ATRテクニカルレポート表紙

〔公　開〕

TR－C－0105

Screening Services Simulation

ファイサル　ジトーニ
Faycal ZITOUNI

1995　2．21

ATR通信システム研究所

Screening services simulation

Faycal Zitouni

February 21th, 1995

Contents

List of Figures

Acknowledgements

I would like to thank Mr. R. Guedj, I.N.T Director for Scientific and International Affairs who gave me the opportunity to accomplish my internship in Japan, Mr. N. Terashima, President of ATR Communication Systems Research Laboratories for having invited me to come to ATR for six months.

I would like to express my sincere gratitude to Mr. T. Ohta, Head of the Communications Software Department who welcomed me in his department; and especially Mr. Y. Kawarazaki, my supervisor in ATR who advised me and helped me for my work.

I would like to thank Mr. Raffy, my scientific advisor for his encouragement, Mr. Tsurukame for his patience and valuable efforts communicating with me in English.

Finally, I would like to thank Mr. K. Fujii, Senior Staff of the Planning Division and all the members of the Planning Division and the Planning Section of ATR Communication Systems research Laboratories for having organized so perfectly my stay in Japan.

Chapter 1

ATR presentation

ATR is a newly founded group (1986) whose main purpose is basic research in telecommunications field. Its mainframe is five independent laboratories working in different areas. The total research and development funds given to the five laboratories each year are about 9 billion yens.

The five laboratories are:

ATR Communication systems research laboratories (Research on Communications with Realistic Sensations, Security and Automatic Generation of Communications Software).

ATR Interpreting Telephony Research Laboratories (Speech recognition, machine translation, speech synthesis).

ATR Auditory and Visual Perception Research Laboratories (Mechanisms of perception and cognitio in the human senses of sight and hearing).

ATR Optical and Radio Communications Research Laboratories (Communication devices based on artificially modulated material structure).

ATR Human Information Processing Research Laboratories enhancing human-machine communication technologies.

see[1] for the hairy details.

grade 3, class 2, and student number 4)

Chapter 2

Presentation of the screening services

2.1 Introduction

Our aim is to introduce a new concept, where the customers can define their own services.

2.2 Definition

A screening service is a service where the customer defines screening elements that are conditions under which STR services perform.

2.3 Generality of the concept

We are in a situation where defining screening services as a new concept is not very easy. Having more details about this notion, and its further developments requires an example.

So many methods are possible, all of them depending on the screening service definition tool that is used.

In our case, NTT's laboratories succeeded in modeling those services. Our whole study here is about the screening services definition using *NTT's system* as definition tool.

2.4 Few definitions more

Here are further definitions, related to the NTT's system.

Network Definition

The Network Definition is the first phase of the service definition. In fact, it is a declaration of items to specify the environment.

Call Control Procedure Definition

The Call Control Procedure Definition is the second phase of service definition. For a given item, we specify a call control procedure by using item control elements and screening elements.

Screening type

A screening type is in fact the type of the screening element. (see section 5.4).

Screening Conditions

Screening Conditions are the conditions under which item control elements perform.

Condition Values

Condition Values are the simulation screening values that allow the call control procedure to choose one item control element. subsectionexample

Chapter 3

Overview of the environment

3.1 Introduction

The software department has been studying the communication service specification with the state transition rule (STR), and the verification of service interactions. It is also developing an Interpreter System of the STR (IPS) as a simulator of service definitions on an Advanced Intelligent Network (IN).

3.2 Theme

Cooperation with NTT Communication Switching Laboratories exists in order to build the Service Creation Environment (SCE) for an Intelligent Network. In this team, ATR is in charge of the simulation whereas NTT has the responsibility of all the others: service definition, compilation and linking. (see figure 2.1).

We know that services are described in S/N-scenarios. A S-scenario is customer defined services. It allows a customer to describe network/resource definition and the service conditions to screen. A N-scenario is a service developer-defined service. It allows to describe the function component to the S-scenarios. These scenarios are confirmed by simulator, then sequentially compiled, linked into the executable form in SCP[1].

[1] Service Control Point: It is a component of the IN.

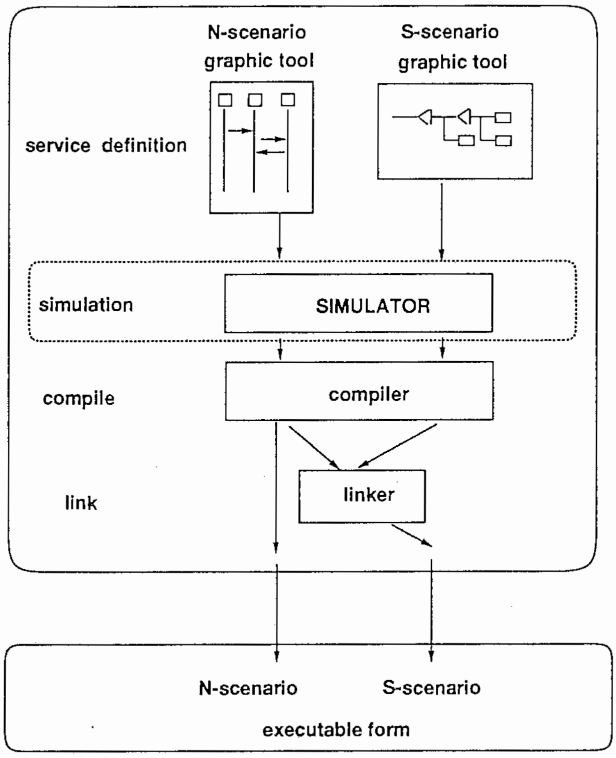


Figure 3.1: Service Creation Environment

3.3 Task

Our purpose is to estimate and design a method of realizing screening functions in a Simulator. To achieve our aim, we should decide the following:

- What kind of screening functions ?
- How to describe the screening functions through STR ?
- How to translate the S-scenario into the description of the screening functions ?
- Difference between the IPS and the Simulator ?
- Relation between IPS and the Screening Functions ?

- Implementation of the Simulator ?
- GUI of the Simulation ?

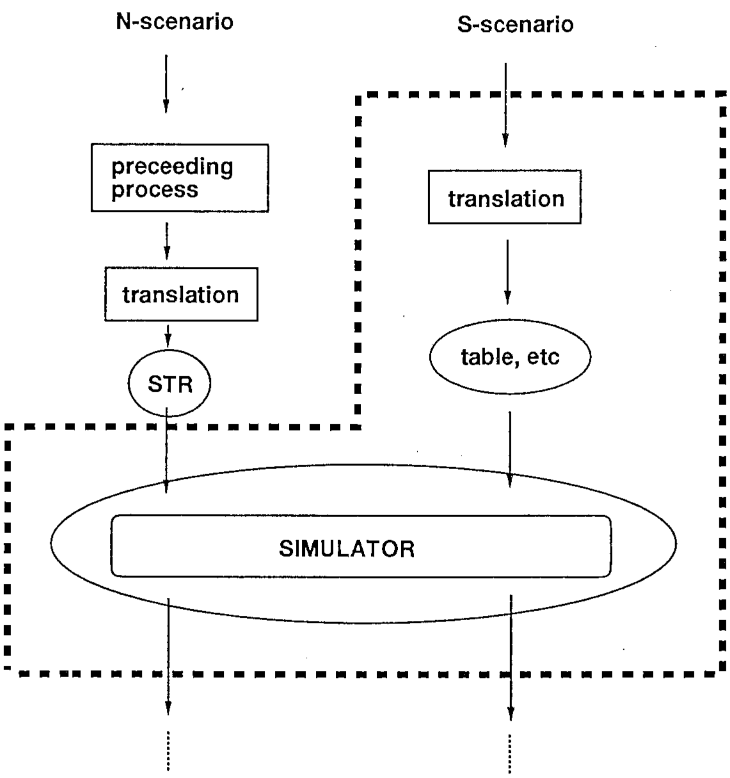


Figure 3.2: Images of each scenario processing

Chapter 4

STR specification for screening

4.1 Introduction

Since we have already defined our aims, the next step is to start making an STR specification for screening. In other words, how can we expess in an STR rule the notion of screening ?

4.2 Simulator structure

According to me, the notion of screening services has two aspects:

- The action that should be performed by the Simulator, in the sense of applying an STR rule.
- And the checking of the screening conditions, that could lead to the application or not of an STR rule.

Our starting point is the current Interpreter (IPS). Therefore, it has been taken for granted, and agreed that the Simulator will be composed of two programs: the Current Interpreter (IPS) and the Screening Function Block (SF Block).

The SF block will have to be created, and the current IPS will have to be changed, so it can handle both the simulation of the screening services and the service definitions.

IPS calls the SF block from time to time (when needed), and carry on simulating.

4.3 Draft ideas

Basically, we could focus our research on three main ideas that are potentially representative of all the tasks mentionned below.

4.3.1 The rule setting method

STR uses rules to describe service operations. Our idea is to set new *screening rules* with a missing next state. The current state would have a trigger which generates the calling of the SF block. The trigger could be a *memory-primitive* like *m-screen(A)*. After consulting a screening table, the SF block finds the next state and sends it back to the IPS.

current state	event	next state
dial-tone(A), idle(B), m-screen(B)	dial(A,B)	Ø

Figure 4.1:

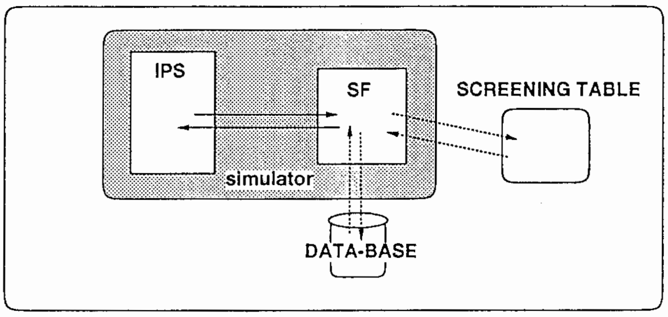


Figure 4.2: Rule setting method

4.3.2 Declaring new primitives

Here, we intend to add some new primitives to set the rules. IPS will call the SF block as soon as a *screening primitive* will be detected in the current state of a rule. The SF block checks which *screening primitive* is satisfied, thanks to the screening table.

Example : use of pseudo-event

To set this idea, we need to divide the STR rule into two parts.

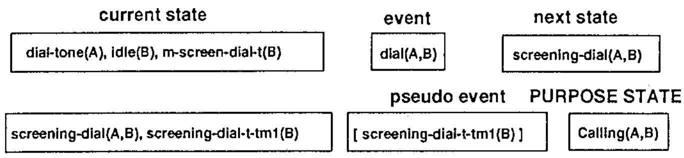


Figure 4.3: Use of pseudo event

When the state of the object "B" becomes one of the set of the primitive(s) in the pseudo event, the second rule will be applicable.

4.3.3 Using variables

In STR, the expressions *equal[count-primitive-string]* and *not-equal[count-primitive-string]* are used in rules to describe comparison of the counter values of primitives that have counters to the declared values.

In our case, and because we intend to write screening rules the *equal* syntax could be used with screening variables in the current state of a rule.

dial-tone (A), idle(B), equal[X1,1] dial(A,B) : Calling(A,B).

Figure 4.4: Format of a screening rule

In fact, the IPS will call the SF block, asking for a value for the screening variable. SF gives a value to this screening variable, after consulting a screening table.(see figure 4.5 in the next page).

4.4 Software department constraints and requirements

The constraint that has the priority, is the non-changing of the STR specification. In other words, the chosen idea among the previously exposed will have to perform a screening simulation with new concepts that fit with STR.

Futhermore, and because the STR interpreter is our starting point of work, we assume that the IPS will have the leading role in the Simulator. No doubt that some changes should be introduced in the current IPS, but those changes have to be reduced.

4.5 The final choice

After the confrontation between the draft ideas, and the constraints we have in our work, some methods are much more suitable than others: the rule setting method for example, would obviously affect directly the STR specification, because of the absence of a next state in the screening STR rules. In this way, one of the fundamental requirement of our research would not be satisfied.

In the remaining two ideas that concern the same constraints and requirements, we prefered to carry on researching with the use of variables. The already existing *equal* syntax in the STR, and the complexity of the *pseudo-event method* have pushed us to choose the use of *screening variables* as a main idea.

The other ideas will of course be kept in mind for later researchers.

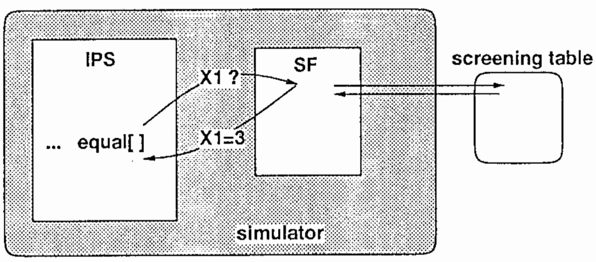


Figure 4.5: Use of screening variables

Chapter 5

Interface between IPS and the SF block

5.1 Introduction

Given the environment we are working in–two different programs that are IPS and the SF block– a simple call procedure is enough to make those two parts of the Simulator communicate. As we said in the previous chapter, the idea that will be developed in our research is the one concerning the use of screening variables. But, even by now, the use of this idea has not been clarified yet.

5.2 Back to the roots

We have already exposed clearly the reasons why we choosed to use the screening variables, instead of the other ideas proposed previously. But we still can find some deep motivation in the fact that even with NTT's system, many concepts are easily expressed through the use of screening variables.

Starting from the elements of the graphic tool used in the screening services definition, we have the concrete representation of these notions when we approach the screening problem with the use of screening variables.

NTT's SYSTEM	SIMULATOR
SCREENING ELEMENT	SCREENING VARIABLE
SCREENING CONDITION	VALUE OF THE SCREENING VARIABLE

Figure 5.1: Correspondence of elements

5.3 Structure of the Simulator

An overview of the Simulator has already been given, saying that there will be two parts that are the current IPS (slightly changed) and the Screening Functions Block.

5.3.1 The current Interpreter in the Simulator

The current Interpreter is already used in the Software Department as a simulator of service definition, although some imperfections has not been solved yet.

As far as all the screening services will be described in STR, the IPS will have to deal with the screening rules as classical rules (in the STR meaning).

$$current\ state...\ equal[X1, 3]\ \ event:\ \ next\ state.$$

The only difference is that the *value of the screening variable X1* is not available in the IPS. This missing data is sent from the SF block, after the Interpreter has called it with the right parameters.

Which parameters are needed for calling the SF ?

- The first one is the name of the screening variable. In our example, it is "X1" that will be sent to the SF.
- But we also need a *CONDITION VALUE*[1] related to the screening type of the screening variable, and corresponding to the condition choosed by the user during the simulation.

[1] Generally, a condition value is a simulation screening value allowing a call control procedure to choose one item control element.

5.3.2 The Screening Function Block.

The SF Block is the second part of the Simulator. For each calling from the IPS, a screening function will have to affect a screening variable a value, thanks to the screening conditions defined before by the customer in the NTT's system.

The internal organization of the SF block should be thought in order to simplify the communication with the IPS. The block will be composed of different Screening Functions that will decide, for each screening type, what is the value to affect to the screening variable. But we also think to introduce a *Head Function* to allow the SF block to read the screening conditions related to the different call control procedures defined in NTT's system.

In fact, the Head Fuction, using the screening type of the screening variable will read the right screening conditions, and will call the right Screening Function.

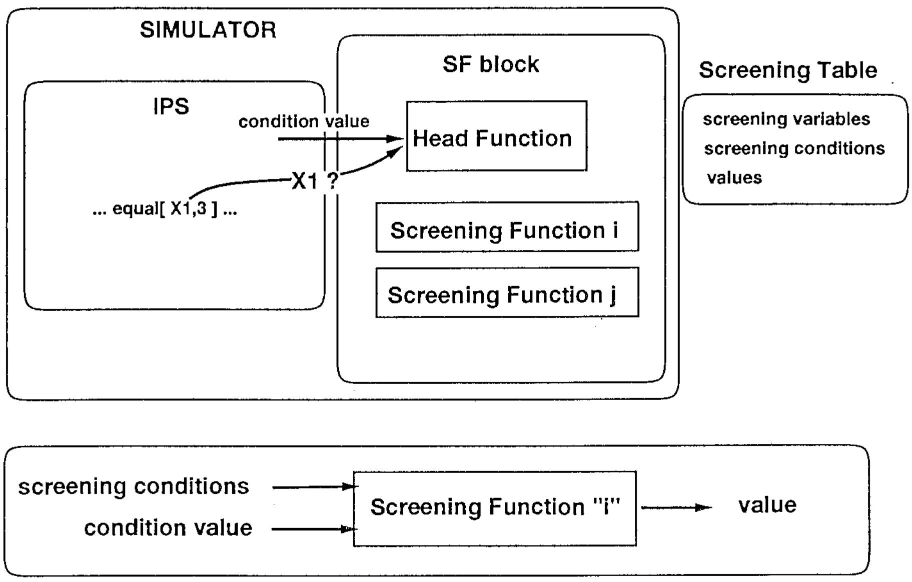


Figure 5.2: The Screening Function Block structure

5.4 The problem of the screening elements

Basically, one of the themes of our research was to think about all the screening elements that could be proposed for our simulator. But, as far as we simulate only S-scenarios defined by NTT's system, the range of screening elements is already decided.

Futhermore, as soon as we carry on finding new screening elements, we face the problem of the *identification of objects in STR*. All the objects represented by the STR rules are terminals. Persons or answering machines—for example, are part of another field of research, that is to create new primitives for describing the different states of those objects, or their service registration status.

For the rest of our work, here is a representative list of usual screening elements.

SCREENING ELEMENT	CONDITION VALUE	CALLER/CALLEE
Time of day	time of day	caller/callee
Day of week	day of week	caller/callee
Day of month	day of month	caller/callee
Holiday of year	holiday of year	caller/callee
Originating area	area code	callee
Originating terminal	tel number	callee
Terminating area	dialed area code	caller
Terminating terminal	dialed tel number	caller

Figure 5.3: Screening Elements

Chapter 6

Identification and registration in a Simulation

6.1 Introduction

STR rules are originally written for all the terminals that are in a certain state (current state included in the real state). That's the reason why STR rules are said to be *general*.

During the network definition, not all the terminals of a network are granted screening services: only some of them are related to a call control procedure definition. Therefore, how can we express this specificity in a STR screening rule ?

6.2 The memory-primitives

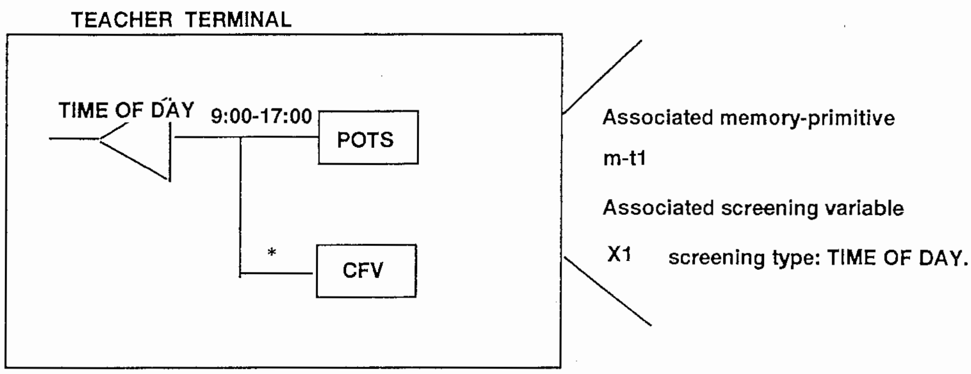
The STR gives us the possibility to represent a *service registration* through primitives that are prefixed with 'm-': the memory-primitives. (for example m-cw(A) used for the call waiting service).

In this way, we consider a screening service as a service that has to be registrated first, before performing its simulation. But this will be solved later[1]. By now, our idea is to associate each CALL CONTROL PROCEDURE to a specific memory-primitive, that will be all through the STR rules, the symbol of a certain screening service.

[1]see the Auto Run File

6.3 The screening variables

Following the same trend as below–relating STR screening rules and call control procedures– we decided to define each screening variable for a certain call control procedure. In fact, mentionning the name of a screening variable would be enough to determine which terminal or group of terminals is concerned.



dial-tone(A), m-t1(B), idle(B), equal[X1,1] dial(A,B) : Calling(A,B).

dial-tone(A), m-t1(B), idle(B), equal[X1,2] dial(A,B) : Calling(A,C).

Figure 6.1: Identification of terminals

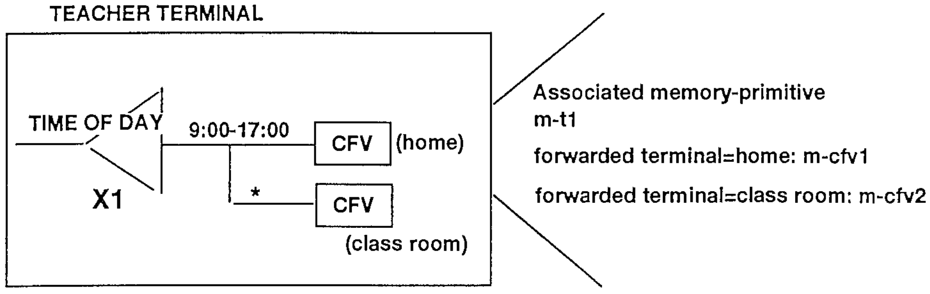
6.4 The CFV service specificity

In some cases, it can happen that more than one CFV service[2] are proposed, as N-scenarios, in the same call control procedure. Knowing that a forwarded terminal has to be designated for each CFV service, how can we make this appear clearly in an STR screening rule ?

In fact, memory-primitives will be used a second time, to solve this problem. Here is an example, with figures, that makes it clearer.

[2] In fact, this section concerns all the STR services where a subscriber must designate a second terminal (for example the SCR service).

v




```
dial-tone(A), idle(B), m-t1(B), m-cfv1(B,C), equal[ X1,1 ] dial(A,B):
                              ringing(C,A), ringback(A,C), pingring(B,C).

dial-tone(A), idle(B), m-t1(B), m-cfv2(B,C), equal[ X1,2 ] dial(A,B):
                              ringing(C,A), ringback(A,C), pingring(B,C).
```

Figure 6.2: m-cfv1:identifying terminals

6.5 Limits of our method

Apparently, the method proposed previously for identifying terminals (especially in the STR screening rules) is correct. But the problem comes from the *generality* of STR rules that may be lost, when using such method.

Views of researchers here are very divided. And pulling through requires much more time than expected. So many meetings have been held and no immediat answer could be found.

Therefore, our work will carry on, assuming that this idea is reliable.

6.6 Registrating screening services

The use of memory-primitives imposes a step of registration, as it is the case for STR services like CFV or 3WC.

For those services, the registration is manual: the user is supposed to make an event occur, and a rule is applied, changing the state of the concerned terminal.

The similarity with screening services is obvious, and our idea is to have *screening service registration events* that will appear in STR rules.

```
idle(B) t1(B) : m-t1(B).
```

Figure 6.3: The screening service registration event

It could also be done automatically. The Auto Run File[3] could be run before a Simulation, registrating all the services that were indicated.

This is the choice that was made for building the Simulator: using screening service registration events, which means writing STR rules (see figure below); and having the Auto Run File responsible of getting those rules to be applied.

[3] This notion has been dealt with so many times. Most researchers are only users of this mechanism. Official documents kept being untranslated from japanese. Therefore, the auto run file should be considered as a black box performing service registrations.

Chapter 7

The condition values

7.1 Introduction

As we said in the beginning, each Simulation requires a number of datas. Some of them are provided by the NTT's system after the customer network definition and call control procedure definitions [1]. But others should be given by the user of the Simulator.

Condition values are part of those datas. Basically, they are very similar to the screening conditions in the NTT's system. They are the simulation conditions that allow a call control procedure to choose one item control element.

Which classification can we apply to them ? How does the user provide those condition values to the Simulator ?

7.2 Classification

7.2.1 Condition values set before hand

Introduction

If some condition values can be given to the Simulator before the beginning of the simulation, they are *general*. I mean that they could be used for several terminals. More precisely useful for all the terminals that are registrated for that kind of screening.

TIME CONDITION VALUES represent this category.

[1] But still a conversion is needed, because NTT's system outputs cannot be used directly by the Simulator. More details will be given later.

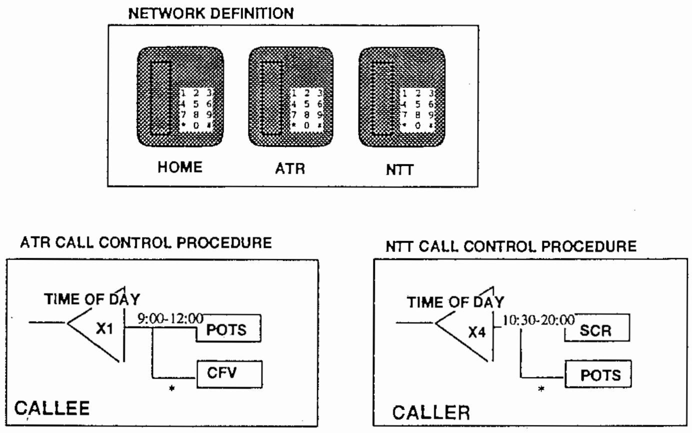


Figure 7.1: Illustration about Time Condition Values

In this example, NTT and ATR have two different call control procedures:

1. NTT's screening is enhanced as a caller screening service, whereas ATR is a callee.

2. The N-scenarios proposed in each call control procedures are different.

3. There are the same screening elements (X1 and X2:Time of Day), but the screening conditions are different.

The generality comes from the fact that, despite these sharp differences, the user can decide before hand (for example 11:04AM) and the simulation could be performed, using a single Time condition value for all the screening variables of this type.

Implementation

Obviously, setting Time condition values is an independant action from the rest. Before each simulation begins, the User, after being asked by the simulator, will decide at what time (day, month or year) he wants to peform the simulation. A simple procedure is needed, to solve this problem.

7.2.2 Other condition values

Introduction

Apart from the Time condition values, all the other screening types belong to the same category.

The generality of the previous category has changed into a complete specificity. Now, each screening variable (for each terminal) will require a seperate condition value. Therefore, none of those condition values could be decided before hand. This should be done during the Simulation.

Implementation

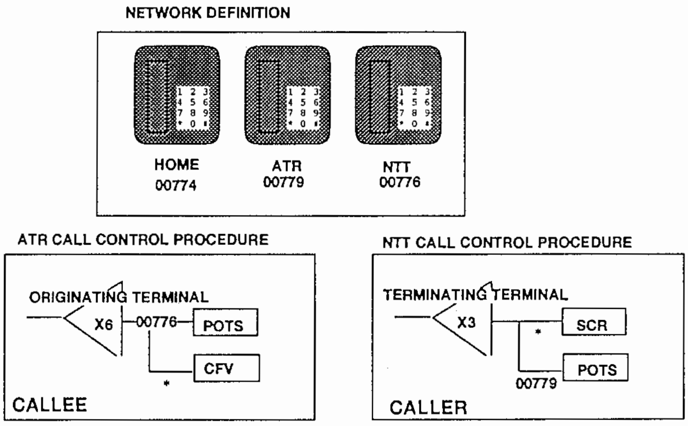


Figure 7.2: Example of “other” condition values

Let’s suppose that an event has just occured. If any STR screening rule is applicable, the none-Time condition values needed for the screening variables of this rule are in fact included in the event of the rule.

dial-tone(A), idle(B), m-t1(B), equal[X6,1] dial(A,B) : Calling(A,B).

dial-tone(A), m-t2(A), idle(B), equal[X3,2] dial(A,B) : Calling(A,B).

Figure 7.3: STR screening rules to be applied

Here, the event *dial(A,B)* will provide the two condition values required for this screening:

1. In the first rule, the screening element is the originating terminal, so it concerns terminating screening services (callee). Therefore, the phone number of terminal A is the condition value needed by the screening variable "X6".

2. In the second rule, the screening element is the terminating terminal, so it concerns originating screening services (caller). Therefore, the phone number of terminal B is the condition value needed by the screening variable "X3".

To refine our classification, and to close the debate concerning condition values, we need to study how the events are represented in the current IPS, and how we can reach the datas needed from those events.

Chapter 8

The Conversion

8.1 Introduction

After the general presentation of the theme, and the draft ideas as a basis of our research, we have reached a point where two main directions appear.

We are building a Simulator and some inputs come from the NTT's system. As a consequence, a Converter should translate all the datas concerning the S-scenarios[1], and present them to the Simulator in an optimized way.A

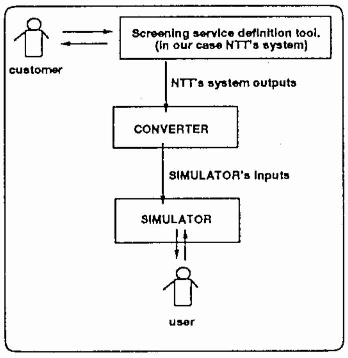


Figure 8.1: The new configuration

During the meetings held, so many discussions allowed us to confirm all

[1] These are NTT's system outputs. See in the Appendix

the datas needed by the Simulator. Those datas could be proposed to the Simulator through 4 files:

1. The Auto Run file.
2. The Object Information file.
3. The Screening Conditions file.
4. The STR Rule file.

The complete samples of those 4 files are located in the appendix B.

The Auto Run file

The Auto Run file is used by the Simulator during the registration of the screening services.

The Object Information file

The Object Information file gives all the datas concerning the objects that compose the network definition:

- The groups constituting the network.
- The number of terminals in each group, their logical and physical numbers.
- And other information concerning their screenings.

The Screening Conditions file

The Screening Conditions file is exclusively used by the SF block. The links between the screening conditions, the screening variables and their values are located in this file.

The STR Rule file

The STR Rule file, as an output of the Converter contains all the STR screening rules, that are the STR translation of the S-scenarios (call control procedures).

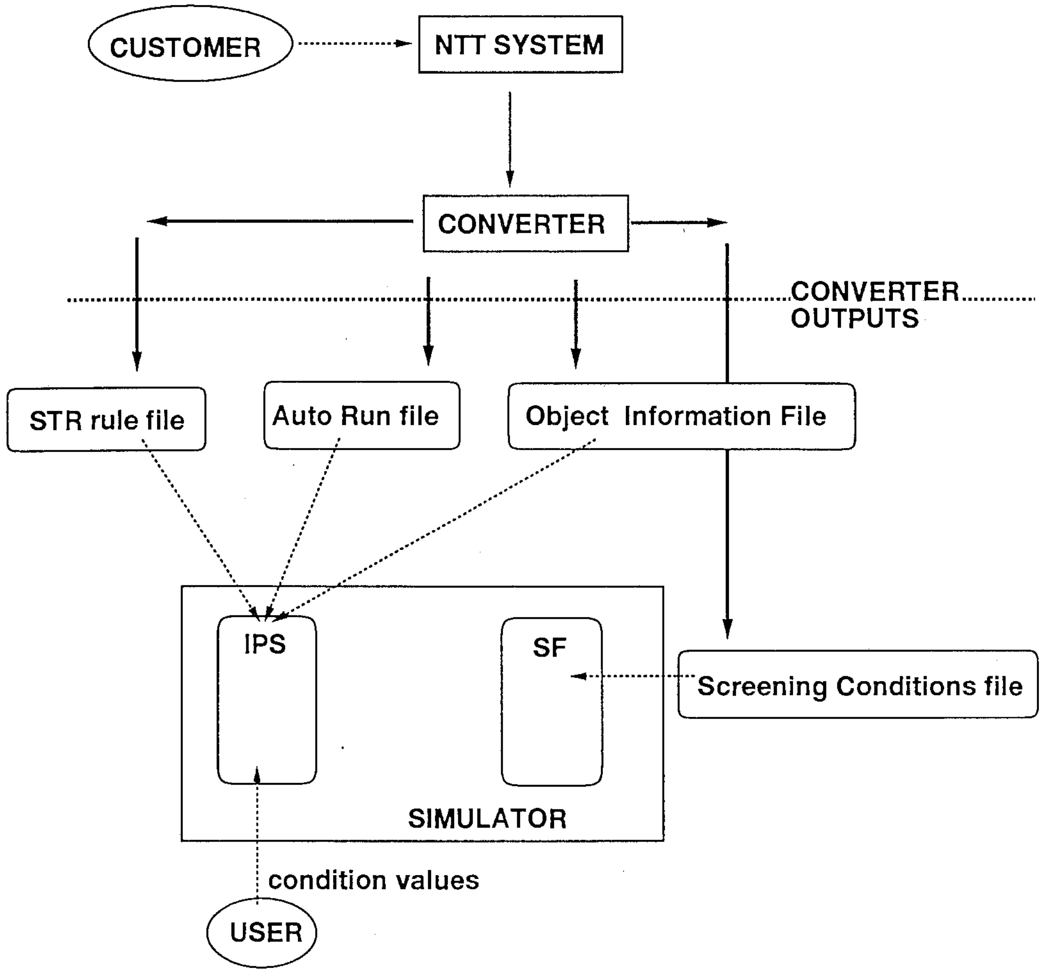


Figure 8.2: The new environment

Chapter 9

The current IPS

9.1 Introduction

Before trying to implement all the concepts of the simulator, starting from the current IPS, maybe is it necessary to clarify the internal mechanism of the IPS.

This chapter gives an overview of the most important procedures of the IPS.

9.2 Frow charts of current IPS

All the following frow charts are official documents from JIP[1]. After the translation from Japanese, the procedures were simplified, to make the document clearer.

[1] The software company who coded the current IPS

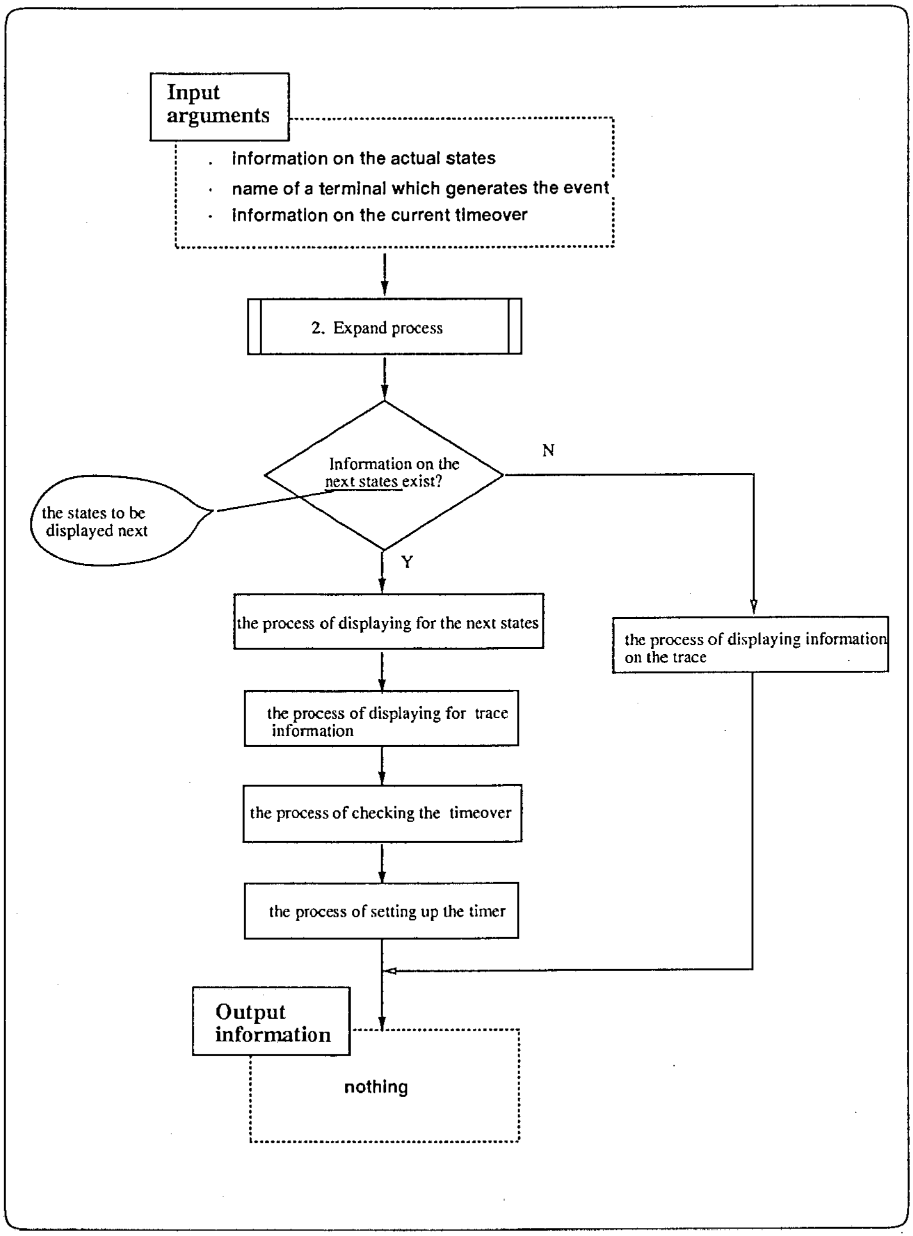


Figure 9.1: The flow chart of the state transition in the current IPS

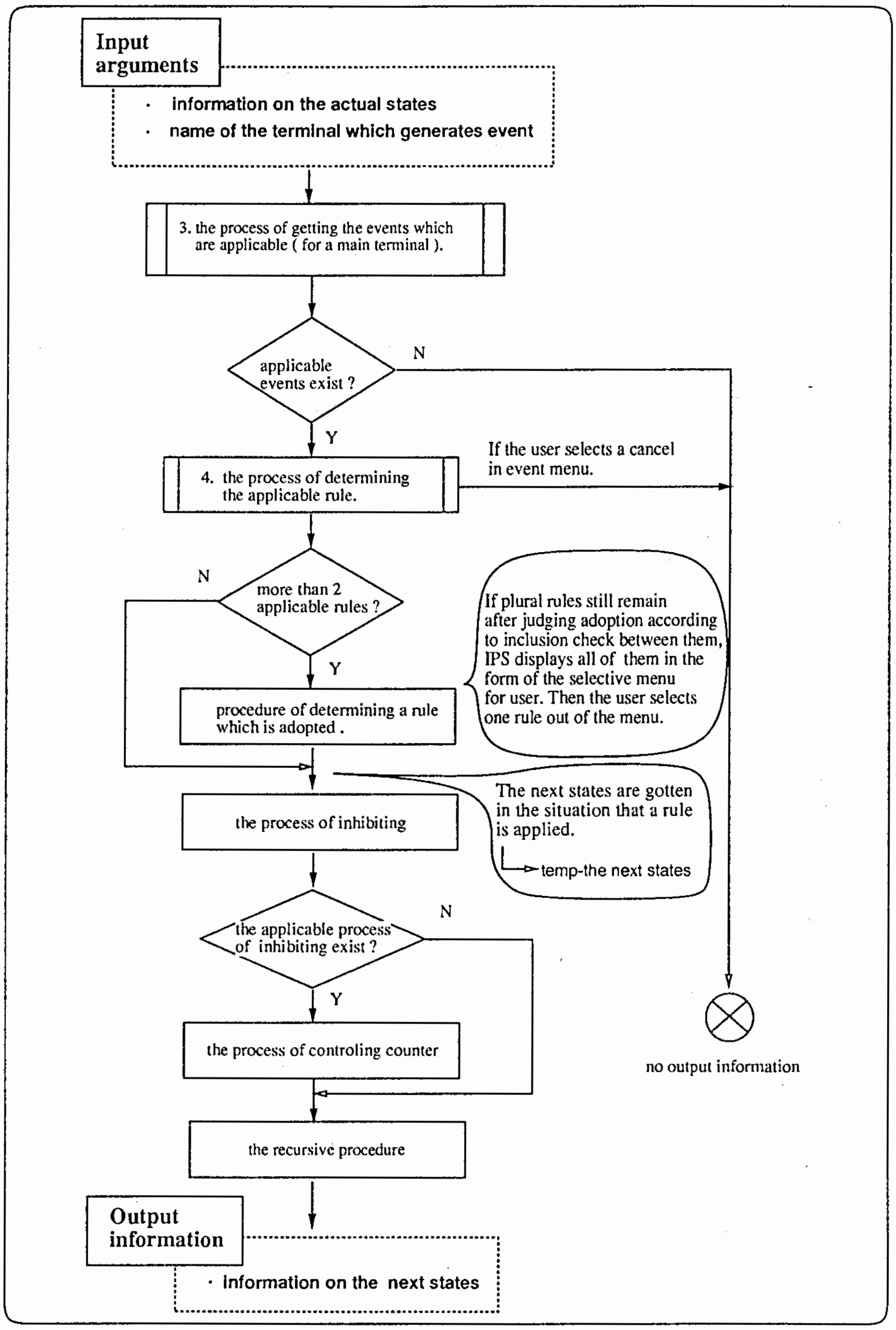


Figure 9.2: The expand process

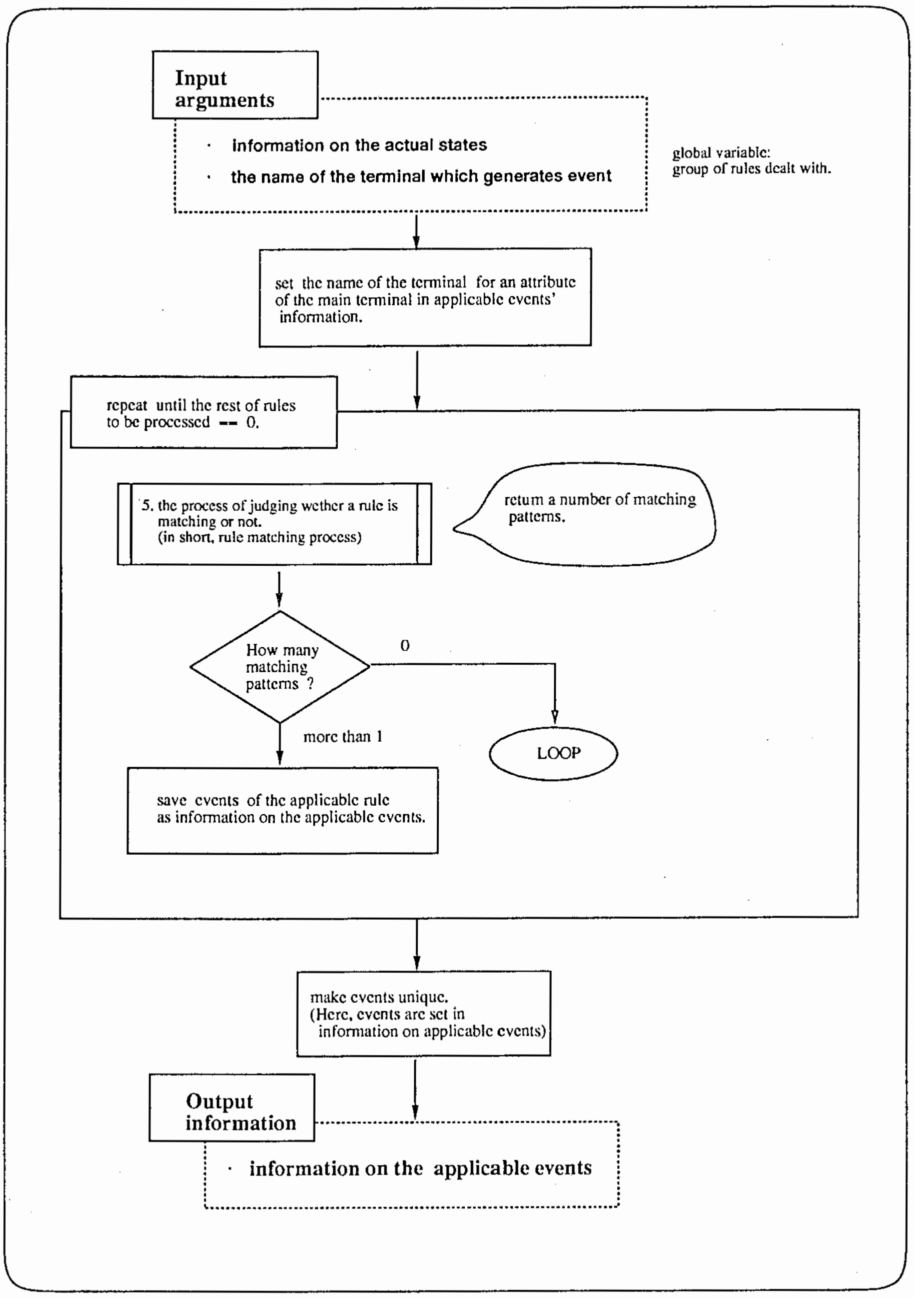


Figure 9.3: The process of getting the applicable events

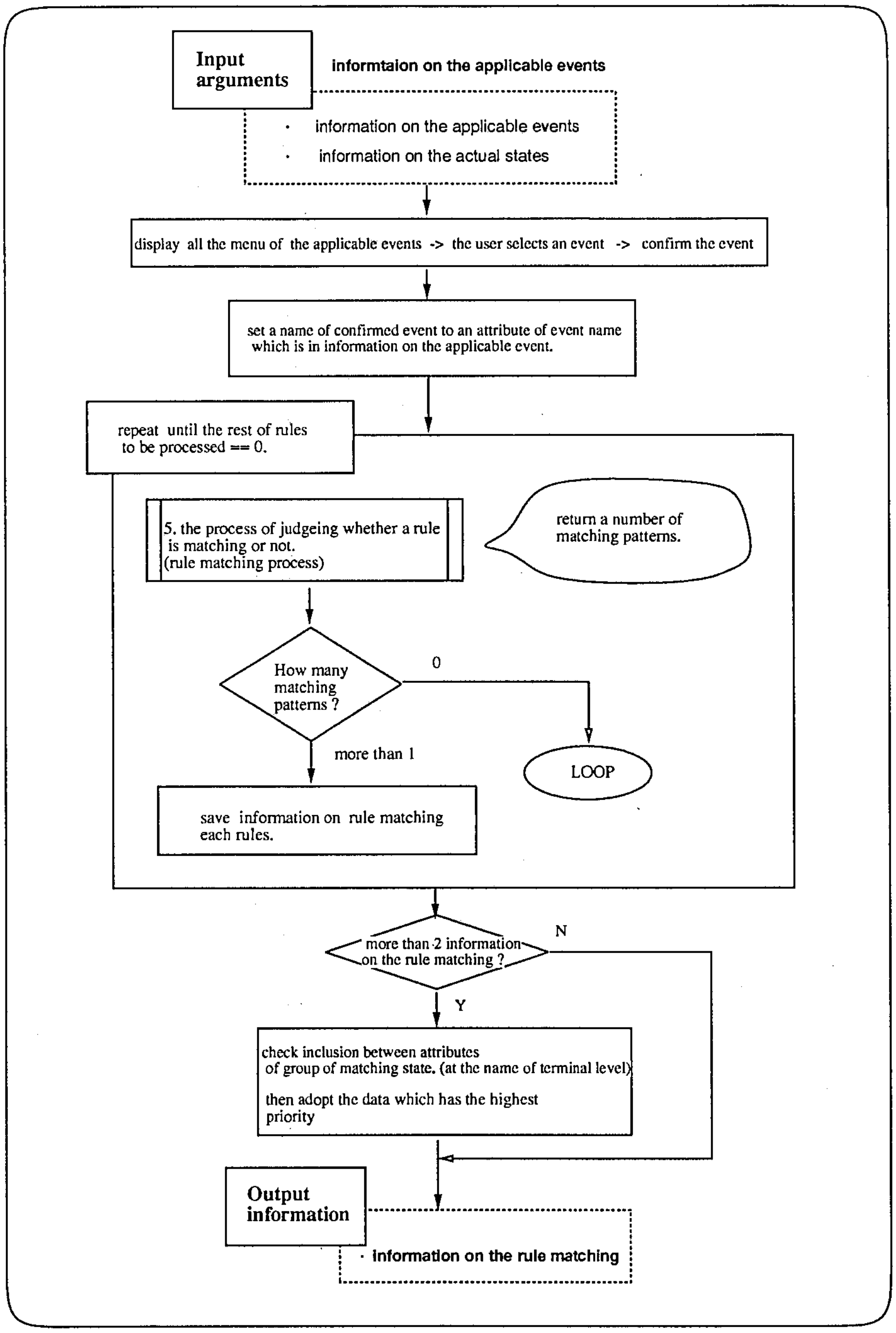


Figure 9.4: The process of determining the applicable rule

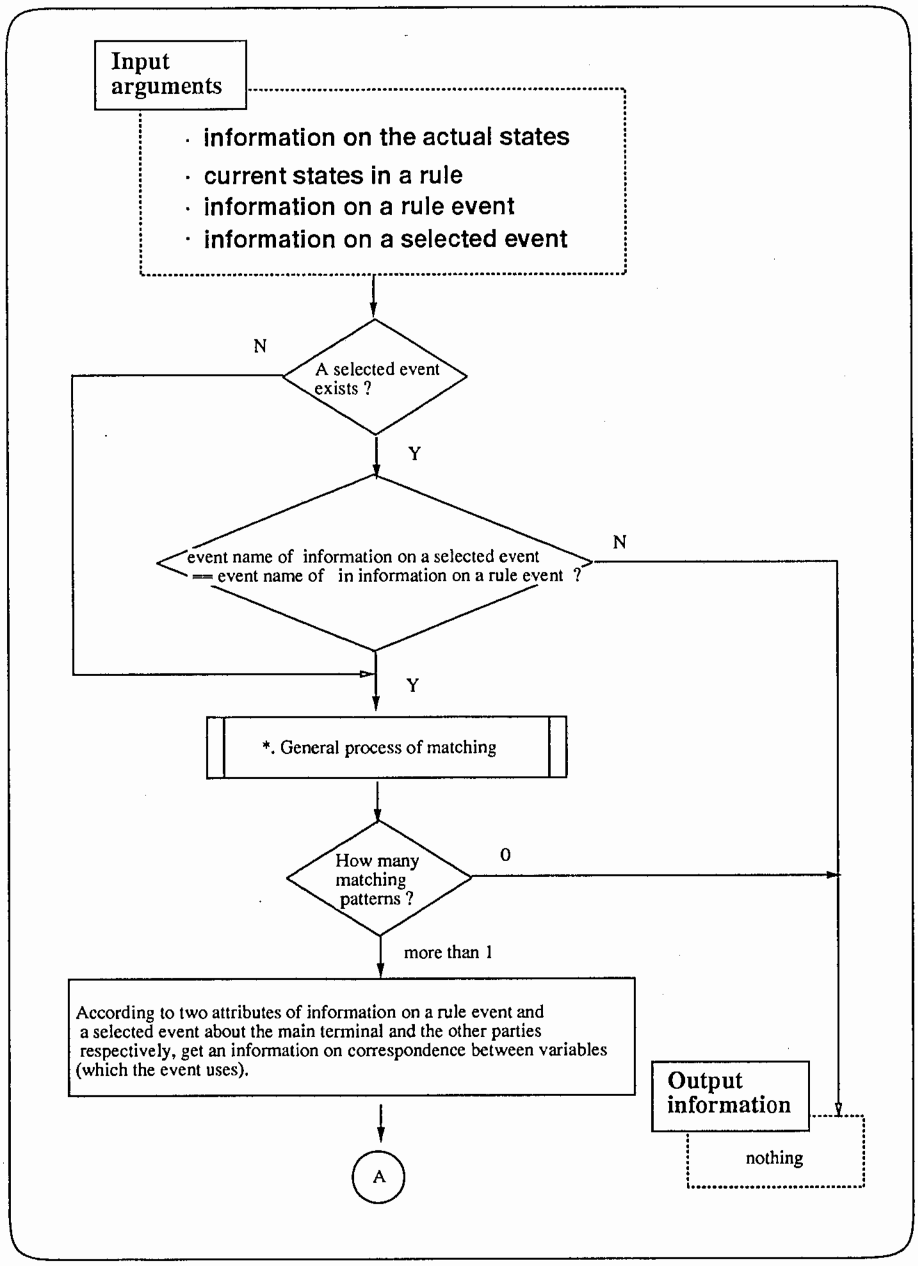


Figure 9.5: The rule matching process (part1)

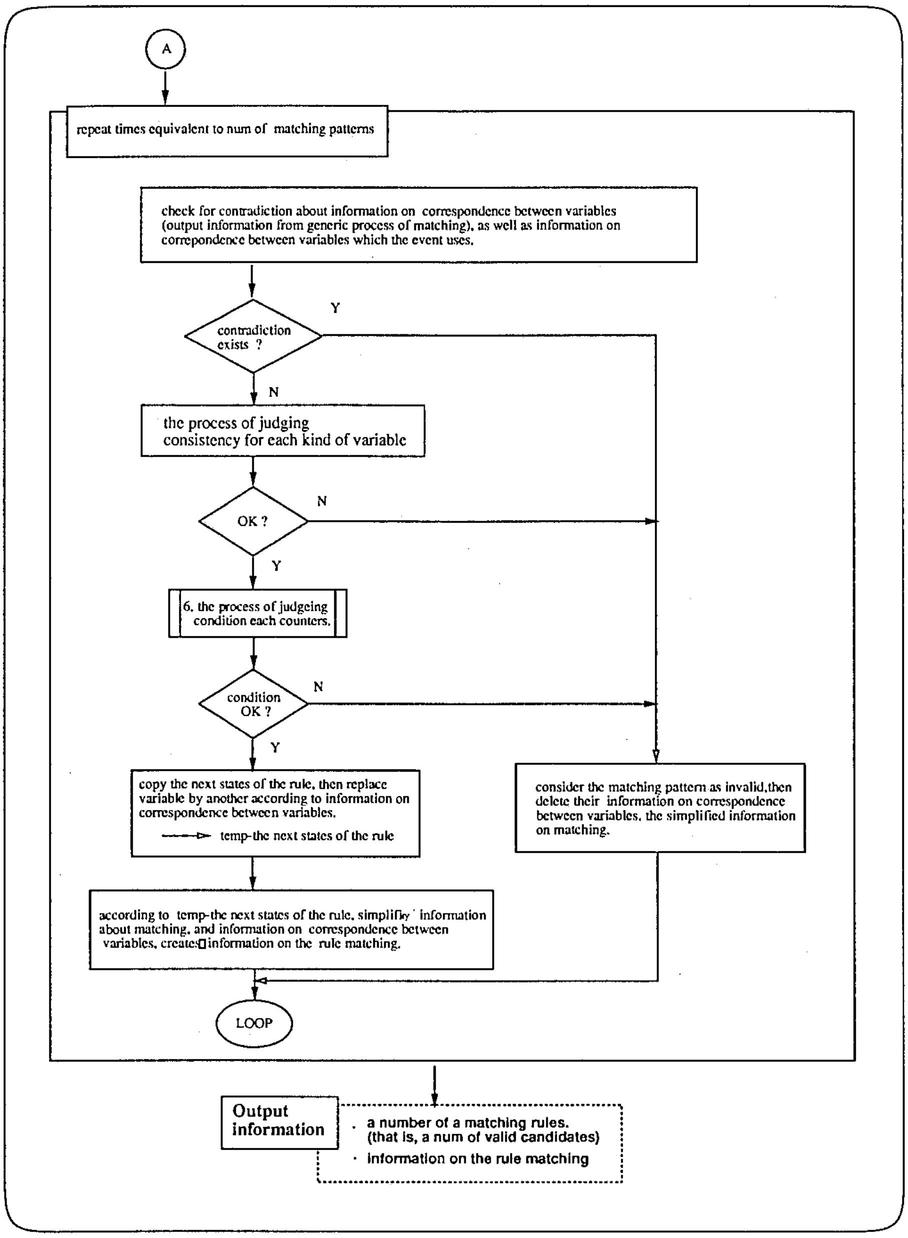


Figure 9.6: The rule matching process (part2)

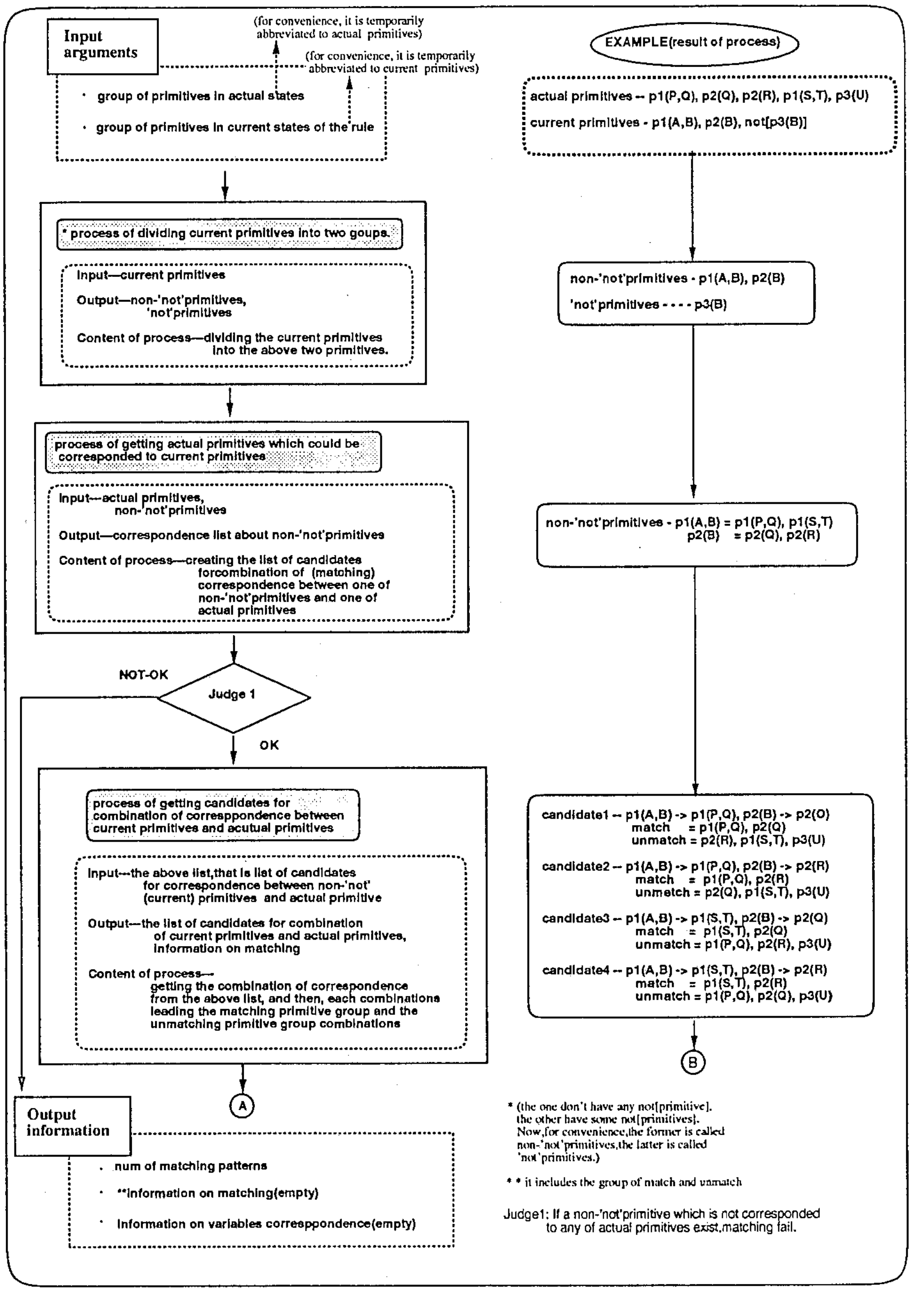


Figure 9.7: The general process of Matching (part1)

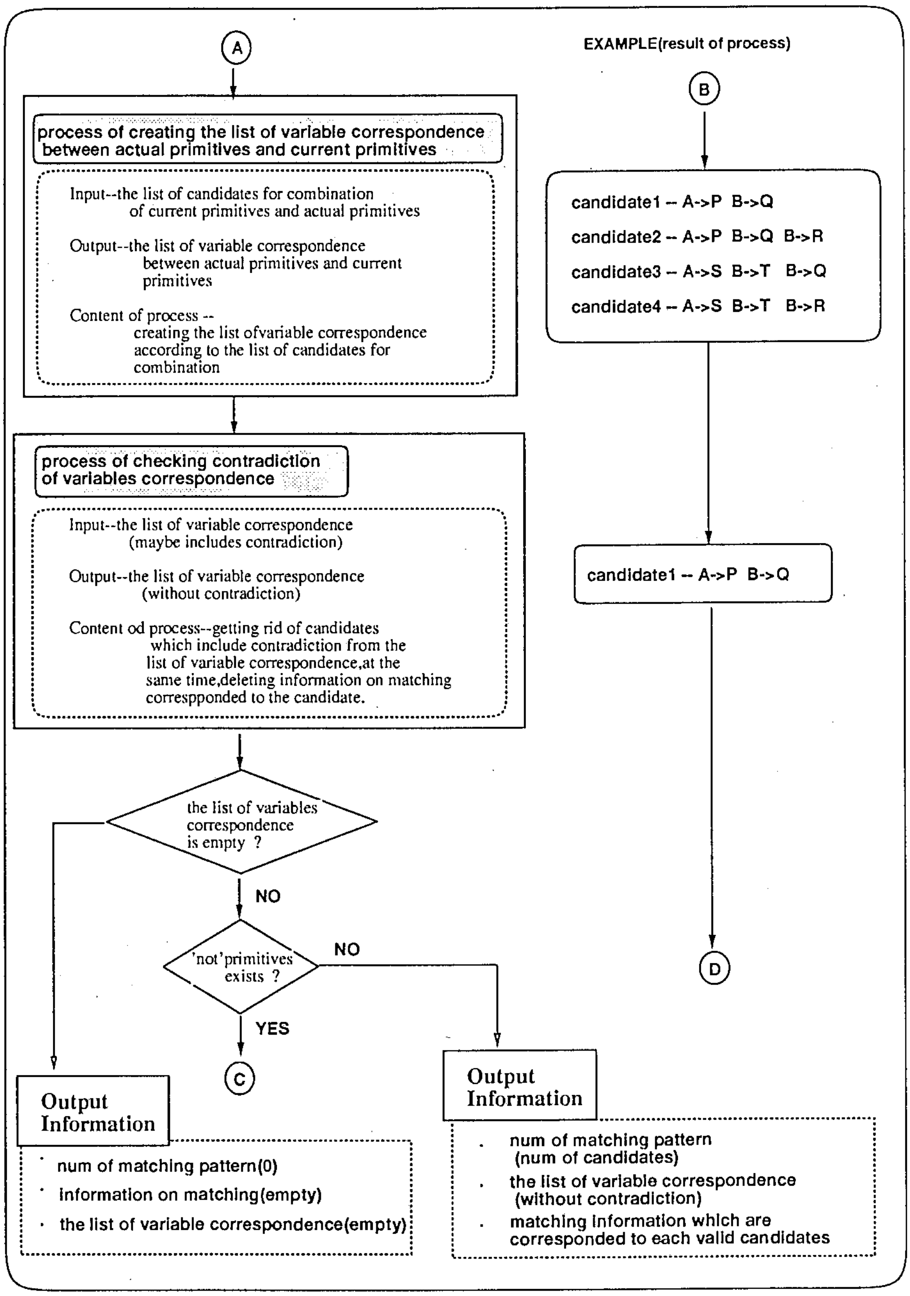


Figure 9.8: The general process of Matching (part2)

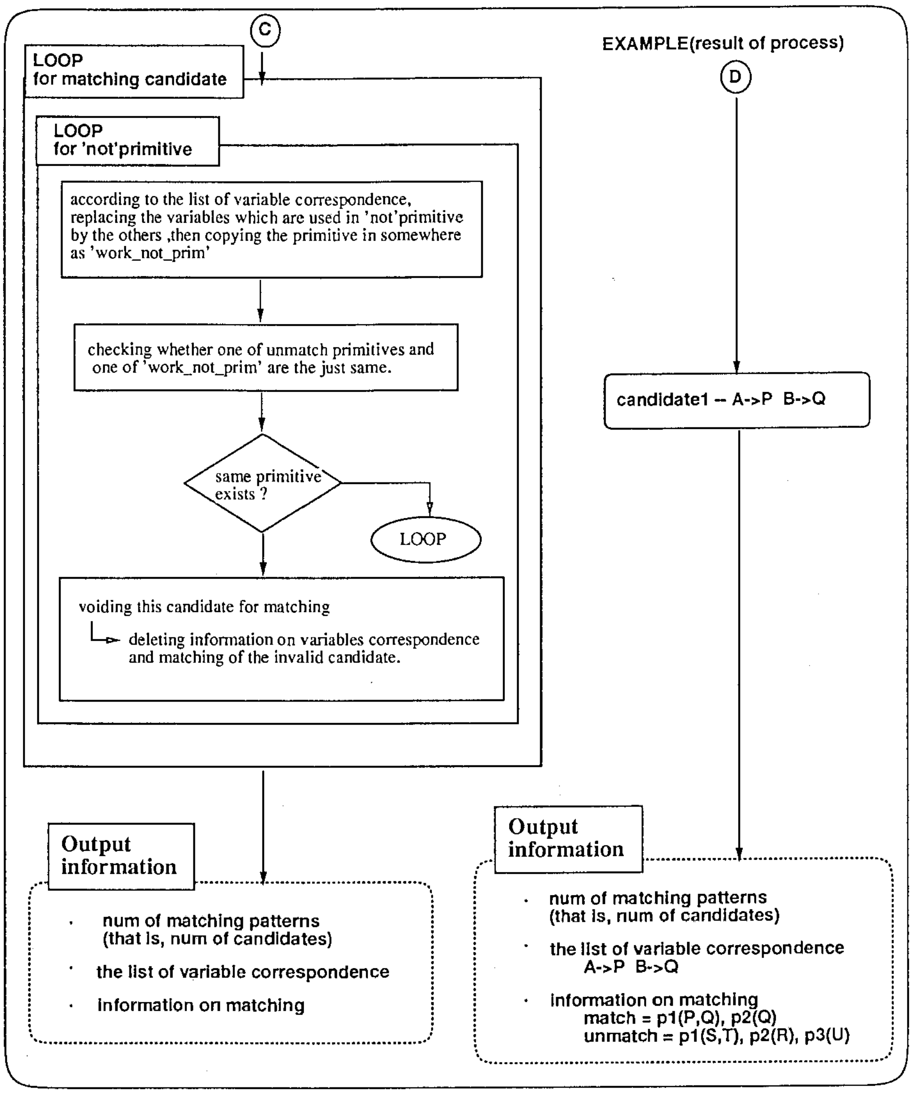


Figure 9.9: The general process of Matching (part3)

Chapter 10

The Simulator: a changing IPS

10.1 Introduction

The internal mechanism of the Current IPS was clarified through the frow charts in the previous chapter. By now, we should find how to harmonize between STR rules and STR screening rules. In this way, the changes in the Current IPS will be minor, because consisting on adding new procedures.

What are those new procedures ?

How to relate them to the existing ones in the Current IPS ?

10.2 Calling the SF Block

10.2.1 Purpose

What should be found first, is the procedure from which the Calling of the SF Block could be performed.

As we know, STR screening rules, in the Simulator, are considered as STR rules. Therefore, so many procedures already existing in the Current IPS can handle STR screening rules. The only difference appears when a value for a screening variable in a current state of an STR screening rule is required. In other words, the only process that is deeply concerned by the changes about the STR screening rules is the *EXPAND PROCESS*.

10.2.2 The state transition

The basic idea here is to try to generalize the state transition for all the STR screening rules. As the figure 9.1 shows it, it is exactly the same event (for three STR screening rules) that makes the state transition from a common current state to three different next states.

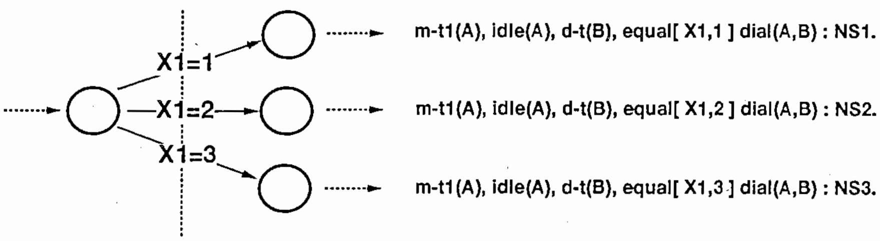


Figure 10.1: The state transition for STR screening rules

As a consequence, the applicable event for an STR rule, and the value of a screening variable (in the sense of applying an STR rule) are independant notions. So, no need changing the *process of getting the applicable event* because through this process, and with STR screening rules, *equal screening primitives* will be let apart.

10.2.3 Calling the SF Block

All the STR rules whose events correspond to the event selected by the user will be checked in the *process of determining the applicable rule*. In this process (more precisely the matching process), and for each STR rule to be checked, the IPS compares the current state of the rule and the actual states.

If there is an inclusion between those two group of states, the rule matching process leads to a number of patterns.(see General Process of Matching).

It is from those patterns (for each pattern) that we will call the SF Block. The constraint set here concerns the number of callings which should be minimized.

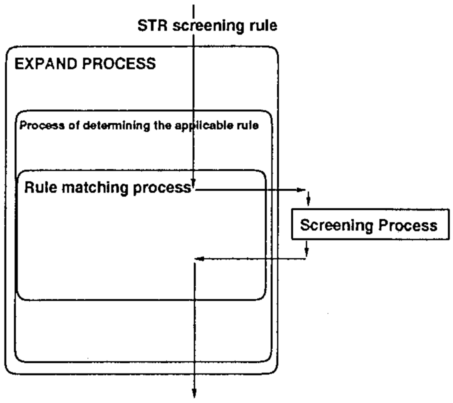


Figure 10.2: Calling the SF Block

10.3 The screening process

10.3.1 Introduction

We said that STR screening rules will be considered as STR rules except for some specific procedures in the Simulator. The screening process is in fact where:

1. IPS calls the SF Block.
2. IPS gets the value given by the SF Block (for each screening variable) and takes the decision wether to apply this rule or not.

10.3.2 Carrying on

Although we know some of the elements of the screening process, the most important is to decide where to call the screening process from, without forgetting our major constraint: minimizing the Calling of the Screening Function Block.

As far as STR screening rules will go through the whole Rule Matching Process, it is suitable to make the Screening Process operate at the end of the Rule Matching Process[1].

At this step, we can introduce the algorithm of the screening process.

[1] Because the last acting procedure in this process is the judging of the conditions of all the counters concerning primitives with counters: *equal[primitive-string]*.

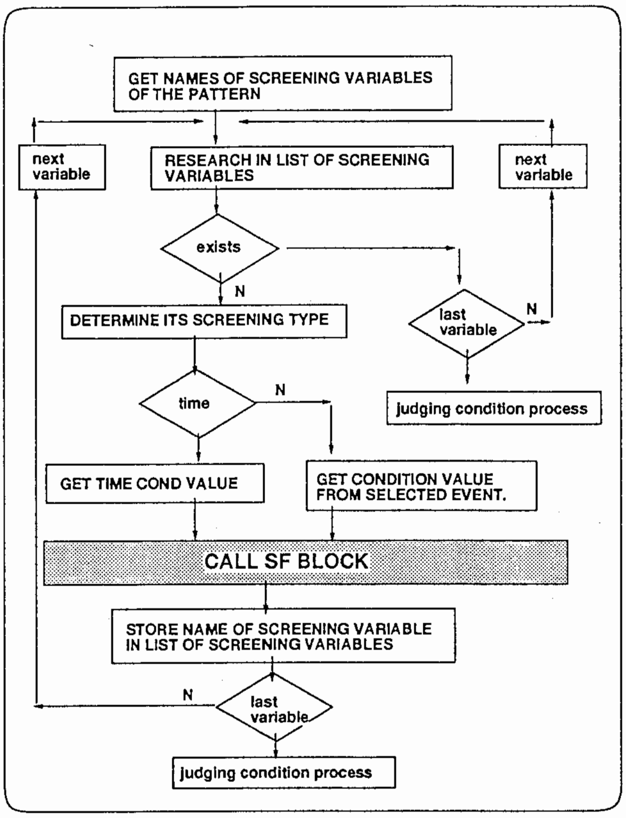


Figure 10.3: The Screening Process

For each procedure, we will precise:

- the inputs.
- the outputs.
- the required datas (global or not).

In this way, the reading part of the 4 input files will become clearer.

10.4 Simulation overview

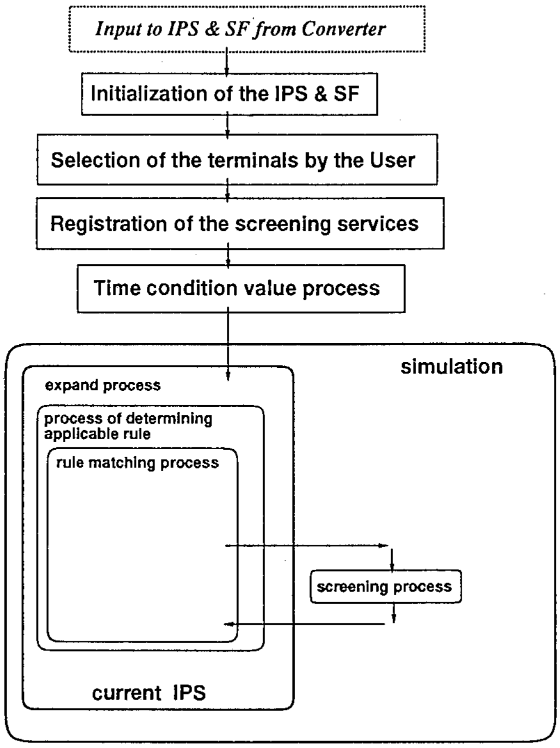


Figure 10.4: Simulation overview

10.5 The Simulator reading part

10.5.1 The initialization

The initialization is the action of converting NTT's system outputs into Simulator inputs: before beginning each Simulation, the user has the choice among several screening service definitions.

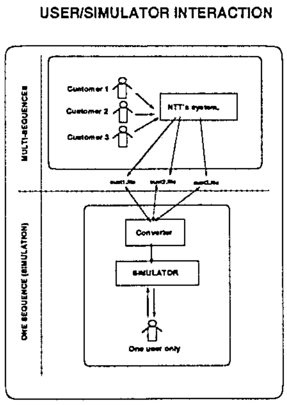


Figure 10.5: The initialization of a Simulation

10.5.2 The selection of terminals

The problem of the selection of terminals was raised since the Current IPS could not display more than 6 terminals whereas NTT's system had no limit. We solved the problem by getting the user to choose no more than 6 terminals for performing a simulation. But we had to face a new situation where:

- Some screening conditions could not be simulated because new terminals had to pop up.
- Implementation would be too long.[2]

Therefore, we decided to design a Simulator fitting with network definitions composed of a maximum of 6 terminals.

10.5.3 The registration of the screening services

As we said in chapter 5, the registration of the screening services is performed by the Auto Run File. Since a whole similarity was set betwen registrating screening services and other STR services (cfv or 3wc), the mechanism of the current IPS is still efficient for the Simulator. Indeed, the Auto Run file sample was not changed. As a consequence, the registrating process remains the same, because it still consists on running the Auto Run file.

[2]According to Japan Information Processing Service CO, LTD (JIP).

10.5.4 The Time condition value process

The Time condition value process is put before the beginning of the Simulation because we can afford calling for all the Time condition values needed in a Simulation.

In the Screening Conditions Information file, the relations between the screening variables and the screening types are available. Therefore, the Time Condition process should check this file, and for each *screening type* concerning Time, ask the user to set the suitable condition value.[3]

10.5.5 The simulation

Getting names of the screening variables of the patterns

In this section, we are in the process of determining the applicable rule, at the end of the rule matching process, with an STR screening rule: the inputs of this process are in fact the outputs of the matching process (information on the matching patterns).

In order to optimize the calling of the SF Block, we decided to store all the screening variables of the *CURRENT RULE* in a list (*List of Screening Variables*). As far as we have the same screening variables for all the patterns of a rule, we agreed to store also the values (answer of the SF Block) in this list. Therefore, if the value is available, we need not call the SF Block.

Research in the list of screening variables

Here, a screening variable requires a value. But we don't know yet if the value is available or not in the list. Therefore, we check it, before calling the SF Block.

Determining a screening type

Determining the screening type of a screening variable could be done by checking the Screening Conditions file.

Getting condition values from selected event

Each time an event occurs, and an STR screening rule happens to be an apllicable rule, the selected event supplies the Simulator with the required condition values.

[3] A global list (for all the Time screening services) could be set in the beginning of the Simulation, concerning this category of condition values. This to avoid checking each time the Screening Conditions file.

But it is still required to make a classification: is it a caller or a callee screening variable that is involved here ?

The file that could provide the answer is the STR Rule file, thanks to the call control procedures identification with the memory primitives.

- The first case:

  $$m - t1(A)... \qquad equal[X1, 2] \;\; dial(A, B): \;\; nextstate.$$

  The screening service is a caller. Therefore, the condition value should be looked for through the variable "B".

- The second case:

  $$m - t3(B)... \qquad equal[X4, 7] \;\; dial(A, B): \;\; nextstate.$$

  The screening service is a callee, and the condition value will be given by the variable "A".

Generally, and for the two cases exposed here, the logical numbers that are widely used for the condition values could be reachable in the actual variables. To achieve that, the combinations between the current and the actual variables (STR rules and actual states) are the only solution. (see the software structure of the current IPS).

10.6 Judging condition process

Introduction

Except from the judging of the equality between the declared value of the screening variable and the value given by the SF Block, the most important problem is the declaration of the *equal screening primitives*.

The limits of a similarity

Our idea started on the similarity between the count-primitives and the notion of screening.for example between

$$equal[m - retry(A)] \qquad and \qquad equal[X3, 6]$$

But the software structure of the IPS (because of the STR specification) revealed a fundamental difference:

- In the constraints of the two notions (equal for screening and for count-primitives).

1. To decide priority between rules, the screening and the count-primitives "equal" have the same status.
2. But the equal primitives for screening have to be ignored in the process of getting applicable events.

- In the declaration of the primitives in the current IPS, no difference is made between the count-primitives and their count decision descriptions. In other words, an expression like

$$equal[m - retry(A)]$$

is in fact declared as a primitive (m-retry(A)). The consequence is its existence in the actual states.

	priority between rules	process of getting the applicable rule
equal[X1,3]	X	O
equal[count-primitive]	X	X

Figure 10.6: The limits of the similarity

Therefore, a new declaration has to be found for the equal syntax of screening.

Declaration of the equal syntax for screening

We decided to declare the equal primitive for screening as an entire primitive. The internal software structure used to declare the usual primitives can still be used here.

Such a structure is directly existing in the actual states, and this implies that the priority between the rules will be automatically performed with the screening equal primitives.

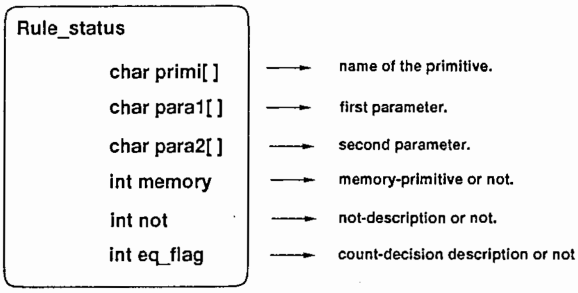


Figure 10.7: Internal data.

- name of the primitive: equal-screen (the same for all of them).
- first parameter: name of the screening variable.
- second parameter: declared value of the screening variable.
- memory and not: put to 0.
- eq-flag: put to 1.

The problem of the current IPS procedures

As far as we have designed a new declaration for the screening equal primitives, we should check if this new declaration satisfies all the constraints.

According to me, some procedures of the current IPS may not handle the new declaration of screening equal primitives.

I propose a solution that is a lot like the general process of matching: we classify all the primitives of the current state of a rule before beginning to run the procedures. Here is an example of the application of this idea to the matching process.

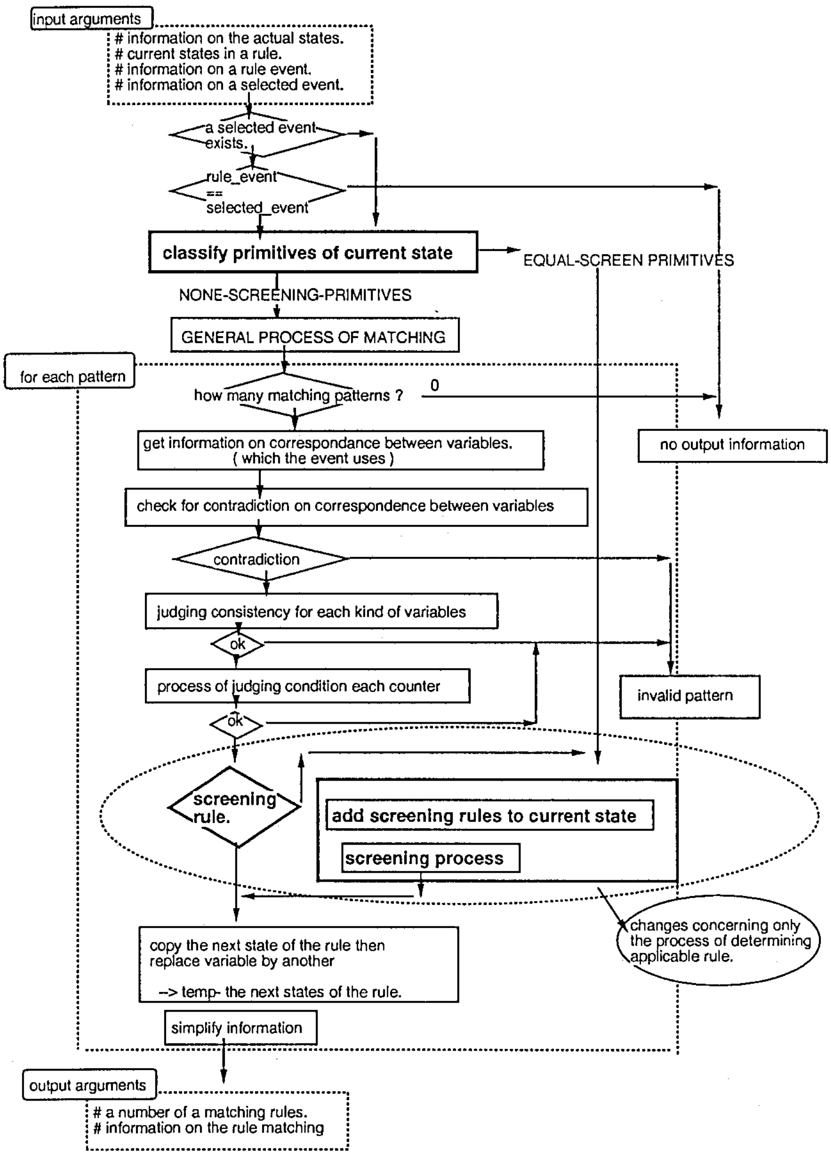


Figure 10.8: The new matching process

Chapter 11

Conclusion

11.1 My internship in ATR

During my six months of stay in ATR, my internship went through several steps, all of them very interesting. I begun by studying the STR, which was all new to me, and I realize now, how free I was to express my ideas, and to carry on getting more and more involved in those in which I believed most.

NTT's system was introduced as a second step, and in fact my purpose was to find out a complete new specification of STR that could enhance this language and make it fit with those new services that are screening services.

The permanent confrontations between the requirements and the constraints of the Software department allowed me to carry on challenging all the problems, one after the other, taking nothing for granted and always trying to improve the solutions.

As a result,and starting from an instinctive and blurred idea of screening variables, we were lead to a whole structured simulator. I guess this is a good engineer experience, full of contact with research field and researchers.

11.2 My personal experience

During my internship in ATR, I acquired experiences and skills from a methodological point of view as well as from a human point of view.

From a methodological point of view, the subject of my research was so wide that I learned how to focus on the main problems, making the difference between the essential and the details. Even more, we had to make assumptions concerning some unsolved problems, or to wait for some confirmations from NTT, and all this was never to stop or delay our schedule and rythm of work.

From the human point of view, I do believe that I had the best oportunity to discover Japan and its people. Being employed in a japanese company, and especially in a department where I was the only foreigner kept me all the time in contact with Japan.

Japanese companies are often seen as big families. When I arrived, I was greeted kindly and everybody tried to make me feel comfortable. In everyday life, the master word seems to be harmony, where everyone avoids any direct clash or remark. This creates a very good atmospher of work, where new people are very quickly integrated.

In fact, after six months in the kansai area, I feel I have some general ideas about the japanese behavior, although I would not dare saying that I understood everything about Japan. The only thing I am sure about is that I appreciated it a lot.

Bibliography

[1] Naoki Uchida, Akira Miura. "Customer-Defined Service Model and Definition Method for Intelligent Networks", ICC (1991)

Appendix A

NTT datas

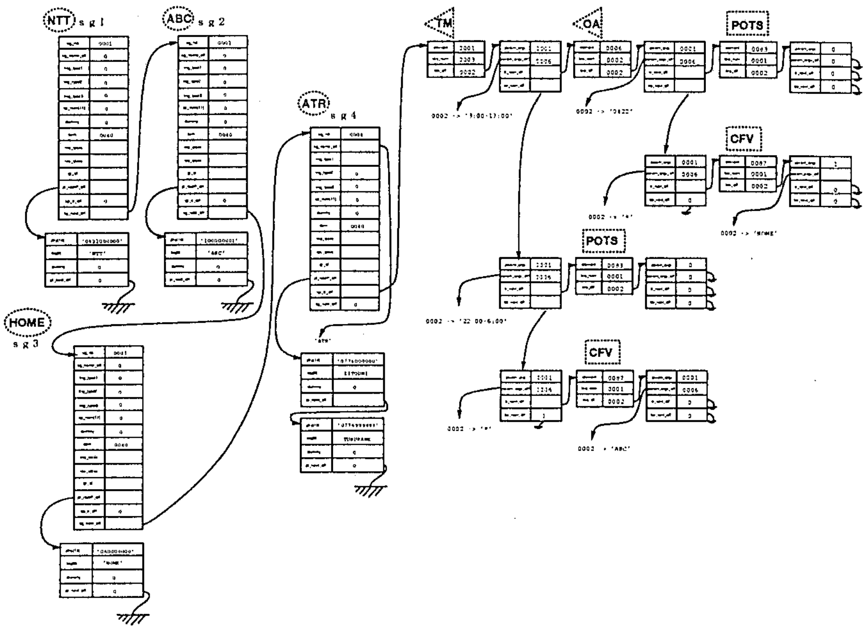


Figure A.1: Screening service definition data

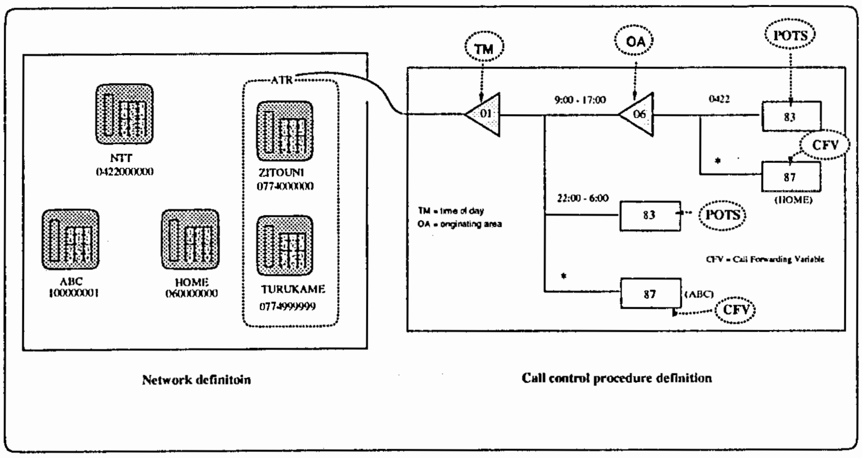


Figure A.2: Screening service definition

Appendix B

The Converter Outputs

B.1 The Auto Run file

```
      ###########################################
      ##     Auto run file for Screen          ##
      ###########################################

Service:
        /STR/rule/pots.str
        /STR/rule/cfv.str
        /STR/rule/scr.str
        /STR/rule/S000001.str

Table:

Terminal:
  :NTT:0422000000::
  :ABC:100000001::
  :HOME:060000000::
  :ZITOUNI:0774951247::m-t1(ZITOUNI),m-cfv1(ZITOUNI,HOME),
                                      m-cfv2(ZITOUNI,ABC)
  :TURUKAME:0774951247::m-t1(TURUKAME),m-cfv1(TURUKAME,HOME),
                                     m-cfv2(TURUKAME,ABC)
Auto:
```

B.2 The Screening Conditions file

```
#################################

          TYPE: TM
          VAR:  X1
          NUM: 3
          COND: 9:00-17:00,22:00-6:00,*
          INFO: 3,ABC

#################################

          TYPE: OA
          VAR:  X2
          NUM: 2
          COND: 0422,*
          INFO: 2,HOME

#################################
```

B.3 The Object Information file

```
###########################

          ID: 1
          OBJECT:  PTN
          NUM: 1
          PHY: 0422000000
          LOG: NTT

###########################

          ID: 2
          OBJECT:  PTN
          NUM: 1
          PHY: 100000001
          LOG: ABC
```

```
############################

          ID: 3
          OBJECT:  PTN
          NUM: 1
          PHY: 060000000
          LOG: HOME

############################

          ID: 4
          NAME: ATR
          TRIG_TYPE: 1
          OBJECT:  PTN
          NUM: 2
          PHY: 0774951291,0774951247
          LOG: ZITOUNI,TURUKAME
          TYPE_VAR: TM=X1 OA=X2
          INIT_INFO: t1,cfv1:HOME,cfv2:ABC

############################
```

B.4 The STR Rule file

```
          ###########################################
          ##          SCREEN(1) Service            ##
          ###########################################

Primitives:
        idle(A),dial-tone(A),ringing(A,B),ringback(A,B),pingring(A,B),
        m-t1(A),m-cfv1(A,B),m-cfv2(A,B),reject-dial(A,B)

Events:
        dial(A,B),t1(A,B),cfv1(A,B),cfv2(A,B)

Rules:
    cond:idle(A) t1(A): m-t1(A).
    cond:idle(B),cond:dial-tone(A),cond:m-t1(A) cfv1(A,B): m-cfv1(A,B).
```

```
cond:idle(B),cond:dial-tone(A),cond:m-t1(A) cfv2(A,B): m-cfv2(A,B).

dial-tone(A),idle(B),cond:m-t1(B),
equal[X1,1],equal[X2,1]
 dial(A,B):
Calling(A,B).

dial-tone(A),idle(B),idle(C),cond:m-t1(B),cond:m-cfv1(B,C),
equal[X1,1],equal[X2,2]
 dial(A,B):
Calling(A,C),pingring(B,A).

dial-tone(A),not[idle(B)],idle(C),cond:m-t1(B),cond:m-cfv1(B,C),
equal[X1,1],equal[X2,2]
 dial(A,B):
Calling(A,C).


dial-tone(A),cond:idle(B),not[idle(C)],cond:m-t1(B),cond:m-cfv1(B,C),
equal[X1,1],equal[X2,2]
 dial(A,B):
busy-dial(A,C).

dial-tone(A),not[idle(B)],not[idle(C)],cond:m-t1(B),cond:m-cfv1(B,C),
equal[X1,1],equal[X2,2]
 dial(A,B):
busy-dial(A,C).

dial-tone(A),idle(B),idle(C),cond:m-t1(B),cond:m-cfv2(B,C),
equal[X1,3]
 dial(A,B):
Calling(A,C),pingring(B,A).

dial-tone(A),not[idle(B)],idle(C),cond:m-t1(B),cond:m-cfv2(B,C),
equal[X1,3]
 dial(A,B):
Calling(A,C).

dial-tone(A),cond:idle(B),not[idle(C)],cond:m-t1(B),cond:m-cfv2(B,C),
equal[X1,3]
 dial(A,B):
busy-dial(A,C).
```

```
dial-tone(A),not[idle(B)],not[idle(C)],cond:m-t1(B),cond:m-cfv2(B,C),
equal[X1,3]
dial(A,B):
busy-dial(A,C).
```