# Capitolo 3
# Espressioni aritmetiche

Le espressioni aritmetiche in UniSim sono state implementate per dare la possibilità all'utente di poter usare in pieno le variabili intere.

## 3.1 Specifica

Le espressioni aritmetiche devono poter essere utilizzate in fase di definizione dell'azione associata ad una fase. La variabile di riferimento dell'azione (reference) deve contenere il risultato dell'espressione mentre, per la definizione dell'espressione stessa, devono poter essere utilizzate un numero arbitrario di variabili intere e costanti numeriche.

Gli operatori che devono essere messi a disposizione sono illustrati nella seguente tabella:

| **Simbolo** | **Significato** |
|---|---|
| + | Addizione |
| - | Sottrazione |
| * | Moltiplicazione |
| / | Modulo |

Deve essere previsto, inoltre, l'uso di parentesi tonde per la determinazione delle precedenze di calcolo.

Per il calcolo del risultato dell'espressione devono essere rispettate le seguenti regole:

- Gli operatori di moltiplicazione e modulo hanno la precedenza sugli operatori di addizione e sottrazione
- Se l'espressione contiene solo operatori con la stessa precedenza, essa si esegue nell'ordine scritto (da sinistra a destra)
- Se l'espressione contiene parentesi tonde si eseguono prima le operazioni all'interno delle parentesi fino ad eliminarle

Ad esempio, seguendo le regole sopra indicate l'espressione:

((5/2+7)*2)-4

ha come risultato 14.

## 3.2 Progettazione

Le modifiche da apportare ad UniSim per l'implementazione delle espressioni aritmetiche riguardano tre ambiti del modello software del tool: interfaccia grafica, motore di simulazione, import/export dei progetti.

### 3.2.1 Progettazione delle modifiche per l'interfaccia grafica

Le modifiche da apportare per l'interfaccia grafica interessano la finestra di dialogo per l'aggiunta di una nuova azione. Come visto nel capitolo 2 riguardo ai contatori, quando l'utente seleziona come variabile di riferimento una variabile intera, il programma permette di scegliere le possibili azioni che si possono attuare su di essa: incremento, decremento e azzeramento. L'opzione di definire un'espressione aritmetica quindi può essere aggiunta alle operazioni già messe a disposizione per una variabile intera, fornendo all'utente una casella di input per la scrittura dell'espressione.

Prima che l'utente possa aggiungere la nuova azione, deve essere controllata la correttezza sintattica dell'espressione fornita e l'esistenza delle variabili utilizzate.

Le classi da modificare sono quindi "ActionDialogForm" per l'aggiunta dei nuovi elementi grafici e "GraphicalAction" (metodo "Draw()") per la corretta visualizzazione nell'editor SFC dell'azione in caso essa utilizzi un'espressione aritmetica.

### 3.2.2 Progettazione delle modifiche per la simulazione

Bisogna prima analizzare le classi che entrano in gioco in fase di simulazione e come esse interagiscono. In figura dodici è riportato il diagramma delle classi di UnisimClassLibrary di interesse per le modifiche da realizzare :

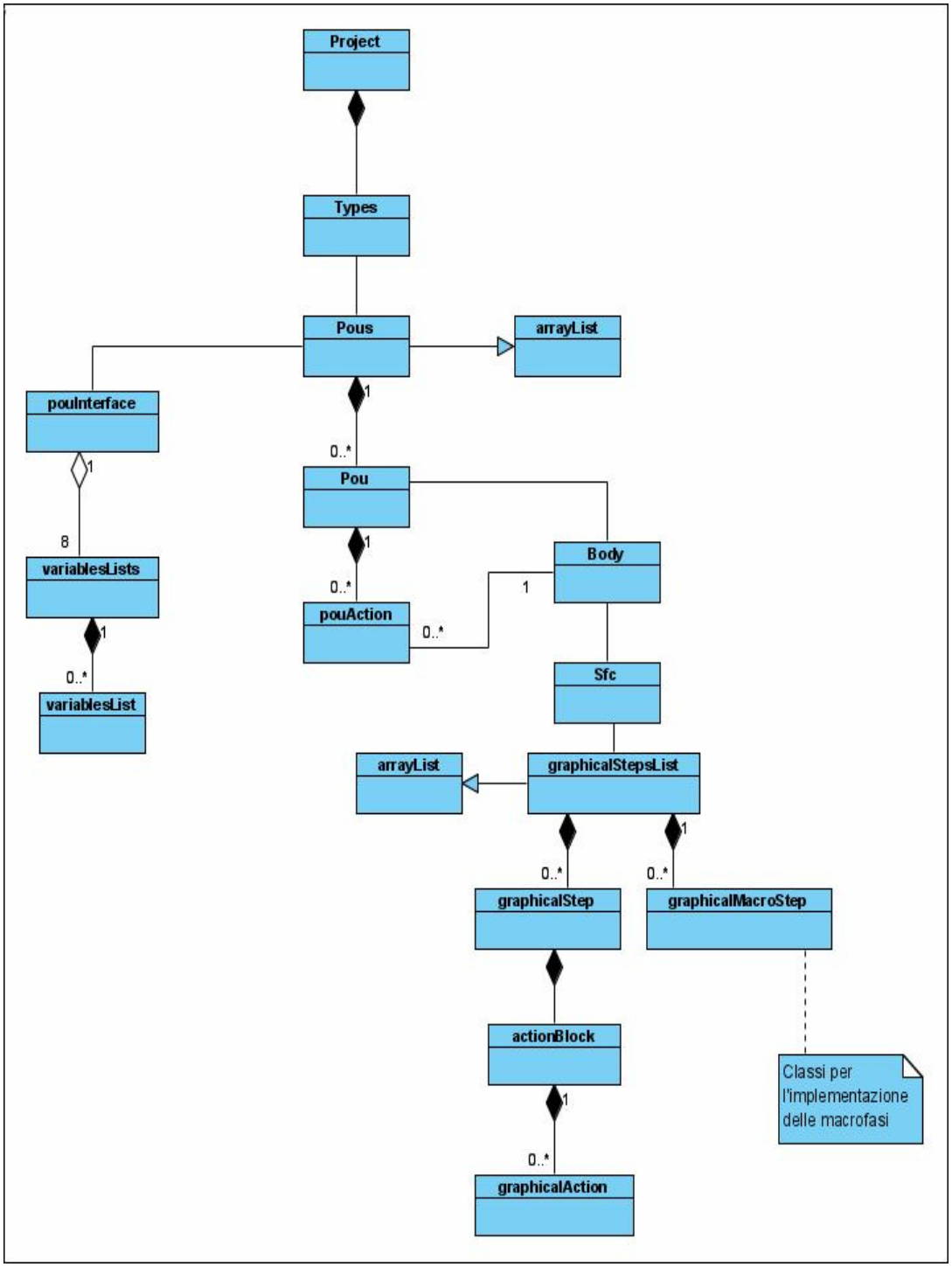


Figura 12: Diagramma delle classi

La classe Pous, che contiene gli oggetti di tipo Pou, eredita la classe ArrayList, che implementa una lista dinamica e fornisce i metodi per l'aggiunta, la rimozione e la ricerca in lista delle istanze della classe Pou. La classe Pou contiene, oltre agli attributi che definiscono il nome ed il tipo, anche un'istanza della classe pouInterface, che implementa l'interfaccia e contiene le liste delle variabili definite dall'utente. La classe Body contine i programmi scritti nei diversi linguaggi di programmazione descritti nello standard IEC 61131-3.

La classe Sfc dispone degli attributi e dei metodi per l'editing dei programmi, per l'esecuzione degli stessi in fase di simulazione e controllo e per il relativo monitoraggio dell'evoluzione dello stato.

La classe GraphicalStep consta di una serie di metodi per la gestione delle operazioni a carattere grafico: disegno, spostamento, cancellazione, calcolo dell'area occupata, ecc. La classe ActionBlock rappresenta una lista dinamica in cui sono contenute tutte le azioni di una determinata fase. La classe GraphicalAction rappresenta la singola azione che deve essere eseguita ed oltre agli attributi ed ai metodi finalizzati al disegno, contiene quelli indirizzati all'esecuzione delle azioni. L'attributo m_quelifier esprime il qualificatore dell'azione e può assumere uno dei seguenti valori definiti dallo standard: "N", "S", "R", "L", "D", "P", "SD", "DS", "SL" più i seguenti: "fr", "ifr", "sus" e "st", per definire quale delle macroazioni implementate si vuole scegliere.

In figura tredici il diagramma di sequenza mostra lo scenario e i messaggi che si scambiano i vari oggetti quando vengono eseguite le azioni legate ad una fase:

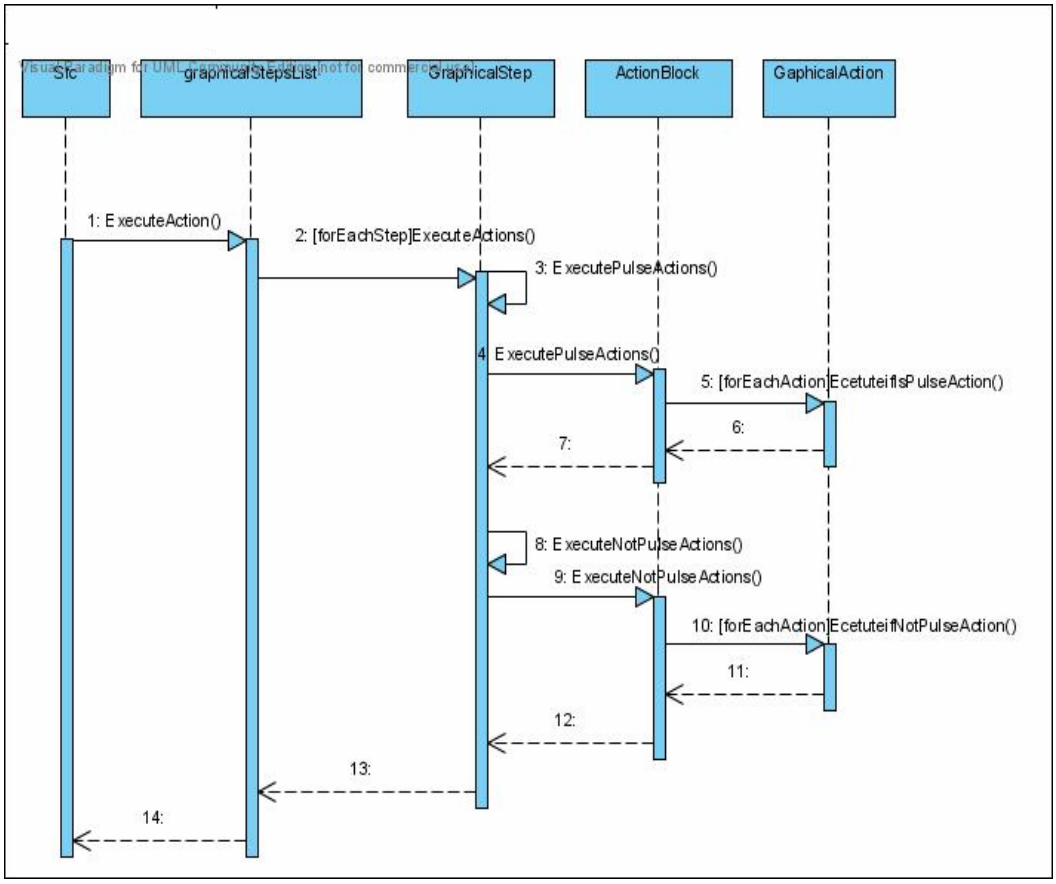


Figura 13: Diagramma di sequenza

L'ultimo motodo invocato è "ExecuteifIsPulseAction()" oppure "ExecuteifNotPulseAction()" a seconda che l'azione sia di tipo impulsivo o no. Poiché le azioni sulle variabili intere devono essere considerate da UniSim come impulsive (tipo "P"), il metodo da modificare per l'implementazione delle espressioni aritmetiche è "ExecuteifIsPulseAction()". In particolare bisogna eseguire il calcolo dell'espressione e assegnare il risultato alla variabile di riferimento, nel caso quest'ultima sia intera e l'espressione sia stata definita.

Bisogna, quindi, aggiungere un attributo del tipo String (m_ArithExp) che tenga traccia dell'espressione aritmetica definita dall'utente al momento dell'aggiunta dell'azione.

### 3.2.3 *La classe ArithmeticExpression*

Dopo aver individuato le classi da modificare per l'interfaccia grafica e per la simulazione bisogna creare una nuova classe che metta a disposizione dei metodi per il calcolo dell'espressione aritmetica e per la verifica della sintassi.

Questa classe sarà utilizzata sia all'interno della classe ActionDialogForm per la verifica della sintassi, sia all'interno della classe GraphicalAction per il calcolo dell'espressione.

La classe ArithmeticExpression fornisce quindi la funzione Parse() che ritorna "true" se la sintassi è corretta, "false" altrimenti, e la funzione calculateExp() che ritorna il risultato dell'espressione generando una eccezione in caso di errori (variabili non trovate, divisioni per zero, ecc…).

Nel capitolo quattro si vedrà l'implementazione di questa classe e l'algoritmo utilizzato per la risoluzione delle espressioni.

### 3.2.4 *Progettazione delle modifiche per l'import/export del progetto*

L'importazione e l'esportazione dei progetti sono gestiti dalla classe XMLProjectManager gestisce. Essa dispone di un XML Validating Parser che effettua la validazione all'atto dell'importazione dei progetti. Attraverso una serie di chiamate ai metodi xmlImport degli oggetti corrispondenti agli elementi presenti nel file xml, viene creato il progetto. Precisamente, ogni oggetto crea gli oggetti di livello gerarchico inferiore e successivamente ne chiama il relativo metodo xmlImport. Ogni metodo xmlImport, leggendo il contenuto degli elementi, fa avanzare la posizione di lettura del suddetto oggetto. L'importazione prosegue sino al termine degli elementi contenuti nel file. L'esportazione del progetto avviene seguendo lo stesso metodo. Ognuno dei metodi xmlExport appartenenti allo specifico oggetto scrive le relative informazioni in un buffer dell'oggetto stesso. Quando è terminata l'intera scrittura viene creato il file xml contente il progetto.

In base al funzionamento descritto si capisce che gli unici metodi da modificare sono xmlImport e xmlExport della classe GraphicalAction. Inoltre si deve modificare lo XML Schema per permettere il salvataggio dell' espressione aritmetica.

## 3.3 Uso delle espressioni aritmetiche in UniSim

Per utilizzare le espressioni aritmetiche in UniSim bisogna aggiungere una nuova azione che agisce su di una variabile intera.

Supponendo di aver definito le variabili “a”, “b” e “c”, nella finestra di dialogo per l’aggiunta di una nuova azione si sceglie come variabile “Reference”, ad esempio, la variabile intera “a” ottenendo la seguente schermata:

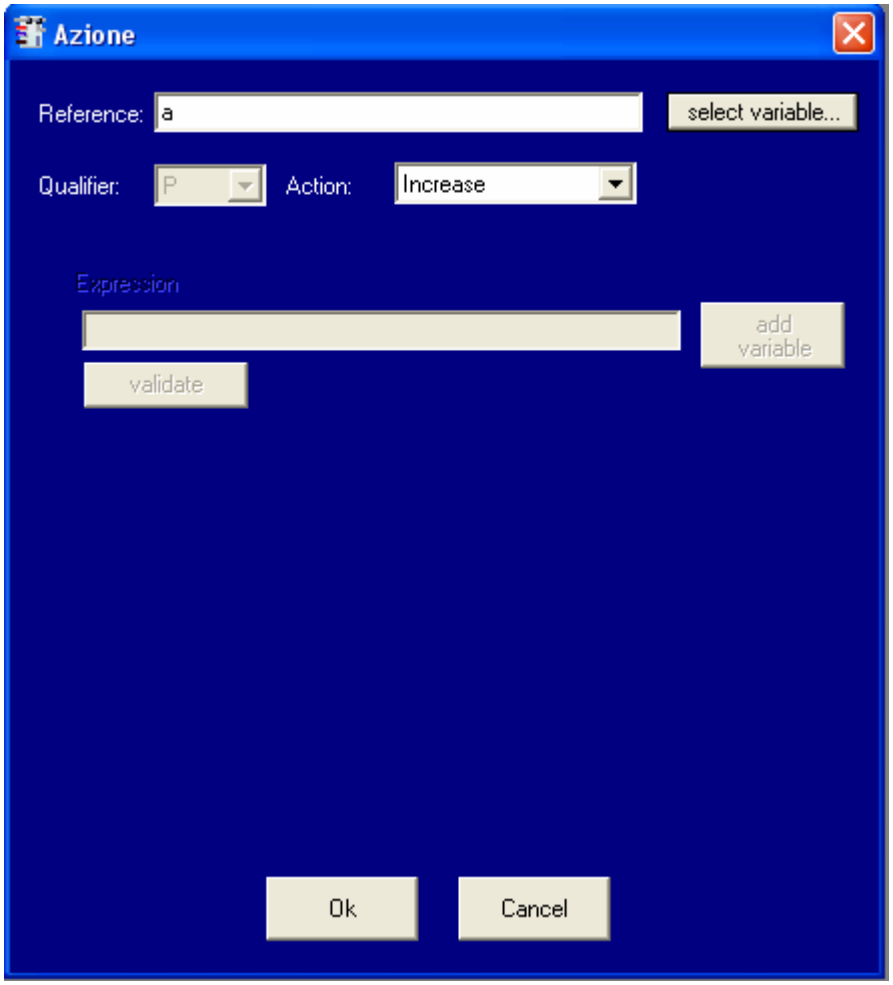


Figura 14: Schermata per la definizione di una azione su una variabile intera

Scegliendo dal combobox "Action" il tipo di azione "Operation" si abilita il textbox "Expression" nel quale è possibile scrivere l'espressione aritmetica. In figura quindici è riportato un esempio:

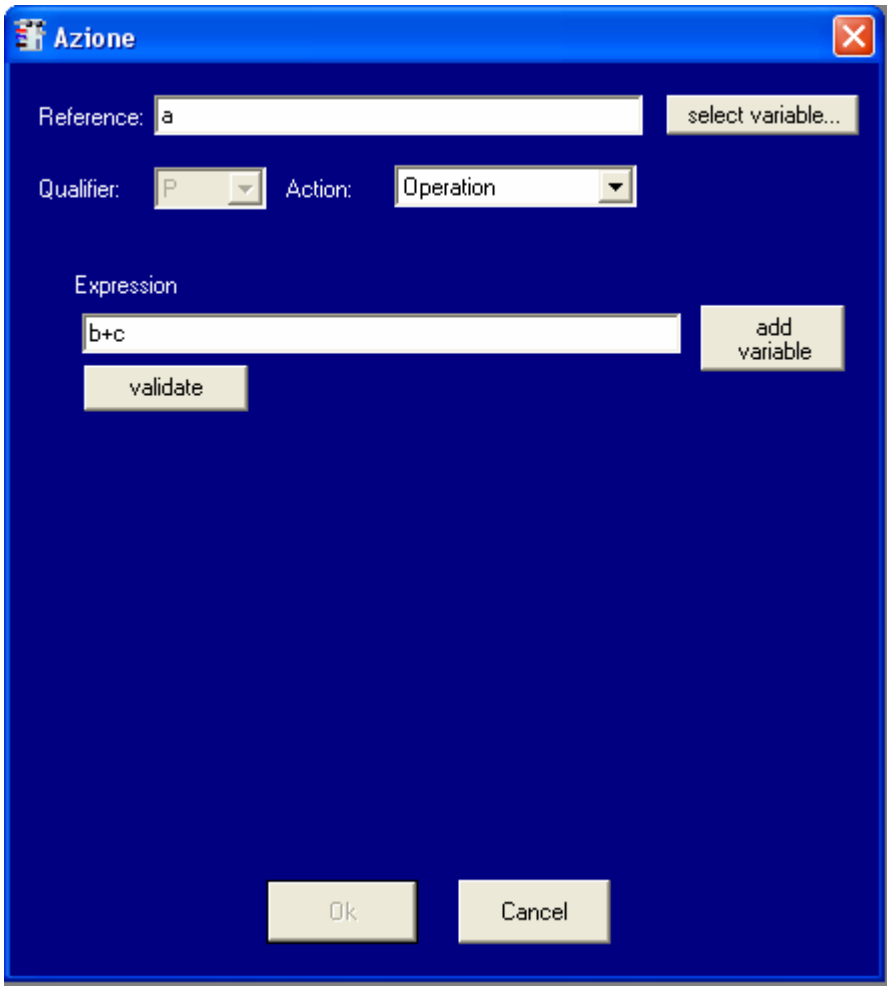


Figura 15: Esempio di definizione di una espressione aritmetica

Facendo click sul bottone "validate" il programma verificherà la correttezza sintattica dell'espressione e abiliterà il bottone "Ok". Facendo click su "Ok" si aggiungerà l'azione.

In figura sedici è riportato un semplice programma di esempio che, supponendo di aver le variabili intere “a” uguale a zero e “b” uguale a uno, incrementa “a” fino al raggiungimento del valore cinque.

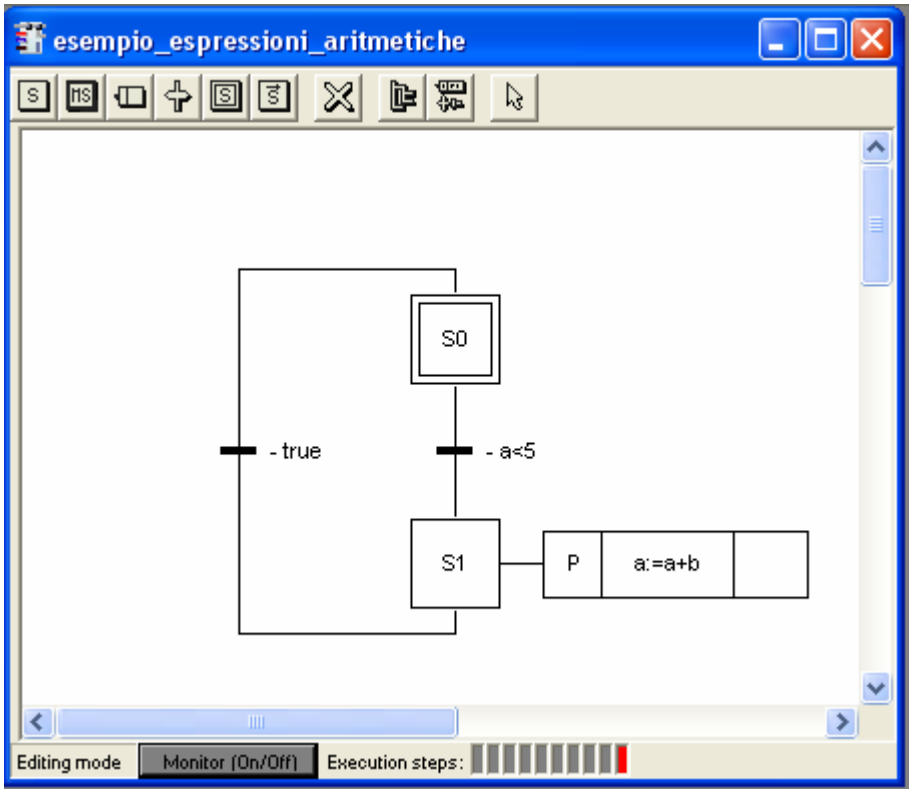


Figura 16: Esempio di utilizzo delle espressioni aritmetiche