATC */
         /\ <4>  /* AC in SATC area */

  [1.2] <-.1> /* CRW has safety reasons for continuing landing */
```

Figure 17.5: The Supposedly 'Final' Textual WB-Graph

would lead to ⟦**2214**⟧ as well as would

> ⟪**2217**⟫ Controller incorrectly recognized displayed information and
> ⟦**2218**⟧ Controller typed recognized information correctly.

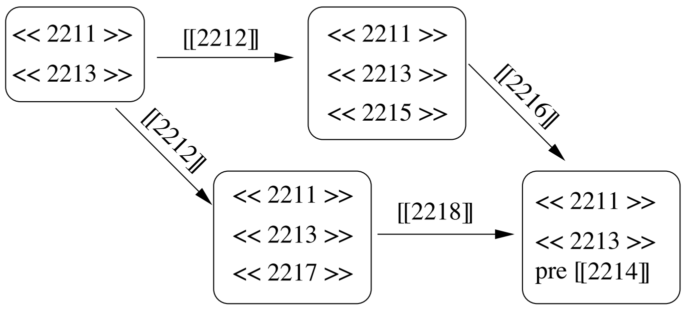

Figure 17.6: Predicate Action Diagram displaying possible worlds

This reasoning is not deterministic: either of these two circumstances could have caused ⟦**2214**⟧. There are alternatives here, also, not distinguished by facts in the history. We can use a form of PAD here, too, to distinguish the possibilities, but this time of actual states and actual actions that occurred, in Figure 17.6

17.2.3 Adapting PADs

We base this form of PAD not on the pure language of TLA, but on its extension EL, which we introduce in Part IV. This also means that the interpretation changes a little, although formal use is similar.

Since we speak of procedures and obligations for computer and physical systems as well as organisational procedures, to say that a system defined as a state machine is specified as having state S_2 follow state S_1 when action A is performed is to say that the performance of action A in state S_1, along with the specification, explains why state S_2 happened. In TLA, an implication is provable iff it is a logical truth. So in our formulation, this implication must be interpreted as logical implication:

$$\frac{\vdash_{TLA} A \Rightarrow B}{A \succ B} \tag{17.1}$$

Using this now to interpret the PAD, it means that

$$\mathit{Spec} \wedge S_1 \wedge A \succ \Diamond S_2 \tag{17.2}$$

(actually, not just $\Diamond S_2$, that S_2 will happen *eventually*, but in fact S_2': that the values of the state variables at the end of the action A will be such as to make the state predicate S_2 true. But this TLA notation is not needed here to explain causality for machines. See also Section 16.1 for a reminder of the meaning of $\Diamond$).

That $\Diamond S_2$ follows tense-logically from $\mathit{Spec} \wedge S_1 \wedge A$ thus means that, if $\mathit{Spec} \wedge S_1 \wedge A$ is in fact true, that is if *Spec* is fulfilled and A is true and the machine is in the state described by S_1, then S_2 will inevitably occur. It is an explanation therefore for the occurence of S_2 – in fact a causal explanation because physical systems, digital or otherwise, are designed to be causal devices – that $\mathit{Spec} \wedge S_1 \wedge A$. And more – this latter is a sufficient causal factor for the occurrence of S_2. Therefore because

$$\mathit{Spec} \wedge S_1 \wedge A \succ S_2 \tag{17.3}$$

Thus our adaptation of PADs is formally similar to their suggested use by Lamport for specification, but we interpret the transitions as causally-explanatory links via this sufficient-causal-factor explanation. In a PAD, of course, the role of the specification is implicit: the PAD *is* the specification. Its meaning is that, if the specification is to be fulfilled, these transitions must happen. Whether one interprets its meta-meaning as causal or rather deontic need not concern us further here. This somewhat wider interpretation of PADs will enable them to serve rigorously also for combinations of human-operator action in procedures, or in standard behavior. It is stretching the use of the language to speak of a human operator performing according to specification – we much prefer to speak of a human as performing according to procedure, because she *ought to*: deontics rather than physical inevitability. After all, unlike machines, a human operator can behave contrary to procedure without anything physically being wrong, or anything being wrong with the design. But formally we may interpret the PAD using deontics, thereby allowing it to be used in the same manner for machines or for human processes – an extension of its original application obtained just by thinking a little more broadly.

17.3 The Really Final WB-Graph

The analysis has not yet been checked for correctness – that is, that each causal assertion is true, and that the Causal Sufficiency Condition is satisfied for each set of causal factors of a node. For this, we must perform a formal proof in EL. When we do this in Part IV, we shall find that our analysis is not yet complete and we need a few more nodes.

This discovery of missing nodes through enforced rigor is a general phenomenon known to those in system verification, and others who concern themselves with precise formal proving. It is common in the system-verification world, for example, to attempt mechanically to check the journal proof of, say, the correctness of a concurrent algorithm, and to find gaps which need to be fixed or filled in. See for example, [Rv93] for commentary on errors found during the formal verification in the published hand proof of a Byzantine-fault-tolerant algorithm for synchronising clocks in the replicated computers of a digital flight control system. Similarly, [SM95] describes the formal specification and verification of the architecture of the Collins AAMP5 avionics processor, the successor of the popular AAMP2 processor. All the instructions for the chip were specified, and a proportion of them verified. One deliberately-seeded error was found, and one error was found which noone previously knew about.

We are not immune to this phenomenon, and we don't know anyone who is. The formal proofs are really necessary if one needs to ensure that one has made no mistakes. The formal proof in the logic EL of our causal analysis of the example may be found in Chapter 22. We found during the proof that we needed to add four extra nodes to the WB-Graph. One discovers such omissions when one realises during a proof that one cannot prove the sentence one is trying to prove. Often, this sentence is a conditional, and upon inquiry into the reason for this, it is common to realise that a crucial assertion is missing from the antecedent of the conditional. One determines what it is, adds it in, and finishes the proof (one may need to do this more than once!). Oftentimes, when the sentence that one realises one cannot prove is *not* a conditional, one realises that in fact some hypotheses are missing and that it should be a conditional (with these hypotheses). One makes it so, performs the proof and continues.

The final pictorial version of the graph appears in Figure 17.7 and the final textual form in Figure 17.8. We passed the text in Figure 17.8 through the *wb2dot* tool [Höh98], which converts the textual form of a WB-Graph into input for the graph-drawing program *dot* [Nor] and produces Figure 17.9.

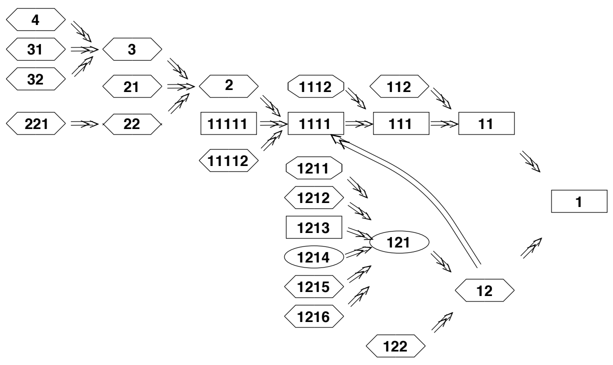

Figure 17.7: The Final WB-Graph

```
[1] /* AC lands at Brussels RWY 25 */
  /\[-.1] /* CRW opts to continue landing */
  /\<-.2> /* AC near Brussels Airport */

  [1.1] /\[-.1] /* CRW realizes they are landing at the wrong airport */
        /\<-.2> /* CRW has safety reasons for continuing landing */
        /\<-.3> /* Standard Operating Procedures */

    [1.1.1] /\[-.1] /* CRW gets visual contact to Brussels airport */
            /\{-.2} /* CRW notices that Brussels' airport layout is different
                       from Frankfurt's */
      [1.1.1.1]  /\[-.1] /* AC breaks out under clouds */
                 /\<-.2> /* CRW procedures */
                 /\<1.2>
                 /\<2>   /* AC in BATC area */

        <2> /\<-.1> /* LATC procedures */
            /\<-.2> /* LATC uses false flight data for NW052 */
            /\<3>   /* AC in LATC area */

          <2.1> [-.1] /* London received false data from SATC */

          <3> /\<-.1> /* SATC handoff procedures under this flightplan
                         are to LATC */
              /\<-.2> /* FI is correct at SATC */
              /\<4>   /* AC in SATC area */

  <1.2> /\(-.1) /* CRW did not realize that they were on wrong course,
                   UNTIL:[111] */
        /\<-.2> /* AC cleared to BATC according to ATC procedures */

    (1.2.1) /\{-.1} /* CRW addresses BATC controller as ``Frankfurt''
                       several times */
            /\<-.2> /* ILS has different frequency for Frankfurt. */
            /\[-.3] /* CRW asks for the Bruno VOR's frequency. */
            /\(-.4) /* Brussels did not question the addressing error
                       although it happened more than once */
            /\<-.5> /* Situation remains safe during landing */
            /\<-.6> /* Current approach plates are used */
```

Figure 17.8: The Final Textual WB-Graph

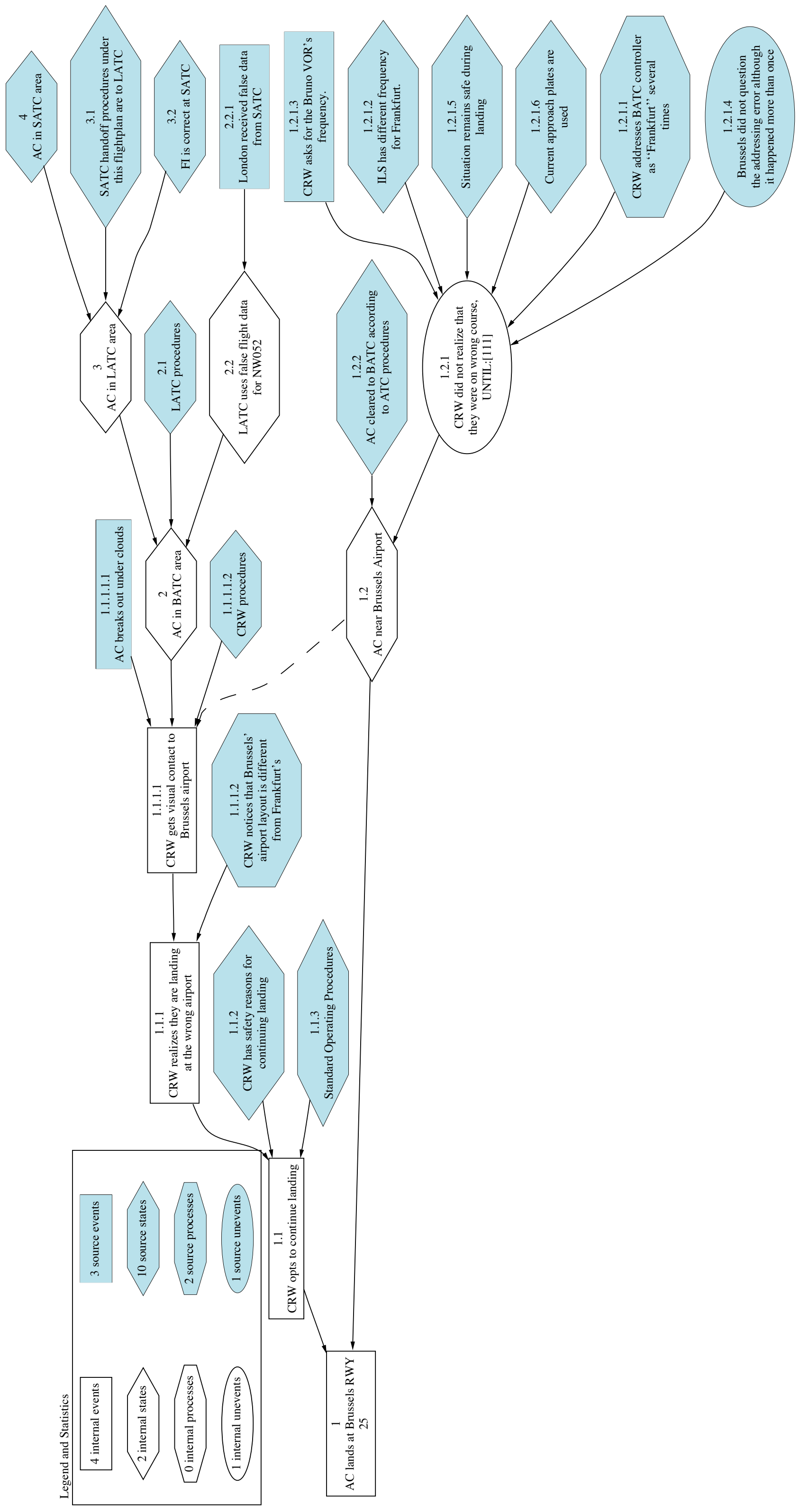
4 AC in SATC area
3.1 SATC handoff procedures under this flightplan are to LATC
3.2 Fl is correct at SATC
2.2.1 London received false data from SATC
1.2.1.3 CRW asks for the Bruno VOR's frequency.
1.2.1.2 ILS has different frequency for Frankfurt.
1.2.1.5 Situation remains safe during landing
1.2.1.6 Current approach plates are used
1.2.1.1 CRW addresses BATC controller as ''Frankfurt'' several times
1.2.1.4 Brussels did not question the addressing error although it happened more than once
3 AC in LATC area
2.1 LATC procedures
2.2 LATC uses false flight data for NW052
1.2.2 AC cleared to BATC according to ATC procedures
1.2.1 CRW did not realize that they were on wrong course, UNTIL:[1.1.1]
1.1.1.1.1 AC breaks out under clouds
2 AC in BATC area
1.1.1.1.2 CRW procedures
1.2 AC near Brussels Airport
1.1.1.1 CRW gets visual contact to Brussels airport
1.1.1.2 CRW notices that Brussels' airport layout is different from Frankfurt's
1.1.1 CRW realizes they are landing at the wrong airport
1.1.2 CRW has safety reasons for continuing landing
1.1.3 Standard Operating Procedures
1.1 CRW opts to continue landing
1 AC lands at Brussels RWY 25
Legend and Statistics
3 source events
10 source states
2 source processes
1 source unevents
4 internal events
2 internal states
0 internal processes
1 internal unevents

Figure 17.9: Automatically generated WB-Graph