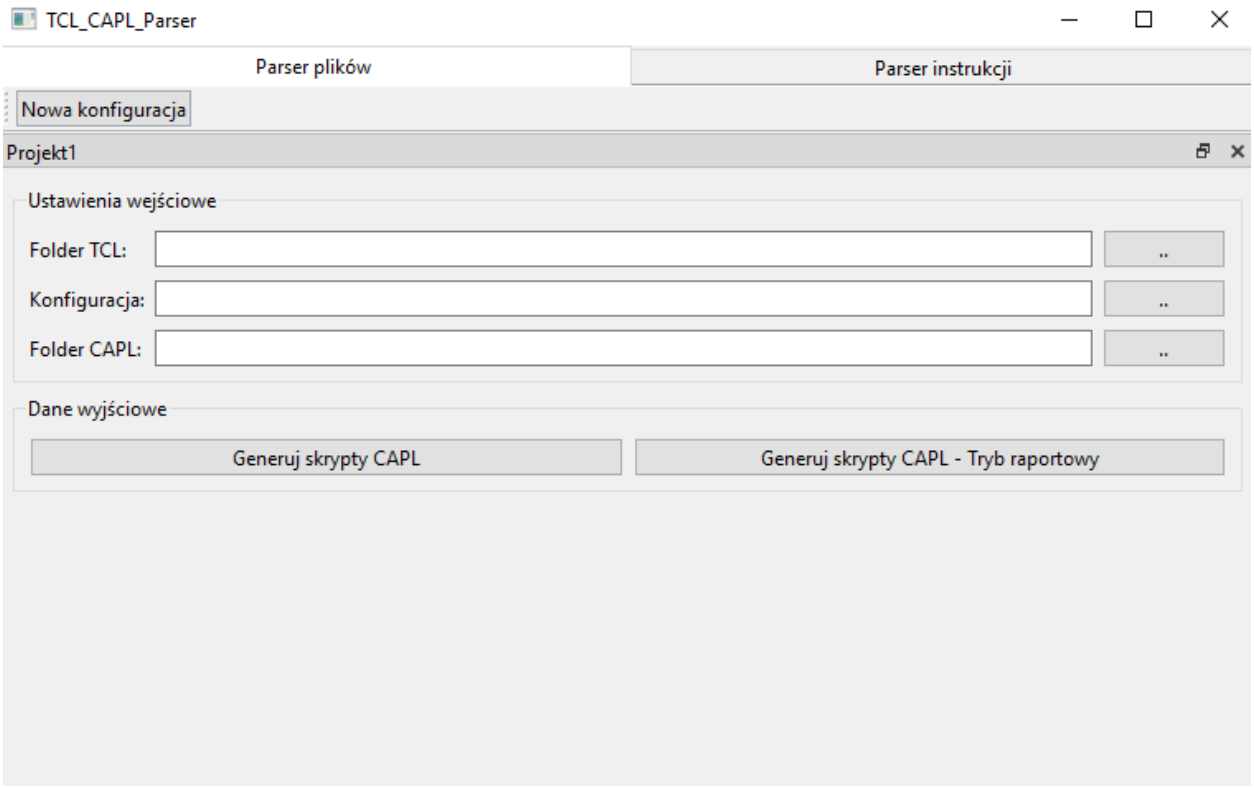
Zawartość

TCL CAPL Parser

Opis programu

Program służy do wspomagania przepisywania testcase'ów CAPL z testcase'ów TCL. Do wyboru mamy dwa tryby pracy programu:

- Parser plików – służy do wygenerowania testów CAPL dla wskazanego przez użytkownika folderu z testami TCL.



Opcja **Nowa konfiguracja** umożliwia dodanie nowej osobnej konfiguracji generowania plików CAPL.

Opcja **Folder TCL** jest wymagana do wygenerowania testów. Dla tej opcji należy wybrać folder z testami TCL (*.tc). Folder ten może zawierać inne foldery ze skryptami oraz pliki o innych rozszerzeniach.

Opcja **Konfiguracja** jest opcjonalna i pozwala na wybranie pliku *.xml z konfiguracją, która zmieni sposób generowania testów.

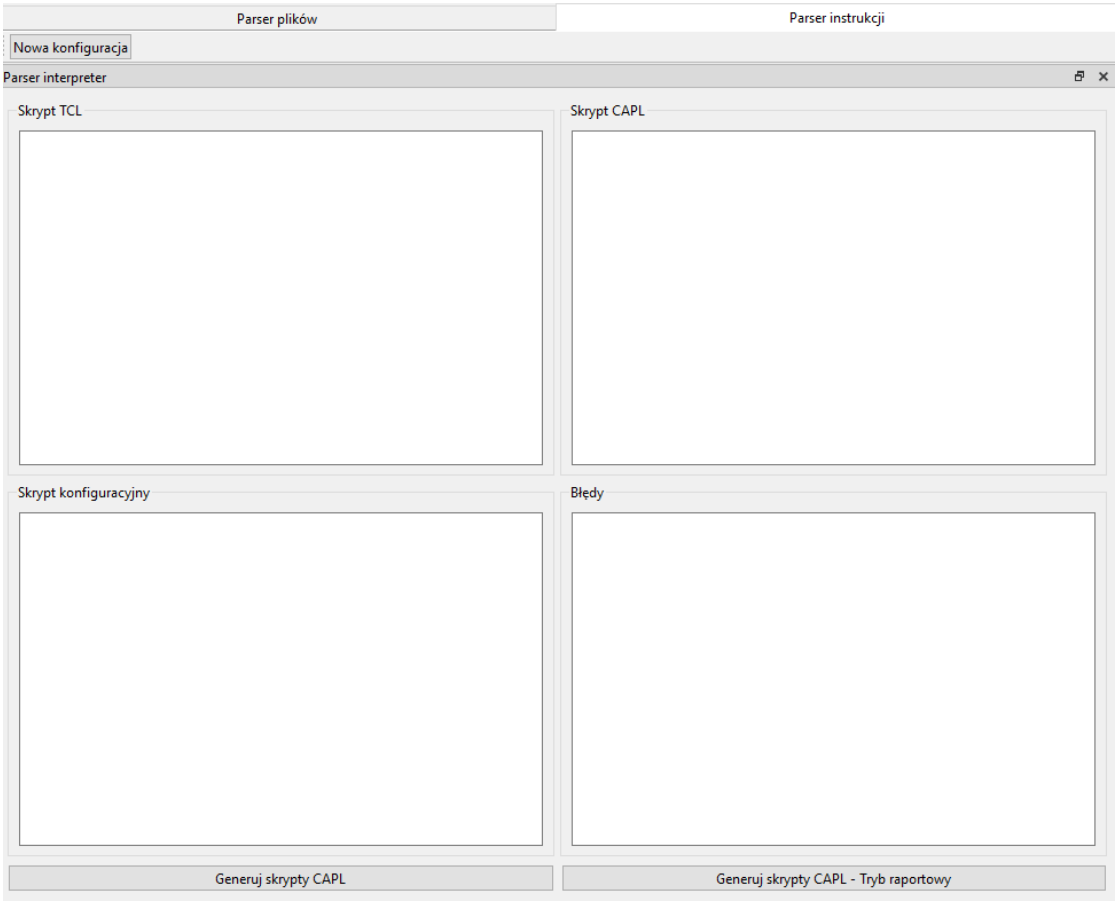
Opcja **Folder CAPL** jest opcjonalna i pozwala wybrać folder, do którego zostaną wygenerowane testy. Jeśli opcja jest pusta, pliki zostaną wygenerowane do folderu, który zawiera **folder TCL.**

Przykład – gdy opcja jest pusta:
Folder TCL: C:_workDir\myTests
Folder CAPL: C:_workDir\FAMOut1_myTests

Opcja **Generuj skrypty CAPL** spowoduje uruchomienie niezależnej procedury generowania testów na podstawie pliku konfiguracyjnego (nie wymagany). Pojawi osobne okno, w którym można obserwować postęp generowania testów.

Opcja **Generuj skrypty CAPL – Tryb raportowy** spowoduje uruchomienie niezależnej procedury generującej skrypty w trybie raportowym na podstawie pliku konfiguracyjnego. Wygenerowane skrypty będą zawierać tylko wybrane przez użytkownika procedury. W przypadku braku konfiguracji, wygenerowane testy będą zawierać głównie komentarze.

- Parser instrukcji – służy do sprawdzania jak program zinterpretuje podany przez użytkownika skrypt TCL z opcjonalnie dodaną konfiguracją.



Opcja **Nowa konfiguracja** umożliwia dodanie nowej osobnej konfiguracji do testowania skryptów.

W polu tekstowym **Skrypt TCL** można wprowadzić skrypt TCL, który chcemy przegenerować na skrypt CAPL.

W polu tekstowym **Skrypt konfiguracyjny** można wprowadzić ustawienia i reguły generowania pliku CAPL. Wymaga formatu XML tak jak w przypadku wprowadzania pliku konfiguracyjnego dla parsera plików.

W polu tekstowym **Skrypt CAPL** po generowaniu skryptu CAPL pojawi się wygenerowany skrypt CAPL.

W polu tekstowym **Błędy** pojawią się błędy w przypadku wystąpienia błędów podczas generowania CAPL'a.

Opcja **Generuj skrypty CAPL** spowoduje generowanie skryptu CAPL na podstawie skryptu TCL oraz skryptu konfiguracyjnego. Po udanym generowaniu w polu tekstowym **Skrypt CAPL** pojawi wygenerowany skrypt. W przypadku błędów generowania, w polu **Błędy** zostaną wyświetlone błędy generowania.

Opcja **Generuj skrypty CAPL – Tryb raportowy** spowoduje generowanie skryptu CAPL na podstawie skryptu TCL oraz skryptu konfiguracyjnego w trybie raportowym. Wygenerowany skrypt będzie zawierać tylko wybrane przez użytkownika procedury. W przypadku braku konfiguracji, wygenerowane testy będą zawierać głównie komentarze. Po udanym generowaniu w polu tekstowym **Skrypt CAPL** pojawi wygenerowany skrypt. W przypadku błędów generowania, w polu **Błędy** zostaną wyświetlone błędy generowania.

Ustawianie pliku konfiguracyjnego

Plik konfiguracyjny zawiera opcje pozwalające zmienić sposób generowania skryptów CAPL. W pliku można ustawić:

- Listę nazw procedur, które będą wygenerowane w trybie raportowym. Listę tę wprowadza się w tokenie ***writeOnlyFunctions***.
- Listę reguł do sterowania interpretacją i generowaniem instrukcji CAPL na podstawie instrukcji/procedur TCL.

Sterowanie interpretacją procedur

Język skryptowy TCL jest oparty na procedurach/funkcjach. Oznacza to, że jakakolwiek instrukcja jest realizowana z wykorzystaniem procedur oraz argumentów, które przekazujemy tej procedurze.

Przykład:

Przykład procedury / instrukcji TCL	*Opis*
set a 10	*Procedura o nazwie **set** z dwa argumentami: **a** oraz **10***
myFunc param1 {param2} „param3”	*Procedura o nazwie **myFunc** z trzemi argumentami: **param1**, **{param2}** oraz **„param3”***
incr a	*Procedura o nazwie **incr** z jednym argumentem: **a***
if ($a==2) {puts „True”}	*Procedura o nazwie **if** z 2 argumentami: **($a==2)** oraz **{puts „True”}***
for {set i 0} {$i<4} {incr i} {puts $i}	*Procedura **for** z 4 argumentami: **{set i 0}**, **{$i<4}**, **{incr i}** i **{puts $i}***

Jak widać na powyższych przykładach, w TCL'u każda instrukcja, nawet pętle, są traktowane jako procedury przyjmujące argumenty. Dlatego w pliku konfiguracyjnym można ustawić reguły sterujące interpretacją procedur TCL.

Procedura TCL zawiera 2 informacje: ***nazwę procedury*** oraz ***listę argumentów***, gdzie każdy argument traktowany jest jako napis. Jak na powyższych przykładach, każdy argument jest tylko napisem.

W TCL_CAPL_Parser procedura TCL zawiera te 2 wyżej wypisane informacje. Jednak każdy argument ma dodatkową informację ***stan***, która ma wspomagać interpretowanie argumentów.

Interpretacja procedur w TCL_CAPL_Parser polega na poinformowaniu programu z pomocą reguł interpretacji procedur, w jaki sposób ma zapisać instrukcję w skrypcie CAPL na podstawie odczytanej procedury TCL oraz jej argumentów.

Zadaniem reguły jest wykonanie pewnych akcji **executableAction** , jeśli zostały spełnione wszystkie akcje warunkowe **conditionalAction** zapisane dla tej reguły. Reguły te można łączyć w grupy tworząc listy reguł. Listy reguł przypisuje się do konkretnych etapów interpretacji procedury. Domyślnie, jeśli jedna reguła została spełniona, to następne reguły z listy nie są sprawdzane. Dla każdej reguły można to zmienić poprzez zmianę atrybutu ***controlFlag*** w tokenie ***rule***.

Każda akcja ma swój określony typ oraz przyjmuje własny zestaw parametrów. Akcje warunkowe ***conditionalAction*** mają sprawdzać, czy zostały spełnione określone warunki. Akcje ***executableAction*** mają wykonać pewne zadania.

Reguły można przypisać do:

- Konkretnej procedury, kiedy znamy jej nazwę,
- Nieznanych procedur, czyli takich, których nazwa nie ma żadnych przypisanych reguł

Przykład:
Załóżmy, że interpreter nie ma żadnych własnych domyślnych reguł (zakodowanych w programie). Załóżmy również, że w pliku konfiguracyjnym definiujemy reguły dla procedury ***set*** oraz reguły dla nieznanych procedur.

Instrukcja TCL	Zachowanie programu
set;	Program korzysta tylko ze zdefiniowanych reguł dla procedury ***set***
set a 2	Program korzysta tylko ze zdefiniowanych reguł dla procedury ***set***
get „test”; # *Jak i każda inna procedura z wyjątkiem* ***set***	Program korzysta tylko ze zdefiniowanych reguł dla nieznanej procedury, ponieważ założyliśmy, że reguły zdefiniowany są tylko dla procedury ***set***

W obu przypadkach, czyli zarówno zdefiniowanej procedury, jak i nieznanych procedur, reguły przypisywane są do poszczególnych **etapów interpretacji procedury**:

- Reguły na koniec wywołania procedury ***rulesOnEndOfCall***,
- Reguły dla argumentu o zadanym indeksie ***rulesForArgument***,
- Reguły dla argumentów o nieznanym indeksie ***rulesForUnspecifiedArgument***, czyli każdy, dla którego nie zostały przypisane reguły.

Reguły na koniec wywołania procedury są sprawdzane, kiedy interpreter natrafi na koniec instrukcji/procedury. Oznacza to, że znana jest nazwa procedury oraz wszystkie jej argumenty.

Reguły dla argumentu o zadanym indeksie są sprawdzane, dla każdego zdarzenia (są dwa rodzaje zdarzeń), które wystąpiło podczas interpretacji argumentu o podanym przez użytkownika indeksie. (Więcej w następnym przykładzie)

Reguły dla argumentów o nieznanym indeksie są sprawdzane, dla każdego zdarzenia (są dwa rodzaje zdarzeń), które wystąpiło podczas interpretacji argumentu o indeksie, który nie ma zdefiniowanych reguł. (Więcej w następnym przykładzie)

Zdarzenia, o których wspomniałem w regułach dla argumentów, dzielą reguły dla argumentów na dwie kategorie:

- Reguły dynamiczne ***dynamicRules***,
- Reguły podczas zapisywania argumentu do procedury ***rulesOnMove***.

Reguły dynamiczne są to reguły, które wymagają wiedzy o sposobie działania interpretera, dlatego nazwałbym je **regułami zaawansowanymi**. Reguły te sprawdzane są przy każdym wykryciu i obsłużeniu nowego stanu interpretera (wykrycie i obsługa słowa kluczowego TCL). Przykładem użycia jest procedura ***if*** , która natrafiając na słowo kluczowe **{** domyślnie zapisze nowy stan interpretera jako otwarcie listy, czyli stan ***List***. Jeśli ta lista ma reprezentować blok kodu, to zamieniam nowo utworzony stan ***List*** na stan ***CodeBlock***, który w inny sposób niż ***List*** obsłuży słowa kluczowe TCLa.

Reguły na dodanie argumentu do procedury są to reguły, które są sprawdzane w momencie zapisywania odczytanego i zatwierdzonego argumentu do wywołania procedury. Oznacza to, że znany jest argument oraz jego stan. Argument jest dodawany w momencie wykrycia następnego argumentu lub wykrycia końca instrukcji.

Przykład:

Reguła	*Instrukcja TCL*	*Opis*
Reguły dynamiczne dla argumentu o indeksie 1	*write a 2 4;*	*Procedura o nazwie **write** będzie sprawdzać reguły przy każdym nowym stanie interpretera tylko dla argumentu o indeksie 1, czyli gdy interpreter zna już argument o indeksie 0 **a** oraz gdy zatwierdził argumentu o indeksie 1 **2***
Reguły na zapisanie argumentu o indeksie 1	*write a 2 4;*	*Reguły, które są sprawdzane tylko, gdy argument o indeksie 1 został odczytany i zapisany do obiektu z procedurą. Nastąpi to w momencie odczytania średnika po **4** , ponieważ **4** zostanie rozpoznane jako następny argument (**String**) , a ostatni zapamiętany stan w tym przypadku to **2 (String)**, to **2** jest zapisana do obiektu procedury jako argument o indeksie 1, ponieważ **a** zostało wcześniej zapisane*
Reguły na zapisanie argumentu o indeksie 1 --- *Reguły na zapisanie nieznanego argumentu*	*write a 2 4;*	*Reguły jak wyżej, jednak dodatkowo wystąpi też obsługa reguł dla każdego nieznanego argumentu, czyli w tym wypadku,dla zapisania **a** oraz **4** sprawdzane są dodatkowe reguły*

Formatowane napisy

Formatowane napisy są używane w celu utworzenia napisów, które mogą zawierać nazwę procedury lub argumenty procedury w odpowiednim formacie.

Tworzy się je w przypadku użycia akcji: ***compare*** (drugi argument ***ParamsList)***, ***Write***, ***tclParse***, ***error***, ***addPreexpression***. W przypadku tych akcji, token ***Param*** służy do wprowadzania tekstu, a FormatRule do sterowania tworzeniem napisu lub pobierania argumentów procedury.

Opis tokenów

userConfig

Parent Token	**Brak**
Token główny pliku konfiguracyjnego.	
Przykład (Pusty token)	
<userConfig> // … </userConfig>	

settings

Parent Token	**userConfig**
Token zawierający ogólne ustawienia interpretera, w tym listę procedur dla trybu raportowego.	
Przykład (Pusty token)	
<userConfig> <settings> // … </settings> </userConfig>	

writeOnlyFunctions

Parent Token	settings

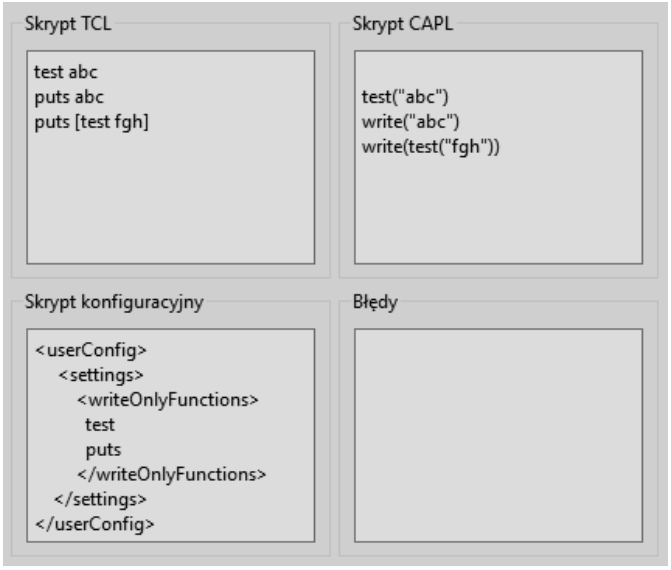
Token zawierający listę nazwy procedur, które będą generowane w trybie raportowym. Każdą nazwę procedury należy pisać w nowej linii.

Przykład

Powyższy skrypt TCL:

1. wywołuję *procedurę **test** z argumentem **abc,***
2. wywołuje procedurę **puts** z argumentem **abc**,
3. wywołuje procedurę **puts** z argumentem, który wywołuje procedurę **test** z argumentem **fgh**

W skrypcie konfiguracyjnym wprowadzona została informacja (poprzez token ***writeOnlyProcedure***), że w trybie raportowym program ma wygenerować tylko te instrukcje, które wywołują procedurę **test**. Ponieważ komenda ***puts [test fgh]*** wywołuje procedurę **test**, dlatego interpretuje całą instrukcję.



Jeśli w skrypcie konfiguracyjnym dodamy procedurę **puts** i uruchomimy generowanie w trybie raportowym otrzymamy wynik jak powyżej.

procedure

Parent Token	userConfig
Token zawiera reguły generowania procedury o nazwie podawanej w atrybucie ***name***.	
Przykład	

```
<userConfig>
        <procedure name = “set_signal”>
        // …
        </procedure>
</userConfig>
```

Powyższy zapis definiuje procedurę o nazwie “set_signal”.

defaultProcedure

Parent Token	userConfig
Token zawiera reguły generowania procedury, która nazwa nie jest zdefiniowana ani w pliku konfiguracyjnym ani wewnątrz programu.	
Przykład	

```
<userConfig>
        <defaultProcedure>
        // …
        </defaultProcedure>
</userConfig>
```

rulesForArgument

Parent Tokens	procedure defaultProcedure
Token zawiera listę reguł tylko dla pojedynczego argumentu o wskazanym w atrybucie index numerze argumentu. Numerowanie od 0.	
Przykład	

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesForArgument index = “0”>
                        // …
                </rulesForArgument>
                <rulesForArgument index = “2”>
                        // …
                </rulesForArgument>
        </procedure>
</userConfig>
```

Opis
Powyższy przykład definiuje procedurę o nazwie ***set_signal***, w której definiujemy reguły dla argumentów o indeksie ***0*** oraz ***2.***
Przykład

<userConfig>
  <defaultProcedure>
  <rulesForArgument index = “0”>
  // …
  </rulesForArgument>
  <rulesForArgument index = “2”>
  // …
  </rulesForArgument>
  </defaultProcedure>
</userConfig>

Opis
Powyższy przykład definiuje procedurę nieznaną dla niezdefiniowanej nazwy procedury, w której definiujemy reguły dla argumentów o indeksie ***0*** oraz ***2.***

rulesForUnspecifiedArgument

Parent Tokens	**procedure** **defaultProcedure**
Token zawiera listę reguł dla argumentu o nieznanym indeksie.	
Przykład	
<userConfig> <procedure name = “set_signal”> <rulesForUnspecifiedArgument> // … </rulesForUnspecifiedArgument > </procedure> </userConfig>	
Opis	
Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły dla każdego argumentu procedury, który nie ma zdefiniowanych reguł **(token *rulesForArgument*).**	
Przykład	
<userConfig> <defaultProcedure> <rulesForUnspecifiedArgument> // … </rulesForUnspecifiedArgument > </ defaultProcedure > </userConfig>	

Opis
Definiowanie nieznanej/niezdefiniowanej procedury, w której definiujemy reguły dla każdego argumentu procedury, który nie ma zdefiniowanych reguł **(**token ***rulesForArgument*****).**

rulesOnEndOfCall

Parent Tokens	**procedure** **defaultProcedure**
Token zawiera reguły, które realizowane są na zakończenie odczytu procedury. Wszystkie argumenty są wtedy znane.	
Przykład	
<userConfig> <procedure name = “set_signal”> <rulesOnEndOfCall> // … </rulesOnEndOfCall > </procedure> </userConfig>	
Opis	
Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.**	
Przykład	
<userConfig> <defaultProcedure> < rulesOnEndOfCall > // … </rulesOnEndOfCall > </ defaultProcedure > </userConfig>	
Opis	
Definiowanie nieznanej/niezdefiniowanej procedury, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.**	

dynamicRules (ZAAWANSOWANE)

Parent Tokens	**rulesForArgument** **rulesForUnspecifiedArgument**
Token zawierający reguły, które realizowane dla wystąpienia słowa kluczowego.	
Przykład	

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesForArgument index = 0>
                        <dynamicRules>
                                // …
                        </dynamicRules>
                </rulesForArgument >
        </procedure>
</userConfig>
```

Opis

Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane dla każdego słowa kluczowego (zdarzenia), które wystąpi podczas interpretacji przed wstawieniem tego argumentu do wywołania funkcji.

Przykład

```
<userConfig>
        <defaultProcedure>
                < rulesForUnspecifiedArgument >
                        <dynamicRules>
                                // …
                        </dynamicRules>
                </rulesForUnspecifiedArgument>
        </ defaultProcedure >
</userConfig>
```

Opis

Definiowanie nieznanej/niezdefiniowanej procedury, w której definiujemy reguły, które wykonywane dla każdego słowa kluczowego (zdarzenia), które wystąpi podczas interpretacji przed wstawieniem tego argumentu do wywołania funkcji.

rulesOnMove

Parent Tokens	**rulesForArgument** **rulesForUnspecifiedArgument**

Token zawierający reguły, które realizowane zaraz przed wstawieniem argumentu do wywołania funkcji. Argument jest znany.

Przykład

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesForArgument index = 0>
                        <rulesOnMove>
                                // …
                        </rulesOnMove>
                </rulesForArgument >
        </procedure>
</userConfig>
```

Opis
Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane zaraz przed wstawieniem argumentu do wywołania procedury.
Przykład
<userConfig>   <defaultProcedure>   < rulesForUnspecifiedArgument >   <rulesOnMove>   // …   </rulesOnMove>   </rulesForUnspecifiedArgument>   </ defaultProcedure > </userConfig>
Opis
Definiowanie nieznanej/niezdefiniowanej procedury, w której definiujemy reguły które wykonywane zaraz przed wstawieniem argumentu do wywołania procedury.

rule

Parent Token	rule
Token zawierający akcje (token executableAction) , które są realizowane jeśli wszystkie warunki zostaną spełnione(token conditionalAction). Dodatkowo atrybutem controlFlag można zdecydować, czy dla spełnionej reguły ma być sprawdzana kolejna reguła, czy sprawdzanie reguł ma być przerwane (domyślna opcja).	
Przykład	
<userConfig> <procedure name = “set_signal”> <rulesOnEndOfCall> <rule> // … </rule> </rulesOnEndOfCall > </procedure> </userConfig>	
Opis	
Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.** W regułach tych zdefiniowana została jedna reguła z atrybutem ***controlFlag*** z domyślną opcją.	
Przykład	
<userConfig> <procedure name = “set_signal”> <rulesOnEndOfCall>	

```
                    <rule controlFlag = “NoBreakRuleCheck”>
                        // …
                    </rule>
                </rulesOnEndOfCall >
        </procedure>
</userConfig>
```

Opis

Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.** W regułach tych zdefiniowana została jedna reguła z atrybutem ***controlFlag*** z wartością „NoBreakRuleCheck”. Oznacza to, że nawet gdyby reguła została wykonana, to interpreter przejdzie do sprawdzania następnej regułę.

conditionalAction

Parent Token	**rule**

Token zawierający typ akcji (atrybut type) oraz jej parametry. Zawiera akcje warunkowe.

Przykład

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesOnEndOfCall>
                        <rule>
                                <conditionalAction type = “CompareNumbOfArguments”>
                                        <param>
                                                0
                                                2
                                                5
                                        </param>
                                </conditionalAction>
                        </rule>
                </rulesOnEndOfCall >
        </procedure>
</userConfig>
```

Opis

Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.** W regułach tych zdefiniowana została jedna reguła z atrybutem ***controlFlag*** z domyślną wartością. Reguła ta składa się z jednej akcji warunkowej (token ***conditionalAction***) z typem „CompareNumbOfArguments”. Warunek ten sprawdza aktualną ilość argumentów wywołania procedury i porównuje z wartościami z listy argumentów, by zatwierdzić warunek lub nie. Jeśli warunek został spełniony, zostaną wykonane wszystkie akcje (***executableAction***).

Przykłady poleceń TCL, które spełnią warunek

```
set_signal ; # Bez argumentów
set_signal „abc” 24; # Dwa argumenty, to samo dotyczyłoby 5 argumentów.
```

executableAction

Parent Token	rule
Token zawierający typ akcji (atrybut type) oraz jej parametry. Akcje te zostaną wykonane, jeśli wszystkie akcje warunkowe zostaną spełnione.	
Przykład	

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesOnEndOfCall>
                        <rule>
                                <executableAction type = “Error”>
                                        <param>Mój błąd</param>
                                </executableAction>
                        </rule>
                </rulesOnEndOfCall >
        </procedure>
</userConfig>
```

Opis

Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.** W regułach tych zdefiniowana została jedna reguła z atrybutem ***controlFlag*** z domyślną wartością. Reguła ta składa się z jednej akcji wykonywalnej (token **executable*Action***) z typem „Error”. Akcja ta spowoduje wywołanie błędu interpretera z wprowadzonym przez użytkownika komunikatem (wynik pojawi się w panelu wyjściowym **Błędy**).

param

Parent Tokens	**conditionalAction** **executableAction** **paramsList**
Token zawierający zwykły tekst przekazywany do reguły jako jej parametr. Nie może być pusty.	
Przykład: Patrz *conditionalAction* **lub** *executableAction*	

formatParam

Parent Token	paramsList
Token używany do tworzenia sformatowanych napisów. Należy podać typ akcji formatującej (atrybut type), a następnie jej argumenty.	
Przykład	

```
<userConfig>
        <procedure name = “set_signal”>
                <rulesOnEndOfCall>
```

```
                    <rule>
                        <conditionalAction type = "CompareNumbOfArguments">
                            <param>2</param>
                        </conditionalAction>
                        <executableAction type = "Write">
                            <param>$</param>
                            <formatRule type = "Target">Command</formatRule>
                            <formatRule type = "NameOrIndex">0</formatRule>
                            <param> = </param>
                            <formatRule type = "ArgumentsFrom">1</formatRule>
                        </executableAction>
                    </rule>
                </rulesOnEndOfCall >
            </procedure>
</userConfig>
```

Opis

Definiowanie procedury o nazwie ***set_signal***, w której definiujemy reguły, które wykonywane są na zakończenie odczytu procedury**.** W regułach tych zdefiniowana została jedna reguła z atrybutem ***controlFlag*** z domyślną wartością. Reguła ta składa się z jednej akcji warunkowej „CompareNumbOfArguments” oraz jednej akcji wykonywalnej „Write”. Akcja ta spowoduje wygenerowanie podanego w argumentach tekstu, jeśli tylko procedura *set_signal* będzie wywoływana 2 argumenty.

Wynik działania reguły

Skrypt TCL

```
set_signal MY_SIGNAL 25
# Oczywiscie zadziała też dla innych "niepoprawnych" zapisów
set_signal 24 {fgh}
set_signal [myFunc 1 2 3] [myFunc]
# Ale tylko dla wywołania z 2 argumentami - ponieważ jest tylko jedna reguła
set_signal
set_signal MY_SIGNAL
set_signal 2 \
"abc" {fgh}
```

Skrypt CAPL

```
$MY_SIGNAL = 25
// Oczywiscie zadziała też dla innych "niepoprawnych" zapisów
$24 = fgh
$myFunc(1, 2, 3) = myFunc()
// Ale tylko dla wywołania z 2 argumentami - ponieważ jest tylko jedna reguła
```

Skrypt konfiguracyjny

```
<userConfig>
    <procedure name = "set_signal">
        <rulesOnEndOfCall>
            <rule>
                <conditionalAction type = "CompareNumbOfArguments">
                    <param>2</param>
                </conditionalAction>
                <executableAction type = "Write">
                    <param>$</param>
                    <formatRule type = "Target">Command</formatRule>
                    <formatRule type = "NameOrIndex">0</formatRule>
                    <param> = </param>
                    <formatRule type = "ArgumentsFrom">1</formatRule>
                </executableAction>
            </rule>
        </rulesOnEndOfCall >
    </procedure>
</userConfig>
```

Błędy

Akcje warunkowe (conditionalAction)

compareNumbOfArguments

Parametry wejściowe
Opis
Akcja pozwalająca porównać ilość argumentów procedury. Akcja wymaga podania parametru w postaci <param>. Parametr przyjmuje listę liczb oddzielone nową linią. Jeśli liczba argumentów wywołanej procedury jest zgodna z przynajmniej 1 liczbą podanej w liście, akcja warunkowa jest spełniona.
Sposób użycia
<conditionalAction type=”compareNumbOfArguments”> <param> 0 1 … </param> </conditionalAction>
Przykład
<conditionalAction type=”compareNumbOfArguments”> <param> 0 1 </param> </conditionalAction> Podana akcja zostanie spełniona, jeśli aktualna procedura nie ma argumentów lub ma tylko jeden argument.
Przykłady instrukcji
myProcedure; # Akcja spełniona bez argumentów *myProcedure 2; # Akcja spełniona z 1 argumentem* *myProcedure 2 3; # Akcja niespełniona, ponieważ procedura została wywołana z 2 argumentami* *# Powyższa akcja zostanie jednak spełniona, jeśli akcja została użyta dla tokenów „dynamicRules” oraz „rulesOnMove” i jeśli nie został jeszcze dodany 2 argument*

isLastStat (ZAAWANSOWANE)

Parametry wejściowe
Opis
Akcja sprawdzająca ostatni stan interpretera Akcja wymaga podania parametru w postaci <paramsList>. Dla tej akcji, <paramsList> przyjmuje tylko tokeny <param>. Jeśli ostatni wywołany stan interpretera jest zgodny z przynajmniej jednym stanem podanym w liście, akcja warunkowa jest spełniona.
Sposób użycia
<conditionalAction type=”isLastStat”> <paramsList> <param>List</param> <param>EndOfList</param> … </paramsList> </conditionalAction>

Aby sprawdzić w jakim celu i jak używać akcji, sprawdź ***Domyślne reguły interpretera*** – **if** lub **for**.

Compare

Parametry wejściowe
Opis
Akcja porównuje listę napisów do napisu sformatowanego. Pierwszy argument to lista napisów, z którą porównujemy sformatowany napis (drugi argument).
Sposób użycia
<conditionalAction type=”compare”> <paramsList> Pierwszy argument – lista napisów <param> <param>CAN1</param> <param>LIN3</param> </paramsList> <paramsList> Drugi argument – lista parametrów, które sformatują napis <param>CAN1</param> </paramsList> </conditionalAction> Dla powyższego przypadku akcja jest zawsze spełniona.

Akcje (executableAction)

Write

Parametry wejściowe
Opis
Najważniejsza akcja pozwalająca zapisać argument lub całe wywołanie funkcji z pomocą sformatowanego napisu. Przyjmuje tylko jeden argument <paramsList> składającą się z tokenów <param> do zapisu zwykłego tekstu oraz <formatRule> do sterowania formatowaniem tekstu. Sposób działania tej akcji zależy od tokenów, w które została wpisana. Dla tokenów: • <rulesForArgument> lub <rulesForUnspecifiedArgument> - napis jest zapisywany zamiast w miejsce argumentu, • <rulesOnEndOfCall> - napis jest zapisywane w miejsce całej instrukcji.
Sposób użycia
<rulesForArgument index=”0”> Można również użyć <rulesForUnspecifiedArgument> <rule> <executableAction type=”write”> <paramsList> Lista parametrów: <param> oraz <formatRule> <param>CAN1</param> </paramsList> </executableAction> </rule> </rulesForArgument>
Dla instrukcji
proc 2 4; # Wynik generowania: proc(„CAN1” , 4)
***Jeśli zmienimy* <rulesForArgument index=”0”> na <rulesForUnspecifiedArgument>**
proc 2 4; # Wynik generowania: proc(„CAN1”, „CAN1”)

tclParse

Parametry wejściowe
Opis
Akcja przekazuje do interpretera sformatowany tekst, a wynik zapisuje jak akcja Write.
Sposób użycia
<rulesForArgument index=”0”> <rule> <executableAction type=”tclParse”>

```
                        <paramsList>    Lista parametrów: <param> oraz <formatRule>
                                <param>CAN1</param>
                        </paramsList>
                </executableAction>
        </rule>
</rulesForArgument>
```

Dla instrukcji
proc 2 4; # Wynik generowania: proc(CAN1() , 4)
***Jeśli zmienimy* <rulesForArgument index="0"> na <rulesForUnspecifiedArgument>**
proc 2 4; # Wynik generowania: proc(CAN1(), CAN1())
Dla tokena <rulesOnEndOfCall>
proc 2 4; # Wynik generowania: CAN1()

error

Parametry wejściowe
Opis
Akcja pozwala wymusić pojawienie się błędu interpretera. Służy głównie do debugownia.
Sposób użycia

```
<executableAction type="error">
        <paramsList>    Lista parametrów: <param> oraz <formatRule>
                <param>CAN1</param>
        </paramsList>
</executableAction>
```

addPreexpression

Parametry wejściowe
Opis
Akcja wprowadza dodatkowe sformatowany tekst, które pojawi się przed procedurą dla której została ta akcja wywołana.
Sposób użycia

```
<executableAction type="addPreexpression">
        <paramsList>    Lista parametrów: <param> oraz <formatRule>
                <param>CAN1</param>
        </paramsList>
</executableAction>
```

Dla instrukcji
proc 2
Wynik generowania
CAN1 *proc(2)*

addUserInteraction

Parametry wejściowe
Opis
Akcja dodaje znacznik dla użytkownika informujący o potrzebie ręcznej modyfikacji.
Sposób użycia
<executableAction type=”addUserInteraction”>   <paramsList> Lista parametrów: <param> oraz <formatRule>   <param>CAN1</param>   </paramsList> </executableAction>

addPredefinition

Parametry wejściowe
Opis
Akcja dodaje definicje zmiennej na początku bloku kodu, w którym się znajduje.
Sposób użycia
<executableAction type=”addPredefinition”>   <paramsList> Lista parametrów: <param> oraz <formatRule>   <param>CAN1</param>   </paramsList> </executableAction>

Parametr formatujący FormatRule

NameOrIndex

Opis

FormatRule z atrybutem NameOrIndex umożliwia pobranie nazwy procedury lub wartości dowolnego argumentu wskazując numer argumentu indeksując od 0. Jeśli wprowadzimy liczbę mniejszą od 0, pobrana zostanie wartość argumentu indeksowana od ostatniego argumentu do pierwszego. Jeśli nie podamy indeksu, pobierana jest nazwa procedury.
Przykład

ArgumentsFrom

Opis
FormatRule z atrybutem ArgumentsFrom umożliwia pobranie wartości argumentów iterując od numeru argumentu indeksując od 0. Jeśli wprowadzimy liczbę mniejszą od 0, pobrana zostanie wartość argumentu indeksowana od ostatniego argumentu do pierwszego. Domyślny separator jest pusty i resetowany po każdym użyciu tej reguły. Aby ustawić separator, użyj tokenu FormatRule z atrybutem o wartości Separator.
Przykład

Separator

Opis
FormatRule z atrybutem Separator ustawia tekst, który będzie pojawiał się pomiędzy kolejnymi argumentami w następnym wywołaniu FormatRule z atrybutem ArgumentsFrom.
Przykład

Target

Opis
FormatRule z atrybutem Target ustawia format w jakim zostaną zapisane argumenty. Każdy argument procedury ma swój status. W zależności od tego statusu oraz wybranego formatu (Target) zapisywana jest zawartość argumentu procedury.

Wartość Target'u	***Stany interpretera***	***Opis***
TclFormat	**Variable**	*Format zapisu: $<command>*
	EndOfList	*Opis pod tabelą*

	SpeechMark ***StringInQuotes***	*Format zapisu: "<command>"*
	FunctionCall	Łączy wszystkie argumenty z domyślnym separatorem. Każdy argument zapisywany jest z formatem Tcl. Dodatkowo dodaje na początku i na końcu nawiasy kwadratowe.
	Snprintf	Łączy wszystkie argumenty z domyślnym separatorem. Każdy argument zapisywany jest z formatem Tcl. Dodatkowo dodaje na początku i na końcu cudzysłowy.
	PendingSnprintf	Łączy wszystkie argumenty z domyślnym separatorem. Każdy argument zapisywany jest z formatem Tcl.
	EndOfExpression **EndOfCodeBlock**	Zwraca błąd, ponieważ nie jest zaimplementowane
	Wszystkie pozostałe stany zapisane jak dla Command	
CaplFormat – Domyślna wartość	*SpeechMark* *String* *PendingString* *StringInQuotes*	*Format zapisu: "<command>"*
	EndOfCodeBlock	*Format zapisu: {<command>}*
	EndOfList	*Opis pod tabelą*
	Wszystkie pozostałe stany zapisane jak dla Command	
ParametersStat	**Tylko FormatRule – NameOrIndex (wersja z argumentem)**	Zwraca nazwę stanu argumentu procedury
Command	Zwraca zawartość argumentu procedury tak jak jest zapamiętana przez interpreter.	

Przykład

Atrybuty

userInteraction

Token
procedure
Wartości
1. *NotRequired* – **Wartość domyślna** 2. *Required*
Opis
Każda procedura z tym atrybutem o wartości ***Required*** spowoduje dodanie znaczników informujących użytkownika o potrzebie ręcznej zmiany w kodzie.

controlFlag

Token
rule
Wartości
1. *BreakRuleCheck* – **Wartość domyślna** **2.** *NoBreakRuleCheck*
Opis
Każda reguła z tym atrybutem o wartości ***BreakRuleCheck*** przerwanie wykonywania kolejnych reguł, jeśli wszystkie akcje warunkowe dla reguły z tym atrybutem zostały spełnione. Jeśli atrybut ma wartość ***NoBreakRuleCheck*** *, to niezależnie od tego, czy akcje warunkowe zostały spełnione czy nie, następna reguła zostanie interpretowana.*
Przykład

Domyślne reguły interpretera

Kolejność zapisu reguł dla poszczególnych procedur: od najmniej do najbardziej złożonych.
Każdy zbyt długi zapis oznaczony jest zapisem <STR-*IDnumber*>, by ograniczyć nieczytelność zapisu. Napisy te zapisane są w podrozdziale ***Za długie napisy*** (na koniec rozdziału)**.**

Set

Zapis do pliku konfiguracyjnego

```
<procedure name=”set”>
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type = „CompareNumbOfArguments”>
                                <param>
                                        0
                                        1
                                </param>
                        </conditionalAction>
                        <executableAction type=”Error”>
                                <paramsList>
                                        <param>STR-1</param>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <executableAction type=”addPredefinition” />
                        <executableAction type=”write”>
                                <formatRule type=”Target”>Command</formatRule>
                                <formatRule type=”NameOrIndex”>0</formatRule>
                                <formatRule type=”Target”>CaplFormat</formatRule>
                                <param> = </param>
                                <formatRule type=”Separator”> </formatRule>  Separator == Spacja
                                <formatRule type=”ArgumentsFrom”>1</formatRule>
                        </executableAction>
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła wystawi błąd STR-1 , jeżeli liczba argumentów procedury będzie równa 0 lub 1.*
- *Druga reguła wykona się, jeśli pierwsza reguła nie zostanie spełniona. Doda ona definicję dla zmiennej oraz zapisze wyrażenie w innej postaci.*

Przykład

set; -> BŁAD
set a; -> BŁAD
set a 2; -> char a[128] a=2

set a 2 + 2 -> char a[128] a = 2 + 2

Continue

Zapis do pliku konfiguracyjnego

```
<procedure name="continue">
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type="CompareNumbOfArguments">
                                <param>0</param>
                        </conditionalAction>
                        <executableAction type="Write">
                                <paramsList>
                                        <param>continue</param>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <executableAction type="Error">
                                <paramsList>
                                        <param>STR-2</param>
                                </paramsList>
                        </executableAction>
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła zapisze wyrażenie continue , jeśli procedura nie będzie miała argumentów.*
- *Druga reguła wykona się tylko, jeśli pierwsza reguła nie zostanie spełniona.*
- *Druga reguła zgłosi błąd z komunikatem **STR-2**.*

Przykład

continue -> continue (bez reguły zostałby zapisany continue())
continue 1 -> BŁĄD

Return

Zapis do pliku konfiguracyjnego

```
<procedure name="return">
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type="CompareNumbOfArguments">
```

```
                        <param>0</param>
                    </conditionalAction>
                    <executableAction type="Write">
                        <paramsList>
                            <param>return</param>
                        </paramsList>
                    </executableAction>
                </rule>
                <rule>
                    <executableAction type="Write">
                        <paramsList>
                            <param>return </param>
                            <formatRule type="NameOrIndex">0</formatRule>
                        </paramsList>
                    </executableAction>
                </rule>
            </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła zapisze wyrażenie return , jeśli procedura nie będzie miała argumentów.*
- *Druga reguła wykona się tylko, jeśli pierwsza reguła nie zostanie spełniona.*
- *Druga reguła zgłosi błąd z komunikatem return <argument o indeksie 0>.*

Przykład

return -> return (bez reguły zostałby zapisany return())
return 1 + 2 -> return 1 (TCL naprawdę dziwnie reaguje na return, będę jeszcze dopracowywał tą regułę)
return 1 2 -> return 1 (TCL natomiast zwraca nic nie zwraca, tak jakby użyto return)

Delay

Zapis do pliku konfiguracyjnego

```
<procedure name="delay">
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type="CompareNumbOfArgument">
                            <param>1</param>
                        </conditionalAction>
                        <executableAction type="Write">
                            <paramsList>
                                <param>testWaitForTimeout(</param>
                                <formatRule type="NameOrIndex">0</formatRule>
                                <param>)</param>
                            </paramsList>
                        </executableAction>
                </rule>
```

```
            <rule>
                <executableAction type="Error">
                    <paramsList>
                        <param>STR-3</param>
                    </paramsList>
                </executableAction>
            </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła zapisze wyrażenie testWaitForTimeout <arg1> , jeśli procedura ma 1 argument.*
- *Druga reguła wykona się tylko, jeśli pierwsza reguła nie zostanie spełniona.*
- *Druga reguła zgłosi błąd z komunikatem STR-3.*

Przykład

```
delay -> BŁAD
delay 1 + 2 -> BLAD
delay 1000 -> waitForTimeout(1000)
delay abc -> waitForTimeout(„abc")
delay [expr 1000 + 2000] -> waitForTimeout(1000 + 2000)
```

Call

Zapis do pliku konfiguracyjnego

```
<procedure name="call">
    <rulesOnEndOfCall>
        <rule>
            <conditionalAction type="CompareNumbOfArgument">
                <param>0</param>
            </conditionalAction>
            <executableAction type="Error">
                <paramsList>
                    <param>STR-4</param>
                </paramsList>
            </executableAction>
        </rule>
        <rule>
            <executableAction type="Write">
                <paramsList>
                    <formatRule type="Target">TclFormat</formatRule>
                    <formatRule type="NameOrIndex">0</formatRule>
                    <formatRule type="Target">CaplFormat</formatRule>
                    <param>(</param>
                    <formatRule type="Separator">, </formatRule>
                    <formatRule type="ArgumentsFrom">1</formatRule>
```

```
                                        <param>)</param>
                                    </paramsList>
                                </executableAction>
                            </rule>
                        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła zwróci błąd STR-4 , jeśli procedura nie ma argumentów.*
- *Druga reguła wykona się tylko, jeśli pierwsza reguła nie zostanie spełniona.*
- *Druga reguła zapisze sformatowany napis.*
  - *Argument o indeksie 0 w formacie TCL (tak jak został odczytany)*
  - *Napis (*
  - *Wszystkie argumenty w formacie CAPL odseparowane poprzez tekst <przecinek><spacja> (zapisane dla przejrzystości)*
  - *Napis)*

Przykład

call -> BŁAD
call Procedure 2 4 -> Procedure(2, 4)
call Procedure -> Procedure()
call Procedure abc $var -> Procedure("abc", var)

Open

Zapis do pliku konfiguracyjnego

<procedure name="open" userInteraction="Required" />

Opis

- *Zapisuje znaczniki informujące o miejscu w kodzie, które wymaga manualnej zmiany przez użytkownika*

Przykład

*_Uzupełnij_*

Close

Zapis do pliku konfiguracyjnego

<procedure name="close" userInteraction="Required" />

Opis

- *Zapisuje znaczniki informujące o miejscu w kodzie, które wymaga manualnej zmiany przez użytkownika*

Przykład

*_Uzupełnij_*

Eval

Zapis do pliku konfiguracyjnego

<procedure name="eval" userInteraction="Required" />

Opis

- *Zapisuje znaczniki informujące o miejscu w kodzie, które wymaga manualnej zmiany przez użytkownika*

Przykład

*_Uzupełnij_*

Expr

Zapis do pliku konfiguracyjnego

```
<procedure name="expr">
        <rulesOnEndOfCall>
                <rule>
                        <executableAction type="TclParse">
                                <paramsList>
                                        <param>expr_parser</param>
                                        <formatRule type="Target">TclFormat</formatRule>
                                        <formatRule type="ArgumentsFrom">0</formatRule>
                                </paramsList>
                        </executableAction>
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Reguła wywoła interpreter w celu ponownego przeparsowania wyrażenia dla procedury. Tym razem wykorzystuję procedurę expr_parser.*
- *Ciekawostka. Procedura expr w TCL zawsze wykonuje podwójne parsowanie. Pierwsze przy odczycie argumentów procedury, a następnie parsuje ponownie już wewnątrz procedury. ☺*

Przykład

expr 2 + 2 -> Parsowanie ponowne expr_parser 2 + 2 -> 2 + 2
expr przyjmuje 3 argumenty - > expr_parser przyjmuje 3 argumenty

expr {2 + 2} -> Parsowanie ponowne expr_parser 2 + 2 -> 2 + 2
expr przyjmuje 1 argument - > expr_parser przyjmuje 3 argumenty, ponieważ lista w formacie TCL jest jak String 2 + 2

Expr_parser

Zapis do pliku konfiguracyjnego
<procedure name="expr_parser"> *<rulesOnEndOfCall>* *<rule>* *<executableAction type="Write">* *<paramsList>* *<formatRule type="Separator"> </formatRule>* *<formatRule type="ArgumentsFrom">0</formatRule>* *</paramsList>* *</executableAction>* *</rule>* *</rulesOnEndOfCall>* *</procedure>*
Opis
• *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>* • *Reguła zapisuje zapisuje wszystkie argumenty procedury w formacie CAPL z separatorem <spacja>*
Przykład
expr_parser 2 + 2 - > 2 + 2

String

Zapis do pliku konfiguracyjnego
<procedure name="string"> *<rulesOnEndOfCall>* *<rule>* *<conditionalAction type="Compare">* *<paramsLIst>* *<param>concat</param>* *</paramsList>* *<paramsList>* *<formatRule type="Target">Command</formatRule>* *<formatRule type="NameOrIndex">0</formatRule>* *</paramsList>* *</conditionalAction>* *<executableAction type="AddSnprintf"/>* PRYWATNA – Niedostępna z pliku *</rule>*

```
            <rule>
                <executableAction type="Write">
                    <paramsList>
                        <formatRule type="NameOrIndex" /> PUSTY
                        <param>(</param>
                        <formatRule type="Separator">, </formatRule>
                        <formatRule type="ArgumentsFrom">0</formatRule>
                        <param>)</param>
                    </paramsList>
                </executableAction>
                <executableAction type="AddFunctionDefinition" />
            </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Reguła potrzebna do tworzenia napisów zawierających wywołania funkcji lub odwołania do zmiennych , jeśli pierwszym argumentem funkcji jest concat.*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła powoduje zapisanie procedury w postaci:*
    - *Pobranie nazwy procedury z wykorzystaniem pustego argumentu <formatRule type="NameOrIndex" />*
    - *Napis (*
    - *Wypisanie argumentów od indeksu 0 w domyślnym formacie CAPL z separatorem <przecinek><spacja>*
    - *Napis)*

Przykład

File

Zapis do pliku konfiguracyjnego

```
<procedure name="file">
        <rulesOnEndOfCall>
            <rule>  <- Rule1 ->
                <conditionalAction type="CompareNumbOfArguments">
                    <param>2</param>
                </conditionalAction>
                <conditionalAction type="Compare">
                    <paramsList>
                        <param>size</param>
                    </paramsList>
                    <paramsList>
                        <formatRule type="Target">TclFormat</formatRule>
                        <formatRule type="NameOrIndex">0</formatRule>
                    </paramsList>
                </conditionalAction>
                <executableAction type="Write">
```

```
                        <param>fileSize(</param>
                        <formatRule type="NameOrIndex">1</formatRule>
                        <param>)</param>
                    </executableAction>
                </rule>
                <rule> <- Rule 2 ->
                    <conditionalAction type="CompareNumbOfArguments">
                        <param>2</param>
                    </conditionalAction>
                    <conditionalAction type="Compare">
                        <paramsList>
                            <param>exists</param>
                        </paramsList>
                        <paramsList>
                            <formatRule type="Target">TclFormat</formatRule>
                            <formatRule type="NameOrIndex">0</formatRule>
                        </paramsList>
                    </conditionalAction>
                    <executableAction type="Write">
                        <paramsList>
                            <param>fileExists(</param>
                            <formatRule type="NameOrIndex">1</formatRule>
                            <param>)</param>
                        </paramsList>
                    </executableAction>
                </rule>
                <rule>  <- Rule 3 ->
                    <executableAction type="AddUserInteraction"/>
                </rule>
            </rulesOnEndOfCall>
</procedure>
```

Opis

- Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>
- Pierwsza reguła sprawdza, czy procedura ma tylko 2 argumenty oraz czy pierwszy argument jest size . Jeśli warunki są spełnione, całe wyrażenie zastąpione jest sformatowanym napisem:
  - Napis fileSize(
  - Wstawienie drugiego argumentu (index = 1) w formacie CAPL
  - Napis)
- Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.
- Druga reguła sprawdza, czy procedura ma tylko 2 argumenty oraz czy pierwszy argument jest exists . Jeśli warunki są spełnione, całe wyrażenie zastąpione jest sformatowanym napisem:
  - Napis fileExists(
  - Wstawienie drugiego argumentu (index = 1) w formacie CAPL
  - Napis)
- Trzecia reguła jest sprawdzana, jeśli druga reguła nie została spełniona.
- Trzecia reguła wystawia znaczniki manualnej modyfikacji kodu.

Przykład

```
file size ..\Data\file.txt  -> fileSize(„..\Data\file.txt")
file exists {..\file.txt} -> fileExists("..\file.txt")
```

Incr

Zapis do pliku konfiguracyjnego

```
<procedure name=”incr”>
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type=”CompareNumbOfArguments”>

                                <param>2 </param>
                        </conditionalAction>
                        <executableAction type=”Write”>
                                <paramsList>
                                        <formatRule type=”NameOrIndex”>0</formatRule>
                                        <param> += </param>
                                        <formatRule type=“NameOrIndex”>1</formatRule>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                         <conditionalAction type=”CompareNumbOfArguments”>

                                <param>1 </param>
                        </conditionalAction>
                        <executableAction type=”Write”>
                                <paramsList>
                                        <formatRule type=”NameOrIndex”>0</formatRule>
                                        <param>++</param>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <executableAction type=”AddUserInteraction” />
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis ++*

Przykład

Incr a 5 -> a += 5
Incr a -> a++

Llength

Zapis do pliku konfiguracyjnego

```
<procedure name="llength">
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type="CompareNumbOfArguments">

                                <param>1</param>
                        </conditionalAction>
                        <executableAction type="Write">
                                <paramsList>
                                        <param>elcount(</param>
                                        <formatRule type="NameOrIndex">0</formatRule>
                                        <param>)</param>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <executableAction type="AddUserInteraction" />
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*

Przykład

Llength array -> elcount(array)

Puts

Zapis do pliku konfiguracyjnego

```
<procedure name="puts">
        <rulesOnEndOfCall>
                <rule>
                        <conditionalAction type="CompareNumbOfArguments">

                                <param>1 </param>
```

```
                    </conditionalAction>
                    <executableAction type="Write">
                        <paramsList>
                            <param>write(</param>
                            <formatRule type="NameOrIndex">0</formatRule>
                            <param> ) </param>
                        </paramsList>
                    </executableAction>
                </rule>
                <rule>
                    <executableAction type="AddUserInteraction" />
                </rule>
            </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*

Przykład

Puts "abc" -> write("abc")
Puts 5 -> write(5)
Puts qwer -> write("qwer")

Stc_section

Zapis do pliku konfiguracyjnego

```
<procedure name="stc_section">
    <rulesForArgument index="0">
        <rulesOnMove>
            <rule>
                <conditionalAction type="Compare">
                    <paramsList>
                        <param>RESULT</param>
                    </paramsList>
                    <paramsList>
                        <formatRule type="Target">Command</formatRule>
                        <formatRule type="NameOrIndex">-1</formatRule>
                    </paramsList>
                </conditionalAction>
                <executableAction type="Write">
                    <paramsList>
                        <param>EXPECTED</param>
                    </paramsList>
                </executableAction>
```

```
                    </rule>
                </rulesOnMove>
        </rulesForArgument>

        <rulesOnEndOfCall>
                <rule>
                        <executableAction type="Write">
                                <paramsList>
                                        <formatRule type="Target">TclFormat</formatRule>
                                        <formatRule type="NameOrIndex">0</formatRule>
                                        <param>(</param>
                                        <formatRule type="Target">CaplFormat</formatRule>
                                        <formatRule type="NameOrIndex">1</formatRule>
                                        <param>)</param>
                                </paramsList>
                        </executableAction>
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis ++*

Przykład

Incr a 5 -> a += 5
Incr a -> a++

If

Zapis do pliku konfiguracyjnego

```
<procedure name="if">
        <rulesForArgument index = „0"> ARGUMENT 1 (Index = 0)
                <dynamicRules>
                        <rule>
                                <conditionalAction type="IsLastStat">
                                        <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                                        <paramsList>
                                                <param>List</param>
```

```
                                        <param>EndOfList</param>
                                        <param>FunctionCall</param>
                                        <param>Whitespace</param>
                                        <param>SpecialSign</param>
                                        <param>Operator</param>
                                        <param>PendingSnprintf</param>
                                        <param>PendingString</param>
                                    </paramsList>
                                </conditionalAction>
                            </rule>
                            <rule>
                                <executableAction type="Error">
                                    <paramsList>
                                        <param>STR-6</param>
                                    </paramsList>
                                </executableAction>
                            </rule>
                        </dynamicRules>

                        <rulesOnMove>
                            <rule>
                                <executableAction type="TclParse">
                                    <paramsList>
                                        <param>expr_parser </param>
                                        <formatRule type="Target">TclFormat</formatRule>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                        <param> ) </param>
                                    </paramsList>
                                </executableAction>
                                <executableAction type="Write">
                                    <paramsList>
                                        <param> ( </param>
                                        <formatRule type="Target">Command</formatRule>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                        <param> )</param>
                                    </paramsList>
                                </executableAction>
                            </rule>
                        </rulesOnMove>
                    </rulesForArgument> END OF ARGUMENT 1 (Index = 0)

                    <rulesForArgument index = „1"> ARGUMENT 2 (Index = 1)
                        <dynamicRules>
                            <rule>
                                <conditionalAction type="IsLastStat">
                                    <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                                    <paramsList>
                                        <param>List</param>
                                    </paramsList>
                                </conditionalAction>
                                <executableAction type="ChangeLastStat">
                                    <param>CodeBlock</param>
                                </executableAction>
```

```
                    </rule>
                    <rule>
                        <conditionalAction type="IsLastStat">
                            <paramsList>
                                <param>EndOfCodeBlock</param>
                                <param>Whitespace</param>
                                <param>CodeBlock</param>
                                <param>Comment</param>
                                <param>Ignore</param>
                                <param>EndOfString</param>
                                <param>SpecialSign</param>
                            </paramsList>
                        </conditionalAction>
                    </rule>
                    <rule>
                        <executableAction type="Error">
                            <paramsList>
                                <param>STR-7</param>
                            </paramsList>
                        </executableAction>
                    </rule>
                </dynamicRules>

                <rulesOnMove>
                    <rule>
                        <executableAction type="Write">
                            <paramsList>
                                <formatRule type="NameOrIndex">-1</formatRule>
                            </paramsList>
                        </executableAction>
                    </rule>
                </rulesOnMove>
            </rulesForArgument> END OF ARGUMENT 2 (Index = 1)

            <rulesForUnspecifiedArgument>
                <dynamicRules>
                    <rule>
                        <conditionalAction type="IsLastStat">
                            <paramsList>
                                <param>List</param>
                            </paramsList>
                        </conditionalAction>
                        <conditionalAction type="Compare">
                            <paramsList>
                                <param>else</param>
                            </paramsList>
                            <paramsList>
                                <formatRule type="Target">Command</formatRule>
                                <formatRule type="NameOrIndex">-1</formatRule>
                            </paramsList>
                        </conditionalAction>
                        <executableAction type="ChangeSavedStat">
                            <param>CodeBlock</param>
```

```
                    </executableAction>
                </rule>
                <rule>
                    <conditionalAction type="IsLastStat">
                        <paramsList>
                            <param>List</param>
                        </paramsList>
                    </conditionalAction>
                    <conditionalAction type="Compare">
                        <paramsList>
                            <param>elseif</param>
                        </paramsList>
                        <paramsList>
                            <formatRule type="Target">Command</formatRule>
                            <formatRule type="NameOrIndex">-2</formatRule>
                        </paramsList>
                    </conditionalAction>
                    <executableAction type="ChangeSavedStat">
                        <param>CodeBlock</param>
                    </executableAction>
                </rule>
            </dynamicRules>
            <rulesOnMove>
                <rule>
                    <conditionalAction type="Compare">
                        <paramsList>
                            <param>elseifelseif</param>
                        </paramsList>
                    </conditionalAction>
                    <executableAction type="Error">
                        <paramsList>
                            <param>STR-8</param>
                        </paramsList>
                    </executableAction>
                </rule>
                <rule>
                    <conditionalAction type="Compare">
                        <paramsList>
                            <param>elseif</param>
                        </paramsList>
                        <paramsList>
                            <formatRule type="Target">Command</formatRule>
                            <formatRule type="NameOrIndex">-2</formatRule>
                        </paramsList>
                    </conditionalAction>
                    <executableAction type="TclParse">
                        <paramsList>
                            <param>expr_parser </param>
                            <formatRule type="Target">Command</formatRule>
                            <formatRule type="NameOrIndex">-1</formatRule>
                        </paramsList>
                    </executableAction>
                    <executableAction type="Write">
```

```
                                <paramsList>
                                        <param>( </param>
                                        <formatRule type="Target">Command</formatRule>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                        <param> )</param>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <conditionalAction type="Compare">
                                <paramsList>
                                        <param>else</param>
                                </paramsList>
                                <paramsList>
                                        <formatRule type="Target">Command</formatRule>
                                        <formatRule type="NameOrIndex">-2</formatRule>
                                </paramsList>
                        </conditionalAction>
                        <executableAction type="Write">
                                <paramsList>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                </paramsList>
                        </executableAction>
                </rule>
                <rule>
                        <conditionalAction type="Compare">
                                <paramsList>
                                        <param>elseif</param>
                                </paramsList>
                                <paramsList>
                                        <formatRule type="Target">Command</formatRule>
                                        <formatRule type="NameOrIndex">-3</formatRule>
                                </paramsList>
                        </conditionalAction>
                        <executableAction type="Write">
                                <paramsList>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                </paramsList>
                        </executableAction>
                </rule>
        </rulesOnMove>
</rulesForUnspecifiedArgument>

<rulesOnEndOfCall>
        <rule>
                <executableAction type="Write">
                        <paramsList>
                                <formatRule type="NameOrIndex" />
                                <formatRule type="Target">Command</formatRule>
                                <formatRule type="ArgumentsFrom">0</formatRule>
                        </paramsList>
                </executableAction>
        </rule>
```

```
            <rule>
                  <conditionalAction type=”CompareNumbOfArguments”>

                        <param>1 </param>
                  </conditionalAction>
                  <executableAction type=”Write”>
                        <paramsList>
                              <formatRule type=”NameOrIndex”>0</formatRule>
                              <param>++</param>
                        </paramsList>
                  </executableAction>
            </rule>
            <rule>
                  <executableAction type=”AddUserInteraction” />
            </rule>
      </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis ++*

Przykład

Incr a 5 -> a += 5
Incr a -> a++

For

Zapis do pliku konfiguracyjnego

```
<procedure name=”for”>
      <rulesForArgument index = „0”> ARGUMENT 1 (Index = 0)
            <dynamicRules>
                  <rule>
                        <conditionalAction type=”IsLastStat”>
                              <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                              <paramsList>
                                    <param>List</param>
                              </paramsList>
```

```
                    </conditionalAction>
                    <executableAction type="ChangeSavedStat">
                        <param>Expression</param>
                    </executableAction>
                </rule>
                <rule>
                    <conditionalAction type="IsLastStat">
                        <paramsList>
                            <param>EndOfCodeBlock</param>
                            <param>EndOfExpression</param>
                            <param>Whitespace</param>
                            <param>CodeBlock</param>
                            <param>Comment</param>
                            <param>Ignore</param>
                            <param>FunctionCall</param>
                        </paramsList>
                    </conditionalAction>
                </rule>
                <rule>
                    <executableAction type="Error">
                        <paramsList>
                            <param>STR-9</param>
                        </paramsList>
                    </executableAction>
            </dynamicRules>

            <rulesOnMove>
                <rule>
                    <executableAction type="Write">
                        <paramsList>
                            <formatRule type="NameOrIndex">-1</formatRule>
                        </paramsList>
                    </executableAction>
                </rule>
            </rulesOnMove>
    </rulesForArgument> END OF ARGUMENT 1 (Index = 0)

    <rulesForArgument index = „1"> ARGUMENT 2 (Index = 1)
            <dynamicRules>
                <rule>
                    <conditionalAction type="IsLastStat">
                        <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                        <paramsList>
                            <param>List</param>
                            <param>EndOfExpression</param>
                            <param>EndOfList</param>
                            <param>FunctionCall</param>
                            <param>SpecialSign</param>
                            <param>Whitespace</param>
                            <param>Operator</param>
                            <param>PendingSnprintf</param>
                            <param>PendingString</param>
                            <param>EndOfString</param>
```

```
                                        <param>EndOfCodeBlock</param>
                                    </paramsList>
                                </conditionalAction>
                            </rule>
                            <rule>
                                <executableAction type="Error">
                                    <paramsList>
                                        <param>STR-10</param>
                                    </paramsList>
                                </executableAction>
                            </rule>
                        </dynamicRules>

                        <rulesOnMove>
                            <rule>
                                <executableAction type="TclParse">
                                    <paramsList>
                                        <param>expr_parser </param>
                                        <formatRule type="Target">Command</formatRule>
                                        <formatRule type="NameOrIndex">-1</formatRule>
                                    </paramsList>
                                </executableAction>
                            </rule>
                        </rulesOnMove>
                    </rulesForArgument> END OF ARGUMENT 2 (Index = 1)
                    <rulesForArgument index = „2"> ARGUMENT 3 (Index = 2)
                        <dynamicRules>
                            <rule>
                                <conditionalAction type="IsLastStat">
                                    <paramsList>
                                        <param>List</param>
                                    </paramsList>
                                </conditionalAction>
                                <executableAction type="ChangeSavedStat">
                                    <param>Expression</param>
                                </executableAction>
                            </rule>
                            <rule>
                                <conditionalAction type="IsLastStat">
                                    <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                                    <paramsList>
                                        <param>EndOfExpression</param>
                                        <param>EndOfList</param>
                                        <param>SpecialSign</param>
                                        <param>Whitespace</param>
                                        <param>EndOfString</param>
                                        <param>EndOfCodeBlock</param>
                                        <param>CodeBlock</param>
                                        <param>Comment</param>
                                        <param>Ignore</param>
                                    </paramsList>
                                </conditionalAction>
                            </rule>
                            <rule>
```

```
                    <executableAction type="Error">
                        <paramsList>
                            <param>STR-11</param>
                        </paramsList>
                    </executableAction>
                </rule>
            </dynamicRules>

            <rulesOnMove>
                <rule>
                    <executableAction type="Write">
                        <paramsList>
                            <formatRule type="NameOrIndex">-1</formatRule>
                        </paramsList>
                    </executableAction>
                </rule>
            </rulesOnMove>
        </rulesForArgument> END OF ARGUMENT 3 (Index = 2)

        <rulesForArgument index = „3"> ARGUMENT 4 (Index = 3)
            <dynamicRules>
                <rule>
                    <conditionalAction type="IsLastStat">
                        <paramsList>
                            <param>List</param>
                        </paramsList>
                    </conditionalAction>
                    <executableAction type="ChangeSavedStat">
                        <param>CodeBlock</param>
                    </executableAction>
                <rule>
                    <conditionalAction type="IsLastStat">
                        <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                        <paramsList>
                            <param>EndOfExpression</param>
                            <param>Whitespace</param>
                            <param>EndOfCodeBlock</param>
                            <param>CodeBlock</param>
                            <param>Comment</param>
                            <param>Ignore</param>
                        </paramsList>
                    </conditionalAction>
                </rule>
                <rule>
                    <executableAction type="Error">
                        <paramsList>
                            <param>STR-12</param>
                        </paramsList>
                    </executableAction>
                </rule>
            </dynamicRules>

            <rulesOnMove>
```

```
                    <rule>
                        <executableAction type="Write">
                            <paramsList>
                                <formatRule type="NameOrIndex">-1</formatRule>
                            </paramsList>
                        </executableAction>
                    </rule>
                </rulesOnMove>
            </rulesForArgument> END OF ARGUMENT 4 (Index = 3)

            <rulesOnEndOfCall>
                <rule>
                    <conditionalAction type="CompareNumbOfArguments">
                        <param>4</param>
                    </conditionalAction>
                    <executableAction type="Write">
                        <paramsList>
                            <formatRule type="NameOrIndex" />
                            <param>(</param>
                            <formatRule type="NameOrIndex">0</formatRule>
                            <param>; </param>
                            <formatRule type="Target">Command</formatRule>
                            <formatRule type="NameOrIndex">1</formatRule>
                            <param>; </param>
                            <formatRule type="Target">CaplFormat</formatRule>
                            <formatRule type="NameOrIndex">2</formatRule>
                            <param>)</param>
                            <formatRule type="NameOrIndex">3</formatRule>
                        </paramsList>
                    </executableAction>
                </rule>
            </rulesOnEndOfCall>
</procedure>
```

Opis

- Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>
- Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:
  - Zapisanie pierwszego argumentu (index = 0) w formacie CAPL
  - Napis +=
  - Zapisanie drugiego argumentu (index = 1) w formacie CAPL
- Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.
- Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:
  - Zapisanie pierwszego argumentu (index = 0) w formacie CAPL
  - Napis ++

Przykład

Incr a 5 -> a += 5
Incr a -> a++

Foreach

Zapis do pliku konfiguracyjnego

```
<procedure name="for">
        <rulesForArgument index = „0"> ARGUMENT 1 (Index = 0)
                <rulesOnMove>
                        <rule>
                                <conditionalAction type="Compare">
                                        <paramsList>
                                                <param>String</param>
                                                <param>PendingString</param>
                                                <param>EndOfList</param>
                                        </paramsList>
                                        <paramsList>
                                                <formatRule type="Target">
                                                        ParametersStat
                                                </formatRule>
                                                <formatRule type="NameOrIndex">-1</formatRule>
                                        </paramsList>
                                </conditionalActio>
                        </rule>
                        <rule>
                                <executableAction type="Error">
                                        <paramsList>
                                                <param>STR-12</param>
                                        </paramsList>
                                </executableAction>
                </rulesOnMove>
        </rulesForArgument> END OF ARGUMENT 1 (Index = 0)

        <rulesForArgument index = „1"> ARGUMENT 2 (Index = 1)
                <rulesOnMove>
                        <rule>
                                <conditionalAction type="Compare">
                                        <paramsList>
                                                <param>StringInQuotes </param>
                                                <param>EndOfList</param>
                                                <param>SpeechMark</param>
                                                <param>PendingSnprintf</param>
                                                <param>Variable</param>
                                                <param>Snprintf</param>
                                                <param>FunctionCall</param>
                                        </paramsList>
                                        <paramsList>
                                                <formatRule type="Target">
                                                        ParametersStat
                                                </formatRule>
                                                <formatRule type="NameOrIndex">-1</formatRule>
                                        </paramsList>
                                </conditionalAction>
```

```
                    </rule>
                    <rule>
                        <executableAction type=”Error”>
                            <paramsList>
                                <param>STR-13</param>
                            </paramsList>
                        </executableAction>
                    </rule>
                </rulesOnMove>
            </rulesForArgument> END OF ARGUMENT 2 (Index = 1)
            <rulesForArgument index = „2”> ARGUMENT 3 (Index = 2)
                <dynamicRules>
                    <rule>
                        <conditionalAction type=”IsLastStat”>
                            <paramsList>
                                <param>List</param>
                            </paramsList>
                        </conditionalAction>
                        <executableAction type=”ChangeSavedStat”>
                            <param>CodeBlock</param>
                        </executableAction>
                    <rule>
                        <conditionalAction type=”IsLastStat”>
                            <- Zapis w formie napisów (paramsList) dla przejrzystości ->
                            <paramsList>
                                <param>EndOfExpression</param>
                                <param>Whitespace</param>
                                <param>EndOfCodeBlock</param>
                                <param>CodeBlock</param>
                                <param>Comment</param>
                                <param>Ignore</param>
                            </paramsList>
                        </conditionalAction>
                    </rule>
                    <rule>
                        <executableAction type=”Error”>
                            <paramsList>
                                <param>STR-14</param>
                            </paramsList>
                        </executableAction>
                    </rule>
                </dynamicRules>

                <rulesOnMove>
                    <rule>
                        <executableAction type=”Write”>
                            <paramsList>
                                <formatRule type=”NameOrIndex”>-1</formatRule>
                            </paramsList>
                        </executableAction>
                    </rule>
                </rulesOnMove>
            </rulesForArgument> END OF ARGUMENT 3 (Index = 2)
```

```
        <rulesOnEndOfCall>
                <rule>
                        <executableAction type="FinalizeForEach" /> PRIVATE
                </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis ++*

Przykład

Incr a 5 -> a += 5
Incr a -> a++

Procedura niezdefiniowana w regułach

Reguły, które dotyczą każdej procedury, dla których nie ma ani domyślnych reguł, ani nie zostały one zdefiniowane przez użytkownika w pliku konfiguracyjnym.

Zapis do pliku konfiguracyjnego

```
<procedure name="for">
        <rulesOnEndOfCall>
                <rule>
                        <executableAction type="Write">
                                <paramsList>
                                        <formatRule type="NameOrIndex" />
                                        <param>(</param>
                                        <formatRule type="Separator">, </formatRule>
                                        <formatRule type="ArgumentsFrom">0</formatRule>
                                        <param>)</param>
                                </paramsList>
                        </executableAction>
```

```
                <executableAction type="AddFunctionDefinition" />
            </rule>
        </rulesOnEndOfCall>
</procedure>
```

Opis

- *Wszystkie reguły wykonywane są na zakończenie procedury <rulesOnEndOfCall>*
- *Pierwsza reguła sprawdza, czy procedura ma 2 argumenty. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis +=*
  - *Zapisanie drugiego argumentu (index = 1) w formacie CAPL*
- *Druga reguła jest sprawdzana, jeśli pierwsza nie została spełniona.*
- *Druga reguła sprawdza, czy procedura ma 1 argument. Jeśli ma, interpreter zapisuje sformatowany napis:*
  - *Zapisanie pierwszego argumentu (index = 0) w formacie CAPL*
  - *Napis ++*

Przykład

Incr a 5 -> a += 5
Incr a -> a++