Asbru Interpreter 1.0
A Short Introduction

Andreas Seyfang, Peter Votruba, Michael Paesold, Silvia Miksch
Institute for Software Engineering and Interactive Systems
Vienna University of Technology

April 12, 2005

This document gives a brief introduction into installing and using the interpreter. It describes the installation, the input formats, the output and the test data enclosed with this package.

1 Installation

1.1 Content of this Package

- The interpreter itself consisting of the following files.
  - The jar-file `AsbruInterpreter.jar` containing the program proper.
  - The batch-file `run-AsbruInterpreter.bat` to start the interpreter.
  - Two XSL-files `showall.xsl` and `showplans.xsl` in subdirectory `scripts`.
- DTD files for the project files and for Asbru. They are found in subdirectory `dtd`.
- Three examples in subdirectory `examples`. They are described in Section 4.
- Documentation consisting of this document and `README.txt`.

1.2 Prerequisites

The interpreter is based on Java 1.4. It must be installed prior to installing the interpreter. It is available at `java.com`.

1.3 Installation

Simply extract the archive in a directory of your choice. All files will be found in a subdirectory named `AsbruInterpreter` in this directory (and newly created subdirectories of that directory).

1.4 Starting the Interpreter

On the command line, type

`run-AsbruInterpreter` *project*

where *project* is the name of your project file without the ending `.project.xml`. The format of this file is described in section 2.1.

The batch file `run-AsbruInterpreter.bat` first starts the interpreter and then the XSL-translator included in the jar-file to translate the internal output format (which is XML-based) to the output described in Section 3.

2 Input Description

The interpreter takes the name of a project file (without ending `.project.xml`) as its argument on invocation. The project file refers to two types of input: Asbru plans and data. The format of Asbru plans is specified in deliverable D2.2a. The project file and the data input is described in the following.

2.1 Project File

The syntax of the project file is XML-based. The following attributes in this file are important for the user.

- Element `project`, attribute `name`: This name is the basis for the output files.
- Element `guideline`, attribute `file-name`: This is the name of the Asbru plan library.
- Element `guideline`, attribute `top-level-plan`: The name of the plan to be started on program start.
- Element `data`, attribute `file-name`: The name of the file containing the input data.
- Element `start`, attributes `date` and `time`: The start of the time line considered by the interpreter. This must be the valid time of the first input data point or earlier.
- Element `end`, attributes `date` and `time`: The end of the time line considered by the interpreter. This must be the valid time of the last input data point plus any waiting period thereafter or later. If this is chosen too restrictive, then some conditions may not be fulfilled (those which would be fulfilled if the last data point would be valid for a little longer). On the other hand, the time-window module (not used in Protocure) will generate new data without input during the idle time after the last data point.

2.2 Data Input

The format of Asbru plans is specified in deliverable D2.2a. The data is formated as table with TAB characters between columns. Each column specifies the input for one parameter, accept for the first two, which specify date and time of the measurement. The first line of the data files gives the parameter names.

3 Output Description

The interpreter generates an interal XML file from which two styles of HTML-formated output are generated: full and abbreviated ("plans only").

3.1 Output Format "Full"

In this output format, all information is given. This means that whenever a module generates an output (i.e., a new data point), a line in the table described here is generated.

The table has the following colums:

- The valid time of the data point, shown as date, time, and microsteps (after the slash).
- The class of the module which generated the output.
- Information to designate the instance of the module which generated the output. This depends on the module and typically shows the plan name, the role of the condition, or the parameter name.

- The category of Asbru code which depicted by the module. This is one of abstraction, condition, parameter, variable, plan, or plan-body.
- The value of the data point.

The format of the value strongly depends on the type of data point as detailed in the following.

Real numbers. These represent quantitative values in Asbru. `NaN` stands for invalid data points.

Integers. These represent qualitative values in Asbru. -1 stands for invalid data.

Episodes. Each data point (and thus line in the table) represents a composite message. First, two flags are shown if appropriate: `End Of Monitoring` means that this is the last output from the module sending it. `INVALID` shows that this data point is simply an invalid value (which is rarely used).

Next, a subtable is displayed which has the following columns. Each line describes one episode which matched the corresponding time annotation.

ID. This is the unique identification of the episode.

PF. The positive flank, i.e., the start of the interval of the episode.

NF. The negative flank, i.e., the end of the interval of the episode (or "/" if it is not yet known).

SV. The start of validity, i.e., the first time point, at which the episode is valid.

EV. The end of validity, i.e., the last time point, at which the episode is valid.

BFI. A flag indicating that the end of a before found interval is found now.

Plan Module Output. This consists of the plan state, optionally numeric output, and an array of synchronization flags for the subplans (children) of this plan. In the table, up to three items are displayed.

- The plan state is is always shown. Here, *synchronizing* is an internal state before *activated* in which the child awaits the suitable synchronization flag from the parent.
- The numerical output if given. This is displayed as a subtable in the same format as any other data point, since the numerical output is a data point inside the plan module output data point.
- A subtable for the synchronization flags sent to the children of this plan. In this subtable, each line shows the child index and the synchronization flag sent to this child. Lines containing the flag *nothing*, i.e., no synchronization flag is sent, are suppressed.

Under circumstances described elsewhere the interpreter starts playback of previous values for a group of modules. In this case, the valid time at which the playback starts is given together with a list of the modules involved. Then the output occuring during the playback is displayed in an intended subtable.

3.2 Output Format "Plans Only"

In this format, only output of plans is shown. This output is presented as described above for *Plan Module Output*.

3.3 Output Format XML

The interpreter originally generates an XML file which is then translated to the above formats using XSL scripts. The original XML output is of little use for the user. All relevant information is shows in the full format described above.

Condition	Parameter	Reference	ESS	LSS	EFS	LFS	MinDu	MaxDu
Filter	A > 3	now			0 min			
Setup	B = 2	now			0 min		5 min	
Suspend	A < 1	now			0 min			
Reactivate	B = 0	self	0 min		2 min			
Complete	A ≥ 5	now			0 min			
Abort	B > 10	now			0 min			

Table 1: Conditions in example *single plan*.

4 Demonstration Cases

This section describes several tests of the interpreter which are designed as demonstrations of its functionality at the same time.

- The first test (`single-plan`) demonstrates how a single plan traverses all its plan states.
- The second test (`parameter-proposition`) demonstrates the operation of the parameter proposition.
- The third test (`multiple-plans`) demonstrates the types of plans and the invokation of one plan by another.

The following sections describe each of them in detail.

Each example consists of seven files:

- *name*`.xml` contains the Asbru plan library.
- *name*`.txt` contains the input data.
- *name*`.project.xml` is the project file for this example.
- *name*`.bat` is a batch-file provided for easy running the example (just double-click it).
- *name*`-all.html` is the output in detailed format.
- *name*`-plansonly.html` is the list of plan state transitions.
- *name*`-logfile.xml` is the output in internal XML format.

4.1 Single Plan

This example demonstrates the differnt plan states showing a single plan T. Table 1 shows the details of the conditions of this plan. Table 2 shows the content of the input file together with the reactions of the interpreter. Note that not all lines in the table correspond to input. Instead, all noteworthy time points are shown in the table including those where something changes based on the elapse of time, not because of new input data.

4.2 Parameter Proposition

This example demonstrates more complex parameter propositions. Again, one plan T is used. Table 3 shows the details of the conditions of this plan. Table 4 shows the content of the input file together with the reactions of the interpreter.

Date	Time	A	B	Comment
01.01.2005	00:00:00	1	1	Plan T is started by the user and becomes considered
01.01.2005	00:00:20	2	2	
01.01.2005	00:01:00	4	2	The filter condition A $>$ 3 now is fulfilled
01.01.2005	00:05:20			The setup condition B = 2 for the previous 5 minutes is fulfilled
01.01.2005	00:06:00	0	0	The suspend condition A $<$ 1 now is fulfilled
01.01.2005	00:07:30	1	0	The suspend condition is no more fulfilled but the plan remains suspended
01.01.2005	00:08:00			The reactivate condition B = 0 ESS=0 min EFS=2 RP=self is fulfilled
01.01.2005	00:08:10	1	1	The reactivate condition is no more fulfilled, the plan remains activated
01.01.2005	00:09:00	5	1	The complete condition A $\geq$ 5 now is fulfilled

Table 2: Inputs in example *single plan* and reactions to the input.

Condition	Parameter	Reference	ESS	LSS	EFS	LFS	MinDu	MaxDu
Filter	B = 1	now			-10 min	-5 min	2 min	10 min
Setup	A $>$ 5	now	-10 min	-1 min	-1 min			
Suspend	A $>$ 10	now			0 min		1 min	
Reactivate	B = 1	self				20 min	1 min	2 min
Complete	A $<$ 5	self			5 min		10 min	
Abort	A $>$ 20	self	-10 min	10 min	0 min	10 min	10 min	15 min

Table 3: Conditions in example *parameter proposition*.

Date	Time	A	B	assumed reaction
01.01.2005	00:00:00	1	1	Plan T is started by the user and becomes considered
01.01.2005	00:10:00	6	0	
01.01.2005	00:11:00			The setup condition is fulfulled, but the filter condition is not yet
01.01.2005	00:15:00			The filter condition is fulfilled, the setup condition is already fulfilled, Plan activated
01.01.2005	00:16:00	12	0	
01.01.2005	00:17:00	11	0	The suspend condition is fulfilled (interval started at 00:16)
01.01.2005	00:20:00	6	1	
01.01.2005	00:22:30	25	0	The reactivate condition is not fulfilled because MaxDu is exceeded
01.01.2005	00:22:40	7	1	
01.01.2005	00:24:30	6	0	The reactivate condition is fulfilled (interval from 00:22:40 to 00:24:30)
01.01.2005	00:26:00	3	0	
01.01.2005	00:36:00	2	0	The complete condition is fulfilled (interval started at 00:26)

Table 4: Inputs in example *parameter proposition* and reactions to them.

4.3 Multiple Plans

Plan *A* is the top-level plan. It takes the following steps sequentially.

- Plan *C* is called.
- After this, the current value of parameter *P* is assigned to variable *V*.
- Then, a new value for *P* is requested.
- Finally plans *parallel*, *unordered*, and *any-order* are called.

Plan *C* is user-performed as well. It completes as soon as the value of parameter *P* lies between 30 and 40 for one minute during the activation of *C*, and it aborts if *P* lies between -30 and -40 with the same timing constraints. In addition, it is suspended if *P* is smaller than zero and reactivated when *P* is greater than zero.

Plan *D* again is user-performed. It is activated when parameter *P* equals zero. It is completed when parameter *P* lies between 40 and 50.

Plan *E* contains an *ask* statement for parameter *P* as its plan body. It is completed as soon as a new value is entered but no later than 10 minutes after its start.

Plan *parallel* calls plans *D* and *E*. Because of the semantics of the parallel plan in Asbru, plan *E* is not activated before plan *D* can be activated. Plan *parallel* is suspended when parameter *P* is below zero and reactivated when it is above zero. It is completed when parameter *P* is greater than 100 (and all subplans have completed).

Plan *unordered* is similar to plan *parallel*, accept that it is of type unordered. Therefore plan *E* is activated immediatelly.

Plan *any-order* calls *C* and *D*. In cases in which the filter condition of *D* is not fulfilled, *C* will start first. Later, *D* can only start if *C* is either suspended, completed, or aborted. If *D* starts while *C* is suspended, *C* is not reactivated before *D* is completed.

Table 5 summarizes the above described plans. Table 6 shows the input and the reactions to it in a test run.

Plan A	Sequential: C, V=P, ask P, parallel, unordered, any-order			
no conditions				

Plan C	user-performed			
Condition	Parameter	Reference	EFS	MinDu
Suspend	$P < 0$	now	0 min	
Reactivate	$P > 0$	now	0 min	
Complete	$30 \leq P \leq 40$	self	1 min	1 min
Abort	$-30 \leq P \leq -40$	self	1 min	1 min

Plan D	user-performed			
Condition	Parameter	Reference	EFS	MinDu
Filter	$P = 0$	now	0 min	
Complete	$40 \leq P \leq 50$	self	1 min	1 min

Plan E	ask P, timeout 10 min
no conditions	

Plan parallel	parallel D, E			
Condition	Parameter	Reference	EFS	MinDu
Suspend	$P < 0$	now	0 min	
Reactivate	$P > 0$	now	0 min	
Complete	$P > 100$	now	0 min	

Plan unordered	unordered D, E
no conditions	

Plan any-order	any-order C, D
no conditions	

Table 5: Plans and conditions in example *multiple plans*.

Date	Time	P	Comment
03.01.2005	03:03:00		A activated, C activated
03.01.2005	03:03:00	-3	C suspended (P<0 now)
03.01.2005	03:04:00	2	C reactivated (P>0 now)
03.01.2005	03:05:00	22	
03.01.2005	03:10:00	33	
03.01.2005	03:11:00		C completed ($30 \leq p \leq 40$ for 1 min), variable-assignment in A, start of ask in A
03.01.2005	03:21:00		ask timed out (10 min after start), parallel activated, D and E considered
03.01.2005	03:22:00	0	filter condition of D fulfilled, D and E activated
03.01.2005	03:24:00	-2	E completed, parallel (and thus D) suspended
03.01.2005	03:25:00	1	parallel and D resumed
03.01.2005	03:28:00	44	
03.01.2005	03:29:00		D completed
03.01.2005	03:30:00	3	
03.01.2005	03:30:30	111	parallel completed, unordered activated, D considered, E activated
03.01.2005	03:31:00	0	D activated, E completed
03.01.2005	03:34:00	41	
03.01.2005	03:35:00		D completed, unordered completed, any-order activated, C activated
03.01.2005	03:36:00	0	D waiting to be activated (synchronizing)
03.01.2005	03:37:00	-1	C suspended, D activated
03.01.2005	03:38:00	1	C waiting to be reactivated (synchronizing)
03.01.2005	03:39:00	35	
03.01.2005	03:40:00		D completed, C reactivated
03.01.2005	03:41:00	45	
03.01.2005	03:42:00		C completed, any-order completed, A completed

Table 6: Inputs in example *multiple plans* and reactions to them.