

## UTS #18 Editorial additions

*M. Davis & M. Scherer, 2021-01-XX*

I propose that the following changes be made to <https://unicode.org/reports/tr18/>. They are primarily editorial issues, but include more substantial changes due to problems in the EBNF.

### Annex D: Resolving Character Classes with Strings

[https://unicode.org/reports/tr18/#Resolving Character Ranges with Strings](https://unicode.org/reports/tr18/#Resolving_Character_Ranges_with_Strings)

*Problem: When implementing complements of sets of strings, it is sometimes tricky to wrap your mind around how the operations work. So it is worth some additional explanation and examples. The proposed additional or changed text is in yellow.*

...

When incrementally parsing and building a resolved boolean expression, the process can be analyzed in terms of a series of core operations. In parsing Character Classes, the intermediate objects are logically *enhanced sets* of strings, such as A and B. The enhancement is the addition of a **boolean** flag to indicate whether the **internal** set is *complemented* or not. The symbol **+** stands for the flag value **= normal**. The symbol **-** stands for the flag value **= complemented** (aka negated).

Thus:

- |          |   |
|----------|---|
| <b>+</b> | means that the internal set is treated normally; the enhanced set is the same as the internal set   |
| <b>-</b> | means that the internal set is complemented; the logical contents of the enhanced set are every possible string <i>except those in the internal set</i> . Where $\mathbb{S}$ stands for the set of all strings, and $\{\alpha \ \beta\}$ is the internal set, then the semantics is: $(\mathbb{S} \setminus \{\alpha \ \beta\})$ , that is, the set of all strings <i>except for</i> $\{\alpha \ \beta\}$ . |

When the flag is complemented, adding or removing from the enhanced set has the reverse effect on the internal set.

- *adding*  $\beta$  to  $(\mathbb{S} \setminus \{\alpha \ \beta\})$  is the same as *removing* from the internal set:  $\Rightarrow (\mathbb{S} \setminus \{\alpha\})$
- *removing*  $\gamma$  from  $(\mathbb{S} \setminus \{\alpha \ \beta\})$  is the same as *adding* to the internal set:  $\Rightarrow (\mathbb{S} \setminus \{\alpha \ \beta \ \gamma\})$

For brevity in the table below,  $\neg\{\alpha \ \beta\}$  is used to express  $(\mathbb{S} \setminus \{\alpha \ \beta\})$ .

While logically the enhanced set can contain an infinite set of strings, internally there is only ever a finite set.

### Creation and Unary Operations

- [expression] and \p{expression} (without negation) create enhanced sets with the **internal** sets corresponding to the expression, and the flags set to **+**.
- [^expression] and \P{expression} (with negation) create enhanced sets with the **internal** sets corresponding to the expression, and the flags set to **-**.
- [^A] where A is an enhanced set with (set, flag) results in the flag being **toggled**: **+**  $\Leftrightarrow$  **-**

## Binary Operations

The table shows **how to process** binary operations **on enhanced sets**, with **each** result being the **internal** set plus flag.

Examples are provided with two overlapping sets:  $A = \{\alpha \ \beta\}$  and  $B = \{\beta \ \gamma\}$ .

Syntax	Flag of A	Flag of B	Result Set	Flag of Result	Example Input	Example Result
A    B  (union)	+	+	$\text{setA} \cup \text{setB}$	+	$\{\alpha \ \beta\} \cup \{\beta \ \gamma\}$	$\{\alpha \ \beta \ \gamma\}$
	+	-	$\text{setB} \setminus \text{setA}$	-	$\{\alpha \ \beta\} \cup \neg\{\beta \ \gamma\}$	$\neg\{\gamma\}$
	-	+	$\text{setA} \setminus \text{setB}$	-	$\neg\{\alpha \ \beta\} \cup \{\beta \ \gamma\}$	$\neg\{\alpha\}$
	-	-	$\text{setA} \cap \text{setB}$	-	$\neg\{\alpha \ \beta\} \cup \neg\{\beta \ \gamma\}$	$\neg\{\beta\}$
A && B  (intersection)	+	+	$\text{setA} \cap \text{setB}$	+	$\{\alpha \ \beta\} \cap \{\beta \ \gamma\}$	$\{\beta\}$
	+	-	$\text{setA} \setminus \text{setB}$	+	$\{\alpha \ \beta\} \cap \neg\{\beta \ \gamma\}$	$\{\alpha\}$
	-	+	$\text{setB} \setminus \text{setA}$	+	$\neg\{\alpha \ \beta\} \cap \{\beta \ \gamma\}$	$\{\gamma\}$
	-	-	$\text{setA} \cup \text{setB}$	-	$\neg\{\alpha \ \beta\} \cap \neg\{\beta \ \gamma\}$	$\{\alpha \ \beta \ \gamma\}$
A -- B  (set difference)	+	+	$\text{setA} \setminus \text{setB}$	+	$\{\alpha \ \beta\} \setminus \{\beta \ \gamma\}$	$\{\alpha\}$
	+	-	$\text{setA} \cap \text{setB}$	+	$\{\alpha \ \beta\} \setminus \neg\{\beta \ \gamma\}$	$\{\beta\}$
	-	+	$\text{setA} \cup \text{setB}$	-	$\neg\{\alpha \ \beta\} \setminus \{\beta \ \gamma\}$	$\neg\{\alpha \ \beta \ \gamma\}$

	—	—	$\text{setB} \setminus \text{setA}$	+	$\neg\{\alpha \beta\} \setminus \neg\{\beta \gamma\}$	$\{\gamma\}$
$A \sim B$	+	+	$\text{setA} \setminus \text{setB}$	+	$\{\alpha \beta\} \ominus \{\beta \gamma\}$	$\{\alpha \gamma\}$
(symmetric difference)	+	—	$\text{setA} \cap \text{setB}$	+	$\{\alpha \beta\} \ominus \neg\{\beta \gamma\}$	$\neg\{\alpha \gamma\}$
	—	+	$\text{setA} \cup \text{setB}$	—	$\neg\{\alpha \beta\} \ominus \{\beta \gamma\}$	$\neg\{\alpha \gamma\}$
	—	—	$\text{setB} \setminus \text{setA}$	+	$\neg\{\alpha \beta\} \ominus \neg\{\beta \gamma\}$	$\{\alpha \gamma\}$

The normal set equivalences hold, eg  $\neg(A \cup B) = \neg A \cap \neg B$

**Note: the operation  $A \sim B$  is handled in terms of other operations according to its definition:  $(A \cup B) \setminus (A \cap B)$**

## Annex F. Parsing Character Classes

*Problem: People may get the impression that the parsing of a UTS #18 Character Class expression is tricky, when it involves a fairly small amount of code. We could add a new annex with guidance, as follows. We would also add some references in the body text to this new annex.*

It is reasonably straightforward to build a parser for Character Classes. While there are many ways to do this, the following provides one example of how to do it. This describes a logical process; implementations can use optimized code; using a DFA ([Deterministic Finite Automaton](#)) for processing, for example.

### Storage

At the core is a class (here called `CharacterClass`) that stores the information that is being built, typically a set of strings optimized for compact storage of ranges of characters (such as ICU's [UnicodeSet](#) (C++)).

The methods needed are the following:

```
void addAll(CharacterClass other);           // A = A ∪ other
void retainAll(CharacterClass other);       // A = A ∩ other
void removeAll(CharacterClass other);       // A = A \ other
void complementAll(CharacterClass other);   // A = A ⊖ other

void add(int cp);                          // A = A ∪ {cp}
```

```

void addRange(int cpStart, int cpEnd);           // A = A ∪
{cpStart..cpEnd}

void addString(String stringToAdd);             // A = A ∪ {stringToAdd}

void complement();                             // A = ¬A = S \ A

void setToProperty(String propertyString);      // A = propertySet

T buildSet();

```

## Building

At the top level a method `parseCharacterClass` can recognize and branch on `\p{`, `\P{`, `[`, and `^`. For `\p{` and `\P{`, it calls a `parseProperty` method that parses up to an unescaped `}`, and returns a set based on Unicode properties. See [RL1.2 Properties](#), [2.7 Full Properties](#), [RL2.7 Full Properties](#), and [2.8 Optional Properties](#).





For `[`, and `^`, it calls a `parseSequence` method that parses out items, stopping when it hits `]`. The type of each item can be determined by the initial characters. There is a special check for `-` so that it can be interpreted according to context. The `targetSet` is set to the first item. All successive items at that level are combined with the `targetSet`, according to the specified operation (union, intersection, etc.). (Note that other binding/precedence options would require somewhat more complicated parsing.)

For the Character Class item, a recursive call is made on the `parseCharacterClass` method. The other initial characters that are branched on are `\u{`, `\u`, `\q{`, `\N{`, `\`, the operators, and literal and escaped characters.

## Examples

In the following examples,  marks where the cursor is as the parsing progresses.

### Example 1

Input	Current Result	Comment
 [[abc] -- [[bcd] && [cde]]]	L0:	Start
[[a  bc] -- [[bcd] && [cde]]]	L2: [a]	
[[ab  c] -- [[bcd] && [cde]]]	L2: [ab]	
[[abc  ] -- [[bcd] && [cde]]]	L1: [a-c]	

$[[abc] \text{ -- } [[b \div cd] \ \&\& \ [cde]]]$	L3: [b]	
$[[abc] \text{ -- } [[bc \div d] \ \&\& \ [cde]]]$	L2: [bc]	
$[[abc] \text{ -- } [[bcd \div] \ \&\& \ [cde]]]$	L3: [b-d]	
$[[abc] \text{ -- } [[bcd] \div \ \&\& \ [cde]]]$	L2: [b-d]	
$[[abc] \text{ -- } [[bcd] \ \&\& \ [c \div de]]]$	L3: [c]	
$[[abc] \text{ -- } [[bcd] \ \&\& \ [cd \div e]]]$	L3: [cd]	
$[[abc] \text{ -- } [[bcd] \ \&\& \ [cde \div]]]$	L3: [c-e]	
$[[abc] \text{ -- } [[bcd] \ \&\& \ [cde] \div]]]$	L2: [cd]	
$[[abc] \text{ -- } [[bcd] \ \&\& \ [cde]]] \div]$	L0: [ab]	Final result

## Example 2

Input	Current Result	Comment
$\div[a-\backslash u\{3b1\}\backslash u\{3b3\}-\zeta]$	L0:	Start; $\alpha = U+03b1$ , $\gamma = U+303b3$
$[a \div -\backslash u\{3b1\}\backslash u\{3b3\}-\zeta]$ $\Rightarrow$	L1: [a]	
$[a-\backslash u\{3b1\} \div \backslash u\{3b3\}-\zeta]$	L1: [a-a]	
$[a-\backslash u\{3b1\}\backslash u\{3b3\} \div -\zeta]$	L1: [a-a $\gamma$ ]	
$[a-\backslash u\{3b1\}\backslash u\{3b3\}-\zeta \div]$	L1: [a-a $\gamma$ - $\zeta$ ]	
$[a-\backslash u\{3b1\}\backslash u\{3b3\}-\zeta] \div]$	L0: [a-a $\gamma$ - $\zeta$ ]	Final result

## EBNF

*This is easier to see in a side-by-side landscape format, so see [L2/21-003 UTS #18 Editorial additions: EBNF](#)*