Использование условных конструкций

Введение

При описании утилит бывают ситуации, когда командная строка утилиты или часть командной строки, число и вид параметров, вид диалога настройки динамически зависят от некоторых внешних условий или от текущих значений параметров этой утилиты.

Например, параметр A некоторой утилиты может иметь смысл, только если значение второго параметра B совпадает со значением другого параметра C. В этом случае от параметра B зависит общий вид командной строки, а также диалог настройки утилиты – бессмысленно предлагать вводить значение бесполезного параметра.

Другой пример. Многие компиляторы, работающие с программами на нескольких языках, например, C/C++, содержат языково-зависимые опции. То есть, для такой утилиты-компилятора при подаче на вход программы на языке C большая группа входных параметров, относящихся только к компилированию программ на C++, просто не имеет смысла.

Для поддержки таких зависимых свойств утилит в язык TSL введены специальные синтаксические конструкции. Эти синтаксические конструкции разбиваются на два класса: строковые и структурные условия.

Структурные условные конструкции

Конструкции этого типа служат для условной активации и деактивации целых групп параметров и командных строк.

В XML-файле структурные условные конструкции задаются специальными XML-тэгами. Общий их синтаксис таков:

```
<COND param1="value1"
      param2="value2"
      ...
      paramN="valueN">
   ...
</COND>
```

Здесь N больше или равно 1. *COND* – одно из слов `"if"`, `"if-not"`, `"if-and"`.

Семантика конструкции `if` такова: элемент (содержимое блока, ограниченного открывающим и закрывающим тэгом данного типа) имеет смысл если и только если текущее значение хотя бы одного параметра с идентификатором `param`*i* равно соответствующему `value`*i* (*i*=1..N).

Семантика конструкции `if-not` противоположна семантике `if`: элемент имеет смысл если и только если текущие значения ни одного из параметров с идентификаторами `param`*`i`* не равно соответствующему `value`*`i`*.

Семантика конструкции `if-and`: элемент имеет смысл если и только если текущие значения всех параметров с идентификаторами `param`*`i`* равны соответствующим `value`*`i`*.

В дальнейшем будем писать `param="value"`, понимая под этим «текущее значение параметра `param` совпадает со значением `value`», и `param≠"value"`, понимая под этим «текущее значение параметра `param` не совпадает со значением `value`». Аналогично, будем писать `param1=param2`, понимая под этим «текущее значение параметра `param1` совпадает с текущим значением параметра `param2`», и `param1≠param2`, понимая под этим «текущее значение параметра `param1` не совпадает с текущим значением параметра `param2`». (Заметим, однако, что последнюю пару выражений нельзя записать в структурной условной конструкции).

Условные конструкции могут быть вложенными. Например,

```
<if param1="value1">
  <if-and param2="value2"
          param3="value3">
    <if-not param4="value4">

      <parameter id="ParamA" .../>

    </if-not>
  </if-and>
</if>
```

В этом случае объявление параметра `ParamA` произойдет только если одновременно `param1="value1"`, `param2="value2"`, `param3="value3"` и `param4≠"value4"`.

Структурные условные конструкции могут применяться в следующих разделах описания контекста:

- в секции параметров;
- в секции `input` (описания диалога ввода);
- в секции `output` (описания командных строк и командных файлов).

Например, при описании утилиты-компилятора C/C++ зададим в соответствующем проекте параметр `UsedLanguage` типа `Enum`, принимающий значения `"C"` и `"C++"`. Тогда языково-зависимые параметры компилятора можно описать так:

```
<if UsedLanguage="C">
  <parameter id="ANSI_C_Compliant" .../>
  <parameter id="Allow_CPP_Comments" .../>
```

```
  ...
</if>
<if UsedLanguage="C++">
  <parameter id="ANSI_CPP_Compliant" .../>
  <parameter id="Enable_RTTI" .../>
  ...
</if>
<!-- Общие параметры -->
...
```

Ясно, что заданные параметры необходимо включать или исключать из диалога настройки данной утилиты/контекста. Для этого необходимо применить условные конструкции в секции `input`, например, так:

```
<input>
  ...
  <group ...>
    ...
    <if UsedLanguage="C">
      "ANSI_C_Compliant"
      "Allow_CPP_Comments"
      ...
    </if>
    <if UsedLanguage="C++">
      "ANSI_CPP_Compliant"
      "Enable_RTTI"
    </if>
    ...
  </group>
  ...
</input>
```

или включать/выключать целиком группы ввода параметров:

```
<input>
  ...
  <if UsedLanguage="C">
    <group name="CParams" label="C specific options">
      "ANSI_C_Compliant"
      "Allow_CPP_Comments"
      ...
    </group>
  </if>
  <if UsedLanguage="C++">
    <group name="CPPParams" label="C++ specific options">
      "ANSI_CPP_Compliant"
      "Enable_RTTI"
    ...
    </group>
  </if>
  ...
</input>
```

Аналогично задаются условные параметры в командной строке компилятора, например:

```
<output>
  <line name = "CommandLine">
    <if UsedLanguage="C">
      "%ANSI_C_Compliant"
      "%Allow_CPP_Comments"
      ...
    </if>
    <if UsedLanguage="C++">
      "%ANSI_CPP_Compliant"
      "%Enable_RTTI"
      ...
    </if>
    ...
  </line>
  ...
<output>
```

В случае необходимости можно поставить под условную конструкцию всю секцию `<line>`.

Строковые условные выражения

Во многих случаях требуется ввести условный параметр, одно или несколько свойств которого зависят от других параметров. В такой ситуации введение структурных условий бывает нецелесообразно: они громоздки, их применение может вести к значительному дублированию описания. Наконец, структурные условия в силу своей упрощенности (не всякое сложное условие можно выразить их средствами) могут не обеспечивать требуемой гибкости.

Строковые условные выражения избавлены от недостатков структурных и предназначены для задания условных атрибутов параметров, таких как значение по умолчанию.

Строковое условное выражение задается строкой одного из двух видов:

```
1. "?strS: str1=res1, … , strN=resN[, resDef]"
2. "?condition: resT, resF"
```

Все поля вида `str` и `res` – произвольные строки, записываемые без кавычек. Границы этих строк определяются предшествующим и последующим символами-разделителями в условном выражении; по этой причине не следует использовать финальный разделитель внутри строки. Кроме того, отсекаются начальные и финальные пробелы (но внутренние пробелы допустимы и значимы при сравнениях). Незначащие пробелы, таким образом, можно вставлять для красоты

только до и после разделителей. Знак `"?"`, по которому строка распознается как условное выражение, должен быть первым символом строки.

Условие `condition` записывается как традиционное логическое выражение:

- элементарные сравнения – пары вида `strA = strB` или `strA # strB`;
- условие строится из элементарных сравнений и логических связок `"|"` (ИЛИ) и `"^"` (И) с использованием скобок для изменения приоритетов связок.

В строковых полях вида `str` и `res` можно использовать генераторы вида `%param` для подстановки текущих значений параметров. Именно их использование придает нетривиальный смысл условному выражению.

Интерпретация условного выражения заключается в подстановке значений всех параметров-генераторов в строки, выполнении серии сравнений полученных строк и выдаче некоторой строки в качестве *результата* данного условного выражения.

Условное выражение вида 1 интерпретируется как традиционный оператор выбора по параметру. После подстановки значений всех генераторов строка `strS` поочередно сравнивается в заданном порядке со строками-*селекторами* `strk`. Если сравнение с *i*-м селектором *успешно* (различий не обнаружено), значением условного выражения становится соответствующая строка-результат `resi`. Если ни одно из сравнений не было успешным и в выражении указана строка `resDef`, результатом становится `resDef`. Если `resDef` не задана, результатом будет пустая строка.

Условное выражение вида 2 может применяться в случаях, когда строка-результат сложным образом зависит от нескольких параметров. Интерпретируется оно так: вычисляется значение условия `condition`, если оно истинно, результатом стновится строка `resT`, в противном случае – `resF`.

Несколько примеров условных выражений.

1) Выражение:

```
?%MyParam: %AnotherParam=Fred, MyValue=Wilma, Barney
```

имеет результатом `"Fred"`, если `MyParam=AnotherParam`. В противном случае, если `MyParam="MyValue"`, результатом будет `"Wilma"`, а иначе `"Barney"`.

2) Выражение:

```
?(%par1 = %par2 | par3 = foo) ^ %par4 # bar: Fred, Wilma
```

будет иметь результатом `"Fred"`, если либо `par1=par2` и `par4≠"bar"`, либо `par3="foo"`, и `par4≠"bar"`. В противном случае результатом будет строка `"Wilma"`.

Строковые условные выражения допускаются в следующих атрибутах параметров: `default`, `omit`, `visible` и `readonly`.

Например, в описании C/C++ компилятора может потребоваться параметр, задающий расширение для входных файлов. Его можно описать так:

```
<parameter id="SourceExtension"
           default="%?UsedLanguage=C: c, cpp"
           .../>
```

или так:

```
<parameter id="SourceExtension"
           default="%?UsedLanguage: C=c, C++=cpp"
           .../>
```

Этот же параметр можно было бы описать с помощью структурного условия, но длиннее:

```
<if UsedLanguage="C">
  <parameter id="SourceExtension"
             default="c"
             .../>
</if>
<if UsedLanguage="C++">
  <parameter id="SourceExtension"
             default="cpp"
             .../>
</if>
```

и при изменении любого другого атрибута в описании параметра пришлось бы следить за синхронностью изменений в обоих вариантах.