Algorithms of "DeriveNameLists" Member Function

Syntax of NameLists

NameList parameters have limited syntax.[1] Only the following characters are permitted :

- Names. To be valid, the only permissible characters in a name are :
  - Letters of the English alphabet, upper & lower case.
  - The characters !, ?, &, and _.[2]
  - The numbers 0-9, except in the initial position.

  Other characters are avoided, either because of their special meaning, or to avoid confusion.
- The assignment **<--**. Note that this assignment associates what is on its right-hand side with the name on its left-hand side. It therefore corresponds to **<--** in expressions, but has a different meaning. It is chosen to provide a consistent syntax throughout the RAQUEL notation. (There is no counterpart to the **<==** or **==** assignments).
- Tilde, '~', to represent negation or 'opposite of'.
- The delimiters ',' and ';'.

Overview

In principle, the algorithms could be relatively simple. In practice, it is important to include checking for possible errors in the parameter text, and the checking makes the algorithms relatively complex.

The following algorithms all assume that the parameter's surrounding square brackets have been stripped off, leaving just the enclosed text to be processed.

Parameter text is treated as being held in a character string variable called *Param*. As *Param* is inspected, successive valid characters forming a name are appended to the character string variable called *Name*. A completed name in *Name* is appended to the relevant list. *Name* must be set to zero length before accumulating the characters of a name. Whenever an error is found, an error code designating that kind of error is appended to the vector variable *ErrorList*, which is initialised to a zero-length.

The following additional general-purpose variables are used by multiple algorithms :

- **Boolean** *endItem* ; initialised to ***true***. (Must be true before a name can start).
- **Vector** *List* ; initialised to a ***zero-length*** value. (Accumulates names).

All result lists are assigned directly to the relevant name list data members of a Raquel Token. They should all be set to be empty if one or more errors are found. *ErrorList* is always the explicit output result of the 'Derive' function.

The following sections give the algorithms for the different kinds of parameter name list.

Multiple-Name List

If '~' appears, it must come before any name. Names must be separated from each other by commas. Whitespace may be inserted where required in order to improve readability.

---

[1] Except the **Pattern List**, which is used for pattern matching.

[2] Currency symbols have been dropped because not all are in the basic, standard ASCII character set.

The following additional variables are used :

- **Boolean** *comma* ; initialised to ***true***. (Must be true before a name can start).
- **Boolean** *firstTilde* [3]; initialised to ***false***. (For checking against multiple '~' symbols).

The algorithm is of the state-transition form. As it is straightforward, state is determined by the values of flags, rather than explicit states *per se*. The algorithm is described in the following table :-

Event	Action
whitespace	**IF** *Name*.length > 0 **THEN** { Append *Name* to *List* ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** }
character	**IF** invalid character **THEN** Append ***Invalid_Character*** to *ErrorList* **ELSE IF** *Name*.length = 0 **THEN** { **IF** (*comma* = ***true*** **AND** *endItem* = ***true***) **THEN** { Append character to *Name* ; *comma* **<--** ***false*** ; *endItem* **<--** ***false*** } **ELSE IF** (*comma* = ***false*** **AND** *endItem* = ***true***) **THEN** { Append ***Comma_Omitted*** to *ErrorList* ; *endItem* = ***false*** } **ELSE** Append character to *Name* **ELSE IF** (*comma* = ***false*** **AND** *endItem* = ***false***) **THEN** Append character to *Name.*
~	**IF** (*Name*.length > 0 **OR** *List*.size > 0) **THEN** Append ***~_NotFirst*** to *ErrorList* **ELSE** { **IF** *firstTilde* = ***false*** **THEN** { Append '~' to ***SpecialList*** ; *firstTilde* = ***true*** } **ELSE** Append ***Repeated_~*** to *ErrorList*. }
,	**IF** comma = ***true*** **THEN** Append ***ItemOmitted_Before_Comma*** to *ErrorList* **ELSE** *comma* **<--** ***true*** ; **IF** *Name*.length > 0 **THEN** { Append *Name* to *List* ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** }
No more input	**IF** *Name*.length > 0 **THEN** Append *Name* to *List* **IF** (*List*.size > 0 **AND** *comma* = ***true***) **THEN** Append ***ItemOmitted_After_Comma*** to *ErrorList*

[3] The '~'symbol is not allowed in C++ variable names.

After completion :

IF *ErrorList*.size > 0
THEN IF ***SpecialList***.size > 0 **THEN** Make ***SpecialList*** zero size.
ELSE { Assign value of *List* to the first element of ***NameLists*** ;
IF ***SpecialList***.size = 0 **THEN** Append ' ' to it }

Single-Assignment List

Such a list consists of a sequence of statements, separated by semi-colons. Every statement consists of a 3-part sequence : a left-hand name, a '**<--**', and a right-hand name. ('~' may not appear in a statement).

Therefore it is convenient to consider the algorithm as a state-transition type, with 3 states, one for each of the 3 parts of a statement, called **Left**, **Assign**, and **Right**. The algorithm starts in the **Left** state.

The following additional variables are used :

- **Character** *state* ; initialised to 'Left'. (Used to indicate the Left, Assign, and Right states).
- **Boolean** *assignQ* : initialised to ***false***. (Used to record an assignment has been made).
- **Integer** *assign* ; initialised to ***zero***. (Used to count the no. of chars. in the '**<--**' symbol).
- **Integer** *Lcount* : initialised to ***zero***. (Used to count Left-hand names).
- **Integer** *Rcount* : initialised to ***zero***. (Used to count Right-hand names).
-

The algorithm is described in the following tables :-

State : **Left**

Event	Action	State Change
whitespace	**IF** *Name*.length > 0 **THEN** { Append *Name* to ***LeftList*** ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** ; *Lcoun*t <-- *Lcount* + 1 }	-
character	**IF** invalid character **THEN** Append ***Invalid_Character*** to *ErrorList* **ELSE IF** *Name*.length = 0 **THEN** { Append character to *Name* ; *endItem* **<--** ***false*** }. **ELSE IF** *endItem* = ***false*** **THEN** Append character to *Name*	-

<	*assign* **<--** 1 ; *assignQ* <-- ***true*** **IF** *Name*.length > 0 **THEN** { Append *Name* to ***LeftList*** ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** ; *Lcoun*t <-- *Lcount* + 1 } *state* **<--** ‘A’ **IF** *Lcount* = 0 **THEN** Append ***No_LeftHand_Value*** to *ErrorList* **IF** *Lcount* > 1 **THEN** Append ***TooMany_LeftHand_Values*** to *ErrorList* *Lcount* **<--** 0	→ **Assign**
No more input	**IF** *ch* = ‘;’ **THEN** Append ***Assignment_Omitted*** to *ErrorList* **ELSE** Append ***No_Assignment_Arrow*** to *ErrorList*.	

State : **Assign**

Event	Action	State Change
-	*assign* **<--** *assign* + 1.	**-**
Any other input	**IF** *assign* = 3 **THEN** *state* **<--** ‘R’ **ELSE** Append ***Invalid_Assignment_Arrow*** to *ErrorList* **IF** whitespace **THEN** *state* **<--** ‘R’ **ELSE IF** invalid character **THEN** Append ***Invalid_Character*** to *ErrorList* **ELSE IF** *Name*.length = 0 **THEN** { Append character to *Name* ; *endItem* **<--** ***false*** ; *state* **<--** ‘R’ }.	→ **Right**
No more input	**IF** *assign* ≠ 3 **THEN** Append ***Invalid_Assignment_Arrow*** to *ErrorList*	

The **Assign** state deals solely with the ‘**<--**‘ arrow.

State : **Right**

Event	Action	State Change
;	*state* **<--** ‘L’ **IF** *Name*.length > 0 **THEN** { Append *Name* to *List* ; Append *List* to ***NameLists*** ; Reset *Name* & *List* to zero-lengths ;; *endItem* **<--** ***true*** ; *Rcount* **<--** *Rcount* + 1 } **IF** *Rcount* = 0 **THEN** Append ***No_RightHand_Value*** to *ErrorList* **IF** *Rcount* > 1 **THEN** Append ***TooMany_RightHand_Values*** to *ErrorList* *Rcount* **<--** 0	→ **Left**
whitespace	**IF** *Name*.length > 0 **THEN** { Create list of one item, and assign *Name* to it ; Append that list to ***NameLists*** ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** }	**-**
character	**IF** invalid character **THEN** Append ***Invalid_Character*** to *ErrorList* **ELSE IF** *Name*.length = 0 **THEN** { Append character to *Name* ; *endItem* **<--** ***false*** }. **ELSE IF** *endItem* = ***false*** **THEN** Append character to *Name* **ELSE** Append ***Syntax*** to *ErrorList.*	**-**
No more input	**IF** *Name*.length > 0 **THEN** { Append *Name* to *List* ; Append *List* to ***NameLists*** ; *Rcount* **<--** *Rcount* + 1 } **IF** *Rcount* = 0 **THEN** Append ***No_RightHand_Value*** to *ErrorList* **IF** *Rcount* > 1 **THEN** Append ***TooMany_RightHand_Values*** to *ErrorList*	

After completion :

IF *ErrorList*.size > 0
THEN Make ***NameLists*** zero size ; Make ***LeftList*** zero size.

Sort-Name List

Names, which may or may not have ‘~’ appearing before them, must be separated by commas.

Pattern List

The list consists of one ‘name’ that consists of a sequence of characters. No check is made of the characters, except to ensure they are all contiguous. Zero or more whitespace characters can appear before and after the ‘name’.

Event	Action
whitespace	**IF** *Name*.length > 0 **THEN** { Append *Name* to ***SpecialList*** ; Reset *Name* to zero-length ; *endItem* **<--** ***true*** }
Any character	**IF** *Name*.length = 0 **THEN** { Append character to *Name* ; *endItem* **<--** ***false*** }. **ELSE IF** *endItem* = ***false*** **THEN** Append character to *Name* **ELSE** { Append ***Multiple_Patterns*** to *ErrorList* ; → Exit. }
No more input	**IF** *Name*.length > 0 **THEN** Append *Name* to ***SpecialList*** **ELSE** Append ***No_Pattern_Given*** to *ErrorList* .