



**IST-FP6-508794**

**PROTOCURE II**

*Integrating formal methods in the development process of  
medical guidelines and protocols*

Specific Targeted Research Project  
Information Society Technologies

**Complementary material to deliverable  
D2.3b Asbru-to-KIV translator**

**Due date of deliverable:** 31 October 2004  
**Actual submission date:** 31 October 2004  
**Submission date of revision:** 23 September 2005

**Start date of project:** 1 January 2004      **Duration:** 30 months

**Organisation name of lead contractor:** Universitat Jaume I

**Revision 2**

Project co-funded by the European Commissions within the Sixth Framework Programme(2002-2006)		
Dissemination Level		
<b>PU</b>	Public	√
<b>PP</b>	Restricted to other programme participants (including the Commission Services)	
<b>RE</b>	Restricted to a group specified by the consortium (including the Commission Services)	
<b>CO</b>	Confidential, only for members of the consortium (including the Commission Services)	



# An Automatic Translator from Asbru to KIV

Andreas Seyfang, Jonathan Schmitt, Michael Balsler, Silvia Miksch, Wolfgang Reif

## Contents

<b>1</b>	<b>General Introduction</b>	<b>2</b>
1.1	Program Installation . . . . .	2
1.2	Program Usage . . . . .	2
<b>2</b>	<b>Introduction to Translation Details</b>	<b>3</b>
2.1	Overall Structure of Asbru Plan Library . . . . .	3
2.2	Overall Structure of KIV Specification . . . . .	3
2.3	Asbru name spaces . . . . .	4
2.4	Plan name generation for KIV . . . . .	4
2.5	Units and scales . . . . .	4
2.6	KIV operators and XML . . . . .	5
2.7	Insertion of Links between Asbru and KIV . . . . .	5
<b>3</b>	<b>Translating the Domain Definition</b>	<b>5</b>
3.1	Qualitative Scale Definition . . . . .	5
3.2	Context Definition . . . . .	6
3.3	Variable Definition . . . . .	7
3.4	Constant Definition . . . . .	7
3.5	Parameter definition . . . . .	7
3.5.1	Raw data definition . . . . .	9
3.5.2	Numeric calculation . . . . .	9
3.5.3	Logical combination . . . . .	9
3.5.4	Qualitative abstractions from quantitative values . . . . .	11
<b>4</b>	<b>Translating Plans</b>	<b>13</b>
4.1	Arguments . . . . .	13
4.2	Value Definition . . . . .	13
4.3	Intentions and Conditions . . . . .	13
4.4	Effects . . . . .	20
4.5	Return Values . . . . .	20
<b>5</b>	<b>Translating the Plan Body</b>	<b>20</b>
5.1	Continuation Specification . . . . .	20
5.2	Plan Activation . . . . .	21
5.3	Variable Assignment . . . . .	21
5.4	Context Change . . . . .	21
5.5	Ask Statements . . . . .	30
5.6	if-then-else . . . . .	30
5.7	User-performed Plan . . . . .	30

<b>6</b>	<b>Translating Basic Elements</b>	<b>33</b>
6.1	Expressions . . . . .	33
6.1.1	Numerical constants . . . . .	33
6.1.2	Qualitative constants . . . . .	33
6.1.3	Constants . . . . .	33
6.1.4	Variable references . . . . .	34
6.1.5	Parameter references . . . . .	34
6.1.6	Now . . . . .	34
6.1.7	Plan state changes . . . . .	34
6.2	Conditions . . . . .	34
6.2.1	Parameter propositions . . . . .	34
6.2.2	Time annotations . . . . .	34
<b>7</b>	<b>Change History and Future Development</b>	<b>37</b>
7.1	Improvements in Version 2.0 . . . . .	37
7.2	Potential for Further Improvements . . . . .	37
7.2.1	Extension of functionality . . . . .	37
7.2.2	Improving output quality . . . . .	37
7.2.3	KIV extensions . . . . .	38

## 1 General Introduction

The aim of task T2.3b of Protocure II is to create an automatic translation from Asbru Light to XML-KIV. This section describes the installation and usage of this program.

The remaining sections of this document describe how each element of Asbru Light is translated to KIV. The details of Asbru are described in the Asbru 7.3 Reference Manual [1] and it is assumed the reader is familiar with Asbru. The subsets of Asbru defined for the Protocure II project are described in deliverable D2.2a, which is available at [www.protocure.org](http://www.protocure.org).

The Document Exploration and Linking Tool with Addons (DELT/A) – formerly known as Guideline Markup Tool (GMT) – which is mentioned in several places in this document is described in detail in deliverable D2.3a.

### 1.1 Program Installation

The translator is written in Perl. Therefore, it requires that Perl is available on the computer on which the translator is started. Linux distributions include Perl. For Windows, the recommended binary distribution is the one supplied for free by ActiveState ([www.activestate.com](http://www.activestate.com)). The program was developed using Perl version 5.8.4. It does not use any new libraries and should run on much older versions of Perl without problems.

To install Perl from ActiveState, download and double-click the installation package (12 MB). The installation is automatic after answering the usual questions but does not work if the computer is not connected to the internet during installation. Issues like changing the PATH variable are handled by the installation program.

The translator program consists of the main program `asbru2kiv.pl` and the module `XMLTree.pm` which must be stored in a subdirectory `MyXML` under the directory containing `asbru2kiv.pl`.

### 1.2 Program Usage

The translation takes one input file – the Asbru file to be translated – and it produces two output files – the KIV translation and a new version of the Asbru file in which `gmt-links` have been inserted to document the connection between elements in this file and in the KIV file.

The syntax of the KIV file is XML-KIV, an XML version of KIV created for the Procure II project to allow the creation of the mentioned gmt-links and usage of the viewing and editing tools created in this project.

The translation program takes two arguments: The name of the original Asbru file and the name of the KIV file to produce. The name of the adapted Asbru file is created by appending `-linked` to the original filename (before the extension).

On Linux systems and on Windows systems on which the filetype `.pl` has been associated with Perl by the ActiveState installation procedure, the translation is started as

```
asbru2kiv.pl asbru-input.xml kiv-output.xml
```

In all other cases, the translation is started as

```
perl asbru2kiv.pl asbru-input.xml kiv-output.xml
```

After this command, `asbru-input-linked.xml` will contain an Asbru file linked to `kiv-output.xml` by numerous gmt-links. This output can be viewed using the Document Exploration and Linking Tool with Addons (DELT/A).

## 2 Introduction to Translation Details

This section describes the overall structure of the Asbru plan library and the XML-KIV specification file. In addition, it explains some basic constructs which are used in many places and their translation.

### 2.1 Overall Structure of Asbru Plan Library

The Asbru Light plan library consists of the following parts which are described in detail below:

**Library information** show author, date and change history of this plan library. This information is not translated to KIV, but it can be stored within the comment tags of the KIV XML Format.

**Domain definitions** contain (qualitative) type definitions, context definitions, grouped parameter definitions, value definitions. Asbru Light uses exactly one domain definition per plan library.

**Plans** are grouped in plan groups and describe the procedural aspects of the guideline.

### 2.2 Overall Structure of KIV Specification

The KIV system uses text files each containing the specification of one Asbru plan. In addition, there may be some files specifying the data abstraction rules.

To improve the utilization of the Document Exploration and Linking Tool with Addons (DELT/A) (described in deliverable D2.3a), XML-KIV has been created. One file of XML-KIV contains an XML representation of all the specification files necessary to implement the modelled guideline.

The specifications in the KIV file are ordered as follows. Within each group, specifications are ordered alphabetically by their name.

- Constant definitions.
- Context variables, except Boolean ones.
- Parameter definitions.
- Qualitative scales.
- The "Asbru Basic Patient Data" specification and the "Asbru Data" specification which contain standard definitions needed the access to parameters and variables.
- The specification "Plans", which contains variables, Boolean constants, and all plans.

Entity	Prefix
variable	variable
parameter	parameter
context variable (name)	contextname
plan	plan
constant	constant
qualitative scale	scale
argument	argument
return value	returnvalue

Table 1: Prefixes for data items and plans. For local variables, arguments, and return values, the name of the plan containing them is inserted between prefix and the name proper.

Subplan relation	Postfix
plan steps in subplans	-1, -2, etc.
on-abort, on-suspend, plan-schema children	-A, -S, -P
argument and return value assignment for plan-activation	-A, -R
plan-body or single-step if used with arguments and/or return values	-B
yes and no-branch in if-then-else	-Y, -N

  

Parameter relation	Postfix
source in qualitative-parameter-def	-S
arguments in calculation	-1, -2, etc.

Table 2: Postfixes for plan and parameter names.

Each specification has a name, which is given as an attribute within the `spec1` tag. Higher level specifications that rely on other specifications have to announce that by putting the names of all used specifications within the `usedname` tags.

### 2.3 Asbru name spaces

Asbru features different name spaces for variables, parameters, constants, context variables and plans. KIV features one global name space. Therefore, each of the above items receives a prefix in the translation process. They are detailed in Table 1. Between the prefix and the Asbru name of the variable etc., a hyphen is inserted.

### 2.4 Plan name generation for KIV

Since KIV does not support nesting in the plan body, an Asbru plan generally maps to a set of KIV plans. The names of the Asbru plans are extended by the postfixes shown in Table 2.

### 2.5 Units and scales

Asbru Light supports only one scale for each unit class. Table 3 shows the supported units. These units may be used in the `unit` attribute in `numerical-constant`. The unit classes are used in the `type` attribute for variable, constant or parameter definitions.

Date and time are treated the same in KIV, they are just temporal distances between a time point and the zero time point.

The values are represented as integers of the smallest unit. This means that the precision is 1 item of the smallest unit (e.g., 1 millisecond for time and date).

Unit Class	Scale	Units
length	metric	mm, cm, m, km
area	metric	mm2, cm2, m2, km2, qm, a, ha
volume	metric	mm3, cm3, m3, km3, ml, cl, dl, hl
mass	metric	mg, g, kg, to
time	normal	ms, s, min, h, d, w, mon, y
date	normal	ms, s, min, h, d, w, mon, y
temperature	Celsius	C
amount	normal	<i>no unit specified</i>
energy	calories	cal, kcal
pressure	bar	mbar, bar
parts	fraction	ppm, ppb, %
currency	EU	Euro, cent

Table 3: Units implemented in Asbru Light.

## 2.6 KIV operators and XML

KIV code is encoded as `PCDATA` in XML. This would prohibit the usage of `<`. To allow the usage of this character, each KIV statement containing conditions is encapsulated in `<![CDATA[...]]>`. In this document, however *this encapsulation is removed from the KIV code shown in examples* to enhance readability.

Table 4 shows the mapping of various operators to KIV. Those preceded by a backslash are replaced by special characters in KIV.

## 2.7 Insertion of Links between Asbru and KIV

To document the connection between the guideline parts in both notations, XML elements named `gmt-link` are inserted by the translator both into the original Asbru file which is translated and the resulting XML-KIV file. Each `gmt-link` has an identification number which designates which parts of Asbru code result in which part of KIV code.

The Guideline Markup Tool used in each step of the guideline modelling process allows the easy browsing using these links. In addition, a simple translation of a pair of XML file with `gmt-links` to a group of HTML files which support similar browsing was implemented.

Links between the Asbru file and the MBH file on which the Asbru file is based are preserved. The first ID number used by the translator is greater the highest ID found in the original Asbru file.

# 3 Translating the Domain Definition

## 3.1 Qualitative Scale Definition

This Asbru element defines a qualitative data type. The example in Figure 1 defines that variables and parameters of data type fever take one of the three symbolic values no-fever, moderate-fever, high-fever.

There are two usages of qualitative data types. First, they can be assigned and used for themselves alone. Second, they can be abstracted from quantitative values. This abstraction is defined by parameter definition as shown below.

Figure 1 shows a simple example for a qualitative scale in Asbru. Figure 2 shows its translation to KIV. The name of the `spec1` element is the same as the KIV name for the data type, i.e., the Asbru name plus the prefix. The `datasortdef` lists the values a variable or parameter of this type can take. The `vardef` creates some variables (3-4 for each type) which are needed in the verification process but not directly linked to Asbru code.

Asbru operator	KIV infix operator
add	+
subtract	-
multiply	*
divide	/
less-than	<
less-or-equal	\le
greater-than	>
greater-or-equal	\ge
equal	=
not-equal	neq
and	and
or	or
xor	xor

  

Asbru operator	KIV predefined function
not	not
minimum	min
maximum	max
absolute-value	abs
sign	sgn
root	sqrt
power	pow
modulo	mod

  

Symbol	KIV representation
⊢	-
↦	->
×	x

Table 4: Operators in KIV

### 3.2 Context Definition

A context is used together with the parameter name, value description and time annotation in the parameter proposition which is the fundamental part of the conditions in Asbru plans. The context definition itself – like the qualitative scale definition – only specifies the existence of a context and the values it can take. There can be several independent contexts in a plan library. A context can be Boolean, i.e., it is either given or not, or it can take several different values.

Figure 3 shows two context definitions in Asbru. First, the Boolean context life-threatening-situation is defined by the one-line context-def. Second, the context mode-of-ventilation can take the values ippv, imv, or cpap according to the second context-def.

The translation of the definition of the Boolean context does not leave traces in KIV. The context variable itself is stored in an aggregation which does not need initialization for each item it contains. And the definition does not introduce a new data type – Booleans are a built-in data type in KIV.

```

<qualitative-scale-def name="fever">
  <gmt-link link-id="1" />
  <qualitative-entry entry="no-fever"/>
  <qualitative-entry entry="moderate-fever"/>
  <qualitative-entry entry="high-fever"/>
</qualitative-scale-def>

```

Figure 1: Definition of a qualitative scale in Asbru.

```

<spec1 name="scale-fever">
  <basicdataspec>
    <gmt-link link-id="1" />
    <datasortdef>
      scale-fever = no-fever | moderate-fever | high-fever
    </datasortdef>
    <vardef>
      scale-fever, scale-fever1, scale-fever2 : scale-fever
    </vardef>
    <vardef>
      Scale-fever, Scale-fever1, Scale-fever2 : scale-fever flexible
    </vardef>
  </basicdataspec>
</spec1>

```

Figure 2: Definition of a qualitative scale in KIV.

```

<context-def name="life-threatening-situation">
  <gmt-link link-id="5"/>
</context-def>
<context-def name="mode-of-ventilation">
  <gmt-link link-id="6"/>
  <qualitative-entry entry="ippv"/>
  <qualitative-entry entry="imv"/>
  <qualitative-entry entry="cpap"/>
</context-def>

```

Figure 3: Sample context definition in Asbru.

Figure 4 shows the translation of the context *mode-of-ventilation*. It constitutes a new qualitative data type for which a group of variables are created.

### 3.3 Variable Definition

Each variable to be used in the plan library is defined in a variable definition. Global variables are defined in the domain definition. Local variables are defined in plans. KIV stores all variable in a global variable aggregation. For this, there are no need to declare each variable. Therefore the variable definitions found in Asbru are not mapped to KIV. Instead, the translator keeps track of the type of each variable.

### 3.4 Constant Definition

Asbru allows the definition of symbolic constants and so does KIV. Figure 5 shows a sample definition of a constant in Asbru. Figure 6 shows its translation to KIV. The KIV `constdef` only states that the value of C is constant. The lemma defines the value which is given as an integer measured in milligramm (which is the smallest unit for mass in this implementation).

### 3.5 Parameter definition

Logically, the Asbru plan library consists of a declarative and a procedural part. While the procedural part is represented by the plans, the declarative one is implemented via parameter definitions. Parameters contain values which are time stamped, i.e., for each value the time at which it was measured is known.

```

<spec1 name="contextname-mode-of-ventilation">
  <basicdataspec>
    <gmt-link link-id="6"/>
    <datasortdef>
      contextname-mode-of-ventilation = ippv | imv | cpap
    </datasortdef>
    <vardef>contextname-mode-of-ventilation,
            contextname-mode-of-ventilation1,
            contextname-mode-of-ventilation2
            : contextname-mode-of-ventilation
    </vardef>
    <vardef>Contextname-mode-of-ventilation,
            Contextname-mode-of-ventilation1,
            Contextname-mode-of-ventilation2
            : contextname-mode-of-ventilation flexible
    </vardef>
  </basicdataspec>
</spec1>

```

Figure 4: Sample definition of a context with qualitative values in KIV.

```

<constant-def name="C" type="mass">
  <gmt-link link-id="46"/>
  <numerical-constant unit="kg" value="3"/>
</constant-def>

```

Figure 5: Sample definition of a global and a local variable in Asbru.

```

<spec1 name="constant-some-constant">
  <enrichedspec>
    <usedname>
      Asbru
    </usedname>
    <specpart>
      <signature>
        <gmt-link link-id="46" />
        <constdef>
          constant-C : int
        </constdef>
      </signature>
      <lemma>
        |- constant-C = 3000000
      </lemma>
    </specpart>
  </enrichedspec>
</spec1>

```

Figure 6: Sample definition of a constant in KIV.

```

<parameter-def required="no" name="sum-of-A-and-B" type="length">
  <calculation-def operator="add">
    <parameter-ref name="A"/>
    <parameter-ref name="B"/>
  </calculation-def>
</parameter-def>

```

Figure 7: Definition of a numeric abstraction in Asbru.

```

<parameter-proposition parameter-name="sum-of-A-and-B">
  <gmt-link link-id="72"/>
  <value-description type="less-or-equal">
    <numerical-constant unit="m" value="4"/>
  </value-description>
  ...
</parameter-proposition>

```

Figure 8: Usage of the abstracted parameter in Asbru.

These values are accessed using parameter propositions (or simple parameter references) as described in Section 6.

The parameters together build a directed graph of abstractions which transform raw data input by the user into high-level concepts used in the guideline. While full Asbru has a rich set of abstractions, Asbru Light is limited to the abstractions described in the following subsections.

### 3.5.1 Raw data definition

Each raw data definition defines one input channel. For Asbru Light, input always comes from the user and never from devices such as monitors in an intensive care unit. This does not imply that input is interactive in any setting. For testing purposes user input is stored in files and read from there. Still, there is a difference in the nature of the data, and it is this difference which is described by the attribute `mode` in Asbru element `raw-data-def`.

The raw data definition is not directly translated to KIV but supplies the type of parameters later used in various places.

### 3.5.2 Numeric calculation

The element `calculation-def` defines a quantitative abstraction based on other quantitative parameters, e.g., the sum. Again, this is not directly mapped to KIV. Instead, each reference to the parameter's value is replaced by a computation of each current value.

Figure 7 shows the definition of the parameter `sum-of-A-and-B` in Asbru. Figure 8 shows the usage of this abstracted parameter in a parameter proposition. Only part of the parameter proposition is shown – it is described in detail in section 6.2.1.

Figure 9 shows the translation of the parameter proposition to KIV. Details of the syntax are given in section 6.2.1, here we focus on the fact that the current values of A and B are added instead of referencing `sum-of-A-and-B` which in fact cannot be found in the KIV translation.

All operators used in the Asbru element `calculation-def` are implemented in KIV.

### 3.5.3 Logical combination

Boolean parameters can be true, false, or undefined in Asbru. The element `logical-combination-def` implements the Boolean abstractions `and`, `or`, `xor`, `not`.

```

<parameter-proposition>
  <gmt-link link-id="72" />
  ...
  <simple-constraint>
    lambda pdh, vh, ash, as, ac. pdh[ac][ 'parameter-A' ] .val
                                + pdh[ac][ 'parameter-B' ] .val
      \le 4000
  </simple-constraint>
</parameter-proposition>

```

Figure 9: Direct calculation of the specified abstraction for each usage in KIV.

```

<parameter-def name="combination-example" type="Boolean">
  <logical-combination-def operator="or">
    <comparison-def operator="greater-than">
      <left-hand-parameter>
        <parameter-ref name="A" />
      </left-hand-parameter>
      <right-hand-parameter>
        <numerical-constant unit="m" value="3" />
      </right-hand-parameter>
    </comparison-def>
    <parameter-ref name="B-bool" />
  </logical-combination-def>
</parameter-def>

```

Figure 10: Definition of a logical abstraction in Asbru.

KIV does not explicitly support three valued logic. However, because of partially defined functions and constants, it may be not possible to evaluate a proposition, if a variable or parameter has an unknown value.

Similar to the above case, the abstracted parameter is replaced by its components in the translation process.

Figure 10 shows the definition of the parameter in Asbru. Figure 11 shows the usage of this abstracted parameter in a parameter proposition. Only part of the parameter proposition is shown – it is described in detail in section 6.2.1.

Figure 12 shows the translation of the parameter proposition to KIV. Details of the syntax are given in section 6.2.1, here we focus on the fact that the definition of the logical combination is inserted into the parameter proposition instead of referencing the parameter's name.

A later version of the translator may remove the unnecessary comparison of the Boolean parameter with the constant true.

```

<parameter-proposition parameter-name="combination-example">
  <gmt-link link-id="73" />
  <value-description type="equal">
    <qualitative-constant value="true" />
  </value-description>
  ...
</parameter-proposition>

```

Figure 11: Usage of the abstracted parameter in Asbru.

```

<parameter-proposition>
  <gmt-link link-id="72" />
  ...
  <simple-constraint>
    lambda pdh, vh, ash, as, ac. pdh[ac]['parameter-A'] .val > 3000
                                or pdh[ac]['parameter-B-bool'] .val = true
  </simple-constraint>
</parameter-proposition>

```

Figure 12: Direct insertion of the specified logical abstraction for each usage in KIV.

```

<parameter-def required="yes" name="fever-qualitative" type="fever">
  <qualitative-parameter-def use-as-context="no">
    <gmt-link link-id="19" />
    <limits>
      <context>
        <any />
      </context>
      <negative-infinite />
      <limit-entry value="37" include-limit-value="yes">
        <gmt-link link-id="16" />
      </limit-entry>
      <limit-entry value="38" include-limit-value="yes">
        <gmt-link link-id="17" />
      </limit-entry>
      <positive-infinite>
        <gmt-link link-id="18" />
      </positive-infinite>
    </limits>
    <source>
      <parameter-ref name="fever-quantitative" />
    </source>
  </qualitative-parameter-def>
</parameter-def>

```

Figure 13: Simple qualitative abstraction in Asbru.

### 3.5.4 Qualitative abstractions from quantitative values

The `qualitative-parameter-def` is one of the important stones in the bridge between raw data and medical concepts. Based on the definition of a qualitative scale, it defines context dependent limits to map numerical input to qualitative output.

Figure 13 shows the definition of `fever-qualitative` based on `fever-quantitative` and the qualitative scale `fever`. The definition of the latter is shown in Figure 1. Figure 14 shows the translation of the mapping in KIV. This abstraction is performed in KIV by functions. Those functions are to be seen as mathematical functions rather than functions in computer programming.

The mapping between numeric (quantitative) and qualitative values can be context-specific in Asbru. I.e., for different contexts, different limits are used. In these cases, the context is integrated into the mapping function. Figures show a condensed and abstracted example for a complex multi-dimensional mapping table. A `numeric-input` is abstracted to a `qualitative-result` which has one of the two values `high` and `low`. The numerical limit between the region of high and low values depends on two different context dimensions: `sex` and `age-category`.

```

<spec1 name="parameter-fever-qualitative">
  <enrichedspec>
    <usedname>
      Asbru
    </usedname>
    <usedname>
      fever-quantitative
    </usedname>
    <specpart>
      <signature>
        <fctdef>
          <gmt-link link-id="19" />
          map-parameter-fever-qualitative
            : int -> parameter-fever-qualitative
        </fctdef>
      </signature>
      <lemma>
        <gmt-link link-id="16" />
        parameter-fever-qualitative-1-1 : |-
          a \le 37 -> map-parameter-fever-qualitative(a) = no-fever;
      </lemma>
      <lemma>
        <gmt-link link-id="17" />
        parameter-fever-qualitative-1-2 : |-
          a > 37 and a \le 38 -> map-parameter-fever-qualitative(a)
            = moderate-fever;
      </lemma>
      <lemma>
        <gmt-link link-id="18" />
        parameter-fever-qualitative-1-3 : |-
          a > 38 -> map-parameter-fever-qualitative(a) = high-fever;
      </lemma>
    </specpart>
  </enrichedspec>
</spec1>

```

Figure 14: Simple qualitative abstraction in KIV.

	adult	child	unknown
male	70	20	
female	60		

Table 5: Context-dependent limits for the mapping of numeric-input to qualitative-result (compare Figures 15 and 16).

Table 5 shows the values the two context variables can take and the limit between high and low result values for different contexts. Figure 15 shows the resulting definitions in Asbru. Figure 16 shows their translation to KIV.

## 4 Translating Plans

Figure 17 gives an overview of the parts of an Asbru plan. They are discussed one by one in this section. Asbru plans are modelled using `asbru-plan` in `specpart` as shown in Figure 18.

An Asbru plan consists of the following parts.

**Explanation** which is displayed to the user when the plan is activated. They are not translated to KIV since they are not relevant in the verification process.

**Arguments** which are passed to the plan on invocation.

**Value definitions** for the local variables.

**Intentions** which lie behind the execution of the plan.

**Conditions** which control the change of the plan-states.

**Effects** which describe assumptions about the environment which hold during execution of this plan.

**Plan-body** which defines the actual actions which are performed when executing this plan.

**Return value(s)** which the plan returns to the invoking plan.

### 4.1 Arguments

For technical reasons, KIV does not support arguments in the Asbru/KIV specifications. Arguments are modeled as global variables prefixed by the word *argument* and the name of the plan to which they belong. See Section 5.2 for an example.

### 4.2 Value Definition

Since KIV stores all variables in a global variable aggregation and definitions for the items in this aggregation are not needed, the definition of variables in Asbru does not have a mapping in KIV.

The name of local variables is prefixed with the name of the plan they belong to (in addition to the word *variable* to distinguish local variables of different plans which happen to have the same name).

### 4.3 Intentions and Conditions

Asbru intentions and conditions share many elements. Therefore they are discussed together here. To illustrate these similarities, Figure 19 shows the Asbru version an intention stating that a body weight of less than 50 kg should be achieved by this plan within one week after its start. Figure 20 shows its translation to KIV. Figures 21 and 22 show the same example as a complete condition. Note however, that in practice a plan will not contain such similarities between the intentions and condition, since this would make the intentions superfluous.

See Section 6.2 for a more detailed discussion of the parts of conditions (and intentions).

```

<parameter-def name="qualitative-result" type="low-high">
  <qualitative-parameter-def>
    <gmt-link link-id="15" />
    <limits>
      <context>
        <context-combination operator="and">
          <one-of name="sex">
            <value-ref name="female" />
          </one-of>
          <one-of name="age-category">
            <value-ref name="adult" />
          </one-of>
        </context-combination>
      </context>
      <negative-infinite />
      <limit-entry value="60"><gmt-link link-id="9" /></limit-entry>
      <positive-infinite><gmt-link link-id="10" /></positive-infinite>
    </limits>
    <limits>
      <context>
        <context-combination operator="and">
          <one-of name="sex">
            <value-ref name="male" />
          </one-of>
          <one-of name="age-category">
            <value-ref name="adult" />
          </one-of>
        </context-combination>
      </context>
      <negative-infinite />
      <limit-entry value="70">
        <gmt-link link-id="11" />
      </limit-entry>
      <positive-infinite>
        <gmt-link link-id="12" />
      </positive-infinite>
    </limits>
    <limits>
      <context>
        <one-of name="age-category">
          <value-ref name="child" />
          <value-ref name="unknown" />
        </one-of>
      </context>
      <negative-infinite />
      <limit-entry value="20"><gmt-link link-id="13" /></limit-entry>
      <positive-infinite><gmt-link link-id="14" /></positive-infinite>
    </limits>
    <source>
      <parameter-ref name="numeric-input" />
    </source>
  </qualitative-parameter-def>
</parameter-def>

```

Figure 15: Asbru definition of the abstraction from numeric-input to qualitative-result as described in table 5.

```

<spec1 name="parameter-qualitative-result">
  <enrichedspec>
    <usedname>Asbru</usedname>
    <usedname>numeric-input</usedname>
    <usedname>contextname-sex</usedname>
    <usedname>contextname-age-category</usedname>
    <specpart>
      <signature>
        <fctdef>
          <gmt-link link-id="15" />
          map-parameter-qualitative-result :
            int x contextname-sex x contextname-age-category
            -> parameter-qualitative-result
        </fctdef>
      </signature>
      <lemma>
        <gmt-link link-id="9" />
        parameter-qualitative-result-1-1 : |- a \le 60 and
          ((b = female) and (c = adult))
          -> map-parameter-qualitative-result(a, b, c) = low;
      </lemma><lemma>
        <gmt-link link-id="10" />
        parameter-qualitative-result-1-2 : |- a > 60 and
          ((b = female) and (c = adult))
          -> map-parameter-qualitative-result(a, b, c) = high;
      </lemma><lemma>
        <gmt-link link-id="11" />
        parameter-qualitative-result-2-1 : |- a \le 70
          and ((b = male) and (c = adult))
          -> map-parameter-qualitative-result(a, b, c) = low;
      </lemma><lemma>
        <gmt-link link-id="12" />
        parameter-qualitative-result-2-2 : |- a > 70
          and ((b = male) and (c = adult))
          -> map-parameter-qualitative-result(a, b, c) = high;
      </lemma><lemma>
        <gmt-link link-id="13" />
        parameter-qualitative-result-3-1 : |- a \le 20
          and (c = child or c = unknown)
          -> map-parameter-qualitative-result(a, b, c) = low;
      </lemma><lemma>
        <gmt-link link-id="14" />
        parameter-qualitative-result-3-2 : |- a > 20
          and (c = child or c = unknown)
          -> map-parameter-qualitative-result(a, b, c) = high;
      </lemma>
    </specpart>
  </enrichedspec>
</spec1>

```

Figure 16: KIV version of the mapping described in Figure 15.

```

<plan-library>
  ...
  <plans>
    <plan-group>
      <plan name="plan-A">
        <explanation
          text="Text to be displayed when this plan is executed."/>
        <intentions>
          <intention type="intermediate-state" verb="avoid">
            ...
          </intention>
          ...
        </intentions>
        <conditions>
          <filter-precondition ... >
            ...
          </filter-precondition>
          <setup-precondition ... >
            ...
          </setup-precondition>
          <suspend-condition ... >
            ...
          </suspend-condition>
          <reactivate-condition ... >
            ...
          </reactivate-condition>
          <complete-condition ... >
            ...
          </complete-condition>
          <abort-condition ... >
            ...
          </abort-condition>
        </conditions>
        <effects>
          ...
        </effects>
        <plan-body>
          ...
        </plan-body>
      </plan>
    </plan-group>
  </plans>
</plan-library>

```

Figure 17: Structure of an Asbru plan.

```

<specs>
  <spec1 name="Plans">
    <enrichedspec>
      <usedname>
        Asbru
      </usedname>
      <usedname>
        ... names of all specifications used by plans ...
      </usedname>
      <specpart>
        <signature>
          ...
          <asbru-plan lemmaname="plan-A" planname="plan-A">
            <control type="sequential">
              <waitfor>
                ...
              </waitfor>
            </control>
            <conditions>
              <filter-precondition ... >
                ...
              </filter-precondition>
              <setup-precondition ... >
                ...
              </setup-precondition>
              <suspend-condition ... >
                ...
              </suspend-condition>
              <reactivate-condition ... >
                ...
              </reactivate-condition>
              <complete-condition ... >
                ...
              </complete-condition>
              <abort-condition ... >
                ...
              </abort-condition>
            </conditions>
            <intentions>
              <intention ...>
                ...
              </intention>
            </intentions>
          </asbru-plan>
          ...
        </signature>
      </specpart>
    </enrichedspec>
  </spec1>
</specs>

```

Figure 18: Overview of XML-KIV file. Asbru plans are contained in `asbru-plan` elements in KIV

```

<intention verb="achieve" type="intermediate-state">
  <parameter-proposition parameter-name="body-weight">
    <gmt-link link-id="53" />
    <value-description type="less-than">
      <numerical-constant unit="kg" value="50" />
    </value-description>
    <context>
      <any />
    </context>
    <time-annotation>
      <time-range>
        <starting-shift>
          <latest>
            <numerical-constant unit="w" value="1" />
          </latest>
        </starting-shift>
      </time-range>
      <self />
    </time-annotation>
  </parameter-proposition>
</intention>

```

Figure 19: Intention of a plan in Asbru.

```

<intention type="intermediate-state" verb="achieve">
  <gmt-link link-id="52" />
  <parameter-proposition>
    <gmt-link link-id="53" />
    <time-annotation>
      <lss>week(1)</lss>
      <referencePoint>leave(plan-A, possible)</referencePoint>
    </time-annotation>
    <simple-constraint>
      lambda pdh, vh, ash, as, ac.
        pdh[ac]['parameter-body-weight'] .val < 50000000
    </simple-constraint>
  </parameter-proposition>
</intention>

```

Figure 20: Intention of a plan in KIV.

```

<complete-condition>
  <parameter-proposition parameter-name="body-weight">
    <gmt-link link-id="75" />
    <value-description type="less-than">
      <numerical-constant unit="kg" value="50" />
    </value-description>
    <context>
      <any />
    </context>
    <time-annotation>
      <time-range>
        <starting-shift>
          <latest>
            <numerical-constant unit="w" value="1" />
          </latest>
        </starting-shift>
      </time-range>
      <self />
    </time-annotation>
  </parameter-proposition>
</complete-condition>

```

Figure 21: Sample complete condition in Asbru.

```

<complete-condition overridable="false" confirmation_required="false">
  <parameter-proposition>
    <gmt-link link-id="75" />
    <time-annotation>
      <lss>week(1)</lss>
      <referencePoint>leave(plan-D, possible)</referencePoint>
    </time-annotation>
    <simple-constraint>
      lambda pdh, vh, ash, as, ac.
        pdh[ac]['parameter-body-weight'] .val < 5000000
    </simple-constraint>
  </parameter-proposition>
</complete-condition>

```

Figure 22: Sample complete condition in KIV.

## 4.4 Effects

Asbru effects come in two varieties: argument-dependency and plan-effect. Asbru Light supports only plan-effect.

Effects are modelled in KIV as first order formulas in combination with a special time annotation. The time annotation allows for several reference points and can therefore state, that the effect will start to influence the patient some time after plan start but will not stop his influence before the plan has ended. The results of an effect can be stated qualitative or quantitative. That is, an effect can claim that a certain value drops or raises (at an unspecified rate). It is also possible to express that the value drops by a certain amount, for example five units. As a combination it is allowed to specify an interval of dropping, that will take place. For example it can be stated, that the drop is at least five, at most seven units.

## 4.5 Return Values

Asbru supports multiple, named return values per plan. This is translated to KIV the same way as arguments. There is no direct support for return values in KIV, but properly named variables can be used as a workaround. To distinguish them from other data stored in the global variable aggregation, they are prefixed with the word *returnvalue* and the name of the plan which returns them.

# 5 Translating the Plan Body

In Asbru Light, the following cases concerning the plan-body can be distinguished:

- The plan consists of set of plan steps. These can be a mixture of
  - plan activations,
  - variable assignments,
  - context changes,
  - ask-statements, or
  - if-then-else statements.
- The plan is user-performed.

If the plan consist of several subplans, it also contains a continuation specification. The following subsections discuss each of these items one by one.

KIV does not support nesting of subplan groups and certain types of plan step are represented as separate plans. Therefore, each Asbru plan is translated to a group of `asbru-plan` elements in KIV. Their names are formed by appending various postfixes to the name of the Asbru plan as detailed in Table 2.

## 5.1 Continuation Specification

The continuation specification lists those subplans which must be completed in order to let the parent plan complete.

It can be one of the following:

- A combination of plan names. These are connected by the Boolean operators *and* or *or*. While in Asbru the number of plans in this list is not limited, KIV only defines these relation als binary relation. Therefore, the translator introduces additional groups of combination as shown in Figures 23 and 24.
- A number of subplans which must complete before the parent plan may complete. This can directly be mapped to KIV as shown in Figures 25 and 26.
- one which specifies that only one plan must finish before the parent plan can finish. This is a special case of the cardinality constraint.

```

<wait-for>
  <gmt-link link-id="63"/>
  <wait-for-group type="and">
    <static-plan-pointer plan-name="A"/>
    <static-plan-pointer plan-name="C"/>
    <static-plan-pointer plan-name="F"/>
    <static-plan-pointer plan-name="G"/>
  </wait-for-group>
</wait-for>

```

Figure 23: Continuation specification *wait for A and C and F and G* in Asbru.

- `all` which specifies that all subplans must complete. This is also implemented as a special case of the cardinality constraint by specifying the total count of subplans as shown in Figure 27.
- `none` which is implemented in KIV by leaving the `waitfor` element empty.

## 5.2 Plan Activation

The Asbru plan-activation specifies a subplan to be invoked, plus optionally a plan to be invoked if the original subplan is aborted and another one (also optionally) for the case that the original subplan is suspended. KIV represents these combination in plans of type `onabort`, `onsuspend`, and `onsuspendabort`. This requires the introduction of a subplan to implement each plan-activation step.

Figure 28 shows the first part of an Asbru plan A. Its first step is the activation of subplan F. If the latter aborts, plan X is activated. Figure 28 shows the start of plan A together with `plan-A-1` representing the first step of plan A, and `plan-1-A` which specifies the `on-abort`-part of the plan activation, which merely calls plan X. The current version does not eliminate such extra plans, but a future version could.

Asbru plans can have arguments and return values. Both are emulated in KIV by global variables stored in the vars aggregation. Figure 30 shows a toy plan which only divides its two arguments and returns both quotient and remainder. For demonstration purposes, it changes the context in the plan-body. Figure 31 and 32 show the translation to KIV.

In the calling plan, the plan activation is implemented as a sequential plan consisting of an assignment subplan for each argument, then the actual subplan invocation, followed by an assignment subplan for each return value. In the called subplan the main plan calls a plan implementing the plan-body as given in Asbru and then an assignment subplan for each return value.

Figure 33 show the plan activation in Asbru. Figures 34 and 35 show the translation to KIV.

Of course arguments and `on-abort` do not exclude each other, but an example showing both is too large to be handy. The translator introduces exactly as many layers of additional subplans as necessary.

## 5.3 Variable Assignment

Variable assignments are implemented in an `asbru-plan` element of type `assignment`. It has two child elements, one for the name of the variable and the other one for the calculation of its new value. The later consist of `mk-value( )` containing an expression based on accesses to patient data which is described in Section 6.1.

In the example in Figures 36 and 37, the current value of parameter P is added to variable A.

## 5.4 Context Change

While Asbru does distinguish between parameters, context, and variables, in the KIV translation context variables are treated like parameters when referred to, but like variables when they are set. Figures 38 shows how the mode of ventilation is changed to CPAP in the third step of plan A. Figure 39 shows the KIV translation.

```

<waitfor>
  <gmt-link link-id="63"/>
  <complex-waitfor type="and">
    <waitfor>
      <complex-waitfor type="and">
        <waitfor>
          <complex-waitfor type="and">
            <waitfor>
              <simple-waitfor>G</simple-waitfor>
            </waitfor>
            <waitfor>
              <simple-waitfor>A</simple-waitfor>
            </waitfor>
          </complex-waitfor>
        </waitfor>
        <waitfor>
          <simple-waitfor>C</simple-waitfor>
        </waitfor>
      </complex-waitfor>
    </waitfor>
    <waitfor>
      <simple-waitfor>F</simple-waitfor>
    </waitfor>
  </complex-waitfor>
</waitfor>

```

Figure 24: Continuation specification *wait for A and C and F and G* in KIV.

```

<wait-for>
  <gmt-link link-id="70" />
  <cardinality>
    <numerical-constant value="2" />
  </cardinality>
</wait-for>

```

Figure 25: Continuation specification *wait for 2* in Asbru.

```

<waitfor>
  <gmt-link link-id="70"/>
  <wait-for-n number="2"/>
</waitfor>

```

Figure 26: Continuation specification *wait for 2* in KIV.

```

<control type="anyorder">
  <subplan>plan-D-1</subplan>
  <subplan>plan-D-2</subplan>
  <subplan>plan-D-3</subplan>
  <subplan>plan-D-4</subplan>
  <waitfor>
  <gmt-link link-id="80" />
  <wait-for-n number="4" />
</waitfor>

```

Figure 27: Continuation specification *wait for all* in KIV. The total count of subplans is given as the number of subplans to wait for.

```

<plan name="A">
  ...
  <plan-body>
    <subplans type="sequentially" label="sub">
      <gmt-link link-id="64" />
      <plan-activation>
        <gmt-link link-id="54" />
        <plan-schema name="F" />
        <on-abort>
          <plan-activation>
            <gmt-link link-id="55" />
            <plan-schema name="X" />
          </plan-activation>
        </on-abort>
      </plan-activation>
    </subplans>
  </plan-body>
  ...

```

Figure 28: Plan activation with on-abort in Asbru.

```

<asbru-plan lemmaname="plan-A" planname="plan-A">
  <gmt-link link-id="64"/>
  <control type="sequential">
    <subplan>plan-A-1</subplan>
    ...
  </control>
</asbru-plan>
<asbru-plan lemmaname="plan-A-1" planname="plan-A-1">
  <gmt-link link-id="54"/>
  <control type="onabort">
    <subplan>plan-F</subplan>
    <subplan>plan-X</subplan>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

```

Figure 29: Plan activation with on-abort in KIV.

```

<plan name="divide">
  <gmt-link link-id="88" />
  <arguments>
    <argument name="first" type="length" />
    <argument name="second" type="length" />
  </arguments>
  <plan-body>
    <gmt-link link-id="85" />
    <set-context value="IPPV" name="mode-of-ventilation">
      <gmt-link link-id="84" />
    </set-context>
  </plan-body>
  <returns>
    <return-value name="quotient">
      <gmt-link link-id="86" />
      <operation operator="divide">
        <argument-ref name="first" />
        <argument-ref name="second" />
      </operation>
    </return-value>
    <return-value name="remainder">
      <gmt-link link-id="87" />
      <operation operator="modulo">
        <argument-ref name="first" />
        <argument-ref name="second" />
      </operation>
    </return-value>
  </returns>
</plan>

```

Figure 30: Plan definition with arguments and return values in Asbru.

```

<asbru-plan lemmaname="plan-divide" planname="plan-divide">
  <gmt-link link-id="88"/>
  <control type="sequential">
    <subplan>
      <gmt-link link-id="85"/>
      plan-divide-B
    </subplan>
    <subplan>
      <gmt-link link-id="86"/>
      plan-divide-R-1
    </subplan>
    <subplan>
      <gmt-link link-id="87"/>
      plan-divide-R-2
    </subplan>
    <waitfor>
      <wait-for-n number="3"/>
    </waitfor>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

<asbru-plan lemmaname="plan-divide-B" planname="plan-divide-B">
  <gmt-link link-id="84"/>
  <control type="assignment">
    <name>'contextname-mode-of-ventilation'</name>
    <value>mk-value (IPPV)</value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

<asbru-plan lemmaname="plan-divide-R-1" planname="plan-divide-R-1">
  <gmt-link link-id="86"/>
  <control type="assignment">
    <name>'returnvalue-divide-quotient'</name>
    <value>
      mk-value( ( vars['argument-divide-first'] .val /
                    vars['argument-divide-second'] .val ) )
    </value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

```

Figure 31: Plan definition with arguments and return values in KIV – part 1.

```

<asbru-plan lemmaname="plan-divide-R-2" planname="plan-divide-R-2">
  <gmt-link link-id="87"/>
  <control type="assignment">
    <name>'returnvalue-divide-remainder'</name>
    <value>
      mk-value(mod ( vars['argument-divide-first'] .val,
                      vars['argument-divide-second'] .val ) )
    </value>
  </control>
</conditions/>
</intentions/>
</asbru-plan>

```

Figure 32: Plan definition with arguments and return values in KIV – part 2.

```

<plan-activation>
  <gmt-link link-id="78" />
  <plan-schema name="divide">
    <gmt-link link-id="77" />
    <argument-value name="first">
      <gmt-link link-id="73" />
      <numerical-constant unit="m" value="1" />
    </argument-value>
    <argument-value name="second">
      <gmt-link link-id="74" />
      <operation operator="add">
        <parameter-ref name="A" />
        <numerical-constant unit="cm" value="1" />
      </operation>
    </argument-value>
    <return-value-assignment variable-name="quotient">
      <gmt-link link-id="75" />
    </return-value-assignment>
    <return-value-assignment variable-name="A"
                             return-value-name="remainder">
      <gmt-link link-id="76" />
    </return-value-assignment>
  </plan-schema>
</plan-activation>

```

Figure 33: Plan activation with arguments and return values in Asbru.

First step of plan D:

```
<subplan>
  <gmt-link link-id="78"/>
  plan-D-1
</subplan>
```

Subplans of D:

```
<asbru-plan lemmaname="plan-D-1" planname="plan-D-1">
  <gmt-link link-id="77"/>
  <control type="sequential">
    <subplan>
      <gmt-link link-id="73"/>
      plan-D-1-A-1
    </subplan>
    <subplan>
      <gmt-link link-id="74"/>
      plan-D-1-A-2
    </subplan>
    <subplan>
      <gmt-link link-id="77"/>
      plan-divide
    </subplan>
    <subplan>
      <gmt-link link-id="75"/>
      plan-D-1-R-1
    </subplan>
    <subplan>
      <gmt-link link-id="76"/>
      plan-D-1-R-2
    </subplan>
    <waitfor>
      <wait-for-n number="5"/>
    </waitfor>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

<asbru-plan lemmaname="plan-D-1-A-1" planname="plan-D-1-A-1">
  <gmt-link link-id="73"/>
  <control type="assignment">
    <name>'argument-divide-first'</name>
    <value>mk-value(1000)</value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>
```

Figure 34: Plan activation with arguments and return values in KIV - part 1.

```

<asbru-plan lemmaname="plan-D-1-A-2" planname="plan-D-1-A-2">
  <gmt-link link-id="74"/>
  <control type="assignment">
    <name>'argument-divide-second'</name>
    <value>mk-value( ( pd[ 'parameter-A' ] .val +
                      10 ) )</value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

<asbru-plan lemmaname="plan-D-1-R-1" planname="plan-D-1-R-1">
  <gmt-link link-id="75"/>
  <control type="assignment">
    <name>'variable-D-quotient'</name>
    <value>mk-value(vars[ 'returnvalue-divide-quotient' ] .val)</value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

<asbru-plan lemmaname="plan-D-1-R-2" planname="plan-D-1-R-2">
  <gmt-link link-id="76"/>
  <control type="assignment">
    <name>'variable-D-A'</name>
    <value>
      mk-value(vars[ 'returnvalue-divide-remainder' ] .val)
    </value>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

```

Figure 35: Plan activation with arguments and return values in KIV - part 2.

```

<variable-assignment variable="A">
  <gmt-link link-id="56" />
  <operation operator="add">
    <parameter-ref name="P" />
    <variable-ref name="A" />
  </operation>
</variable-assignment>

```

Figure 36: Variable assignment in Asbru.

```

<asbru-plan lemmaname="plan-A-2" planname="plan-A-2">
  <gmt-link link-id="56"/>
  <control type="assignment">
    <name>'variable-A'</name>
    <value>
      mk-value( ( pd['parameter-P'] .val +
                  vars['variable-A'] .val) )
    </value>
  </control>
</conditions/>
</intentions/>
</asbru-plan>

```

Figure 37: Variable assignment in KIV.

```

<set-context value="CPAP" name="mode-of-ventilation">
  <gmt-link link-id="57" />
</set-context>

```

Figure 38: Context change in Asbru.

```

<asbru-plan lemmaname="plan-A-3" planname="plan-A-3">
  <gmt-link link-id="57"/>
  <control type="assignment">
    <name>'contextname-mode-of-ventilation'</name>
    <value>mk-value(CPAP)</value>
  </control>

```

Figure 39: Context change in KIV.

```

<ask>
  <gmt-link link-id="52" />
  <parameter-ref name="B" />
  <time-out>
    <numerical-constant unit="min" value="10" />
  </time-out>
</ask>

```

Figure 40: Ask statement in Asbru.

```

<asbru-plan lemmaname="plan-A-4" planname="plan-A-4">
  <gmt-link link-id="52"/>
  <control type="ask">
    <parameter-ref name="parameter-B"/>
    <waitingperiod>minute(10)</waitingperiod>
  </control>
  <conditions/>
  <intentions/>
</asbru-plan>

```

Figure 41: Ask statement in KIV.

## 5.5 Ask Statements

The ask statement means that a new value for a certain parameter is requested from the user. Optionally, the further execution of the guideline is postponed for a certain period of time. There is no guarantee that the user actually supplies a new value for the parameter before or after the waiting period is over.

Figure 40 shows the fourth step of plan A which asks for a new value for parameter B and wait 10 minutes for it. Figure 41 shows the translation to KIV.

## 5.6 if-then-else

The Asbru element `if-then-else` consists of a condition and branch to be performed if the condition evaluates to true and another one for the contrary case. Both branches contain one or more plan steps.

Similar to plan activation, the translation to KIV requires the creation of extra plans for each branch.

In the plan step (the sixth one of plan A) shown in Figure 42 subplan D is called if the current values of parameter B and variable A are equal. Otherwise, three steps are taken: First, subplan C is called. Then the context `mode-of-ventilation` is changed to `IPPV`. Finally, subplan F is called.

Figure 43 shows the translation to KIV. Note how the `then-branch` of Asbru is mapped to a direct call of D in KIV, while for the multiple steps in the `else-branch` a subplan `plan-A-6-N` is created. It sequentially performs the direct calls of `plan-C` and `plan-F`, and in between them the call to another plan `plan-A-6-N-2` to implement the second step of the *no-branch* of the sixth step of plan A.

## 5.7 User-performed Plan

Figure 44 shows an example of a user-performed plan in Asbru. Figure 45 shows how it is mapped to its KIV equivalent.

```

<if-then-else>
  <gmt-link link-id="59" />
  <simple-condition>
    <comparison type="equal">
      <left-hand-side>
        <variable-ref name="A" />
      </left-hand-side>
      <right-hand-side>
        <parameter-ref name="B" />
      </right-hand-side>
    </comparison>
  </simple-condition>
  <then-branch>
    <plan-activation>
      <gmt-link link-id="54" />
      <plan-schema name="D" />
    </plan-activation>
  </then-branch>
  <else-branch>
    <gmt-link link-id="58" />
    <plan-activation>
      <gmt-link link-id="55" />
      <plan-schema name="C" />
    </plan-activation>
    <set-context value="IPPV" name="mode-of-ventilation">
      <gmt-link link-id="56" />
    </set-context>
    <plan-activation>
      <gmt-link link-id="57" />
      <plan-schema name="F" />
    </plan-activation>
  </else-branch>
</if-then-else>

```

Figure 42: if-then-else in Asbru.

```

<asbru-plan lemmaname="plan-A-6" planname="plan-A-6">
  <gmt-link link-id="59"/>
  <control type="ifthenelse">
    <simple-condition>
      lambda pdh, vh, ash, as, ac.
        pdh[ac][ 'variable-A' ] .val
        = pdh[ac][ 'parameter-B' ] .val
    </simple-condition>
    <subplan>
      <gmt-link link-id="54"/>
      plan-D
    </subplan>
    <subplan>
      plan-A-6-N
    </subplan>
  </control>
</conditions/>
</intentions/>
</asbru-plan>
<asbru-plan lemmaname="plan-A-6-N" planname="plan-A-6-N">
  <gmt-link link-id="58"/>
  <control type="sequential">
    <subplan>
      <gmt-link link-id="55"/>
      plan-C
    </subplan>
    <subplan>
      <gmt-link link-id="56"/>
      plan-A-6-N-2
    </subplan>
    <subplan>
      <gmt-link link-id="57"/>
      plan-F
    </subplan>
    <waitfor>
      <wait-for-n number="3"/>
    </waitfor>
  </control>
</conditions/>
</intentions/>
</asbru-plan>
<asbru-plan lemmaname="plan-A-6-N-2" planname="plan-A-6-N-2">
  <gmt-link link-id="56"/>
  <control type="assignment">
    <name> 'contextname-mode-of-ventilation' </name>
    <value>mk-value (IPPV) </value>
  </control>
</conditions/>
</intentions/>
</asbru-plan>

```

Figure 43: if-then-else in KIV.

```

<plan name="plan-C">
  <gmt-link link-id="65"/>
  <explanation text="Administer 5mg of Polymalorum"/>
  <plan-body>
    <user-performed/>
  </plan-body>
</plan>

```

Figure 44: User-performed Asbru plan.

```

<asbru-plan lemmaname="plan-C" planname="plan-C">
  <gmt-link link-id="65"/>
  <control type="user"/>
  <conditions/>
  <intentions/>
</asbru-plan>

```

Figure 45: The user-performed plan shown in Figure 44 in its KIV representation.

## 6 Translating Basic Elements

### 6.1 Expressions

Expressions in Asbru Light are an combination of

- numerical constants
- qualitative constants
- constants defined with constant-def
- variable references
- parameter references
- the time point now (the time of evaluation of the expression)
- plan state changes

#### 6.1.1 Numerical constants

In Asbru, each numerical constant has a value and a unit. The type of the constant is inferred from the context, e.g., by the type of the variable to which this constant is assigned to. This type must match the unit. The translator converts all constants to integers of the smallest unit for this type. This is necessary to facilitate the verification (real numbers would be too complex to verify).

Constants of type date and time are treated differently. They are translated to predicates returning a value of KIV type `asbru-clock`.

#### 6.1.2 Qualitative constants

The value of qualitative constants is directly used in KIV. They are not prefixed since qualitative values of different scales cannot be combined due to the different type on the Asbru level and KIV does not prohibit the use of a single name such as "normal" for different scales (i.e., qualitative data types).

#### 6.1.3 Constants

The name of constants is prefixed by constantname when used in KIV to distinguish it from variables etc. with the same name.

### 6.1.4 Variable references

The KIV syntax for accessing variable A is `vars[ 'variable-A' ] .val`. This represents the access to the patient data, which contains the current value of A which is selected by the selector `.val`.

### 6.1.5 Parameter references

The current value of a parameter is accessed in KIV as `pd[ 'parameter-A' ] .val`.

### 6.1.6 Now

The current time point is referenced as `now` in Asbru. In expression it is accessed as `ac`. As a reference point in time annotations (see below) it is accessed as `*now*`.

### 6.1.7 Plan state changes

In the current implementation, KIV does not allow the reference to plan state changes in expressions. As a reference point, plan state changes are referred to as `enter (plan name, plan state)`.

## 6.2 Conditions

Figure 46 shows a condition combining most of the features of a condition in Asbru. The plan containing this condition is suspended if variable A is less than the current value of parameter B and parameter B is equal to 1 m while mode-of-ventilation is either IPPV or CPAP and there is no life-threatening situation for an interval starting no earlier than 2 hours after plan C has been suspended. The duration of the interval must be 1 to 2 hours.

Asbru distinguishes between parameter-proposition and simple-condition. The first has a temporal dimension, the second has not. In KIV, the parameter proposition contains a constraint (combination) which is identical to a translation of a simple-condition. The only difference is the presence of a time annotation.

### 6.2.1 Parameter propositions

Asbru parameter propositions consist of a parameter name, a value description, a context and a time annotation. The parameter proposition is fulfilled, if the parameter has the described value and the context is given for an interval matching the time annotation.

Conditions (also those used in parameter propositions), consist of an expression preceded by the lambda function `lambda pdh, vh, ash, as, ac. pdh[ac]`. This expression combines the Asbru value-description and the context.

### 6.2.2 Time annotations

Although syntactically different in structure, the time annotations of Asbru and KIV parallel each other strongly. In the current implementation of Asbru in KIV, only constants are allowed in the shifts of the time annotation while in Asbru arbitrary expression (referring to variables) are admitted.

The reference point of a time annotation is a plan-state change or the current time `*now*`. In the first case, the last entering (`enter`) or leaving (`leave`) of a plan state for a certain plan is specified.

The Asbru reference point `self` refers to the time of activation of the plan containing it. This is mapped to `leave (planname, possible)` where `planname` is the name of the plan which contains the time annotation since this is safer than the more intuitive `enter (planname, activated)` since this would match the most recent time the plan was resumed, too. I.e., it would only work as expected if the plan is never suspended.

```

<suspend-condition confirmation-required="no" overridable="no">
  <constraint-combination type="and">
    <simple-condition>
      <comparison type="less-than">
        <left-hand-side>
          <variable-ref name="A" />
        </left-hand-side>
        <right-hand-side>
          <parameter-ref name="B" />
        </right-hand-side>
      </comparison>
    </simple-condition>
    <parameter-proposition parameter-name="B">
      <gmt-link link-id="50" />
      <value-description type="equal">
        <numerical-constant unit="m" value="1" />
      </value-description>
      <context>
        <context-combination operator="and">
          <one-of name="mode-of-ventilation">
            <value-ref name="IPPV" />
            <value-ref name="CPAP" />
          </one-of>
          <context-not>
            <context-ref name="life-threatening-situation" />
          </context-not>
        </context-combination>
      </context>
      <time-annotation>
        <time-range>
          <starting-shift>
            <earliest>
              <numerical-constant unit="h" value="2" />
            </earliest>
          </starting-shift>
          <duration>
            <minimum><numerical-constant unit="h" value="1" />
            </minimum>
            <maximum><numerical-constant unit="h" value="2" />
            </maximum>
          </duration>
        </time-range>
        <plan-state-transition direction="enter"
          instance-type="last" state="suspended">
          <plan-pointer>
            <static-plan-pointer plan-name="C" />
          </plan-pointer>
        </plan-state-transition>
      </time-annotation>
    </parameter-proposition>
  </constraint-combination>
</suspend-condition>

```

Figure 46: Sample condition in Asbru.

```

<suspend-condition overridable="false" confirmation_required="false">
  <constraint-combination type="and">
    <simple-constraint>
      lambda pdh, vh, ash, as, ac. pdh[ac]['variable-A'] .val
                                     < pdh[ac]['parameter-B'] .val
    </simple-constraint>
    <parameter-proposition>
      <gmt-link link-id="50" />
      <time-annotation>
        <ess>hour(2)</ess>
        <minDuration>hour(1)</minDuration>
        <maxDuration>hour(2)</maxDuration>
        <referencePoint>leave(plan-C, suspended)</referencePoint>
      </time-annotation>
      <simple-constraint>
        lambda pdh, vh, ash, as, ac. pdh[ac]['parameter-B']
                                       .val = 1000
          and ( ((pdh[ac]['contextname-mode-of-ventilation']
                  .val = IPPV
                  or pdh[ac]['contextname-mode-of-ventilation']
                  .val = CPAP)
                and not (pdh[ac]['contextname-life-threatening-situation']
                          .val)) )
      </simple-constraint>
    </parameter-proposition>
  </constraint-combination>
</suspend-condition>

```

Figure 47: Sample condition in KIV.

## 7 Change History and Future Development

### 7.1 Improvements in Version 2.0

Besides several bug fixes, the following improvements were made.

**Logical combination in data abstraction.** Asbru elements `logical-combination-def` and `boolean-def` are implemented with the limitation that the parameter proposition may only contain context *any* and time annotation *now*.

**Cyclical plan.** To the extend implemented in KIV, Asbru's cyclical-plan is now translated.

**Plan state constraint.** Asbru element `plan-state-constraint` is translated to KIV.

**Plan grouping.** Plans are grouped now using an ad-hoc heuristic: First all plans implementing the same Asbru plan are grouped together. Then these grouped are merged to arrive at groups of 10 to 20 plans each.

**Special complete condition for user-performed plans.** User-performed plans now have a special `usercomplete-condition` if they do not have a normal one.

**Special time-annotation.** The time-annotation `now` translates to the symbolic time annotation `default-ta` now instead of its standard equivalent.

### 7.2 Potential for Further Improvements

In ranking the priorities, we were guided by the need to implement the Dutch Breast Cancer guideline which is focus of this project. Therefore, several issues are left open for future version of the translator and for extensions of the implementation of Asbru in KIV.

#### 7.2.1 Extension of functionality

The following issues would extend the functionality of Asbru Light without requiring changes to the KIV implementation of Asbru.

**String constants** could be translated to KIV symbols if they were used in the guideline.

**Time-annotation in plan-activation.** Each Asbru plan-activation can contain a time-annotation limiting the execution time of the invoked plan. This is only partially supported by KIV, but the translator could create additions to the `complete-` and `abort-conditions` to implement this feature to a larger.

**Boolean-valued functions.** KIV requires that abstraction functions returning Booleans are translated to `prddef` instead of `fctdef`. The current implementation of the translator does not consider this exception.

#### 7.2.2 Improving output quality

The following changes of the translator would produce better KIV code with equal functionality.

**Substitute arguments by their value.** Mapping functions would be further improved by replacing "`a < 20` and `c = A  $\mapsto$  map(a, c) = X`" by `a < 20  $\mapsto$  map(a, A) = X`. I.e. the argument `c` is replaced by its value in the argument list and it is dropped from the conjunction.

**Split lemmas on disjunction.** Currently, an `or` is translated like an `and` in the mapping functions which perform parameter abstraction. Provided that the arguments are substituted by their values, the verification process would benefit, if "`c = A or c = B  $\mapsto$  map(c) = X`" would be replaced by two lemmas "`c = A  $\mapsto$  map(c) = X`" and "`c = B  $\mapsto$  map(c) = X`". This complicates the translation, because several disjunctions can occur in one mapping function and all permutations must be produced.

**Asbru comments, explanation, and library-info.** KIV provides an element for comments. The mentioned pieces of information could be stored there. Currently they are lost in the translation.

**Redundant Boolean comparison.** The current implementation does not treat Booleans special in comparisons. Therefore, Booleans may be compared to true or false which could be optimized.

**Nice indentation in nested structures.** The current implementation only produces an approximation of what is considered a nicely formatted XML file. However, this is ignored by the tools used for further processing.

### 7.2.3 KIV extensions

The following issues appear to be desirable in order to extend the scope of Asbru Light. However, in the model of the guideline created in this project they are not used.

**Variables in time annotations.** The restriction to constants in the shifts of the time annotation appears to be severe.

**Parameter change as reference points in time-annotation.** Since the parameter history is implemented similar to the plan state history in KIV, the reference to changes can be referred to. However, verification is complicated by such references.

**Trust period of parameters.** Each measurement of a parameter expires after a certain time – the trust period. This is given by the parameter definition in Asbru. The implementation in KIV is under construction.

## References

- [1] A. Seyfang, R. Kosara, and S. Miksch. Asbru 7.3 reference manual. Technical report, Vienna University of Technology, 2002. available at [www.asgaard.tuwien.ac.at](http://www.asgaard.tuwien.ac.at).