Definizione Formato File

Giorgio Mariani

Sommario

In questo documento vengono specificate informazioni relative al file contenente le formule *BLTL*.

Codifica file

Il file deve utilizzare la codifica **ISO/IEC 8859-1**, questa è un estensione della codifica **ASCII** pensata per codificare i caratteri utilizzati in Europa.

Formato file

Il file deve essere composto da una sequenza di formule *BLTL* (La cui grammatica è definita più avanti) separate dalla stringa "**::**". Deve essere presente almeno una formula nel file.

1 Grammatica utilizzata

Consideriamo una formula in *Bounded Linear Temporal Logic* (*BLTL*) ϕ, definiamo in base alla struttura di ϕ come questa debba essere rappresentata in forma di stringa nel file dato in input:

- $\phi = True$ oppure $\phi = False$: sono indicati rispettivamente attraverso le stringhe **TRUE** e **FALSE**.
- $\phi = \rho$, con:

  $$\rho = \sum_{i=0}^{n} \alpha_i x_i \ \mathcal{R} \ \beta$$

  Viene indicato con la seguente sintassi:

  $$\alpha_1\text{*}x_1 \text{ + } \alpha_2\text{*}x_2 \text{ ... } \alpha_n\text{*}x_n \ \mathcal{R} \ \beta$$

  Espressioni del tipo α_i e x_i rappresentano rispettivamente un valore reale non negativo ed un nome di variabile.

  I nomi di variabile devono rispettare la seguente *regular expression*[1]:

  [a-zA-Z_][a-zA-Z0-9_]*

  Espressioni del tipo α_i*x_i possono anche essere scritte come: $\alpha_i x_i$, x_i*α_i, $x_i \alpha_i$ oppure soltanto x_i se il valore di α_i è 1.

  Con ... abbiamo indicato una sequenza arbitrariamente lunga di espressioni del tipo α_i*x_i precedute dal carattere **+** oppure **-** (nel primo prodotto può essere omesso).

  La espressione $\mathcal{R}$ indica una tra seguenti stringhe: <, <=, >, >=, ∼=, =. La espressione β è un valore reale.

[1] sintassi del programma **egrep**.

- $\phi = \neg f_1$:
è indicata dalla parola chiave **NOT** ed è espressa come segue:

  NOT *formula*

- $\phi = \Diamond^{\alpha} f_1$:
è indicata dalla parola chiave **FUTURE** ed è espressa come segue:

  FUTURE[α] *formula*

  Con α valore reale positivo.

- $\phi = \Box^{\alpha} f_1$:
è indicata dalla parola chiave **GLOBALLY** ed è espressa come segue:

  GLOBALLY[α] *formula*

  Con α valore reale positivo.

I costrutti **NOT**, **GLOBALLY** e **FUTURE** hanno la precedenza più alta, quindi nel caso si voglia utilizzarli per espressioni non banali occorre racchiudere il corpo dentro a delle parentesi tonde.

ESEMPIO

FUTURE[1](*formula* **AND** *formula*)

- $\phi = f_1 \wedge f_2$: è indicata dalla parola chiave **AND** ed è espressa come segue:

  formula **AND** *formula*

  L'operatore **AND** è il costrutto con la seconda precedenza più alta. La associatività dell'operatore è da sinistra verso destra.

- $\phi = f_1 \vee f_2$: è indicata dalla parola chiave **OR** ed è espressa come segue:

  formula **OR** *formula*

  L'operatore **OR** è il costrutto con la terza precedenza più alta. La associatività dell'operatore è da sinistra verso destra.

- $\phi = f_1 \mathbf{U}^{\alpha} f_2$:
Un espressione dei questo tipo è indicata dalla parola chiave **UNTIL** ed è espressa come segue:

  formula **UNTIL**[α] *formula*

  Con α valore reale positivo. L'operatore **UNTIL** è il costrutto con la precedenza minore. La associatività dell'operatore è da sinistra verso destra.

2 Dettagli grammatica

2.1 Token utilizzati

I token non banali utilizzati per generare il linguaggio sono:

<NUMBER> token utilizzato per rappresentare un valore reale non negativo. Le stringhe identificate da questo token sono quelle che rispettano la seguente *regular expression*:

[0-9]+(\.[0-9]+)?(e(\+ | \-)?[0-9]+)?

<VARIABLE> token utilizzato per rappresentare un nome di variabile. Le stringhe identificate da questo token sono quelle che rispettano la seguente *regular expression*:

[a-zA-Z_][a-zA-Z0-9_]*

<SIGN> token utilizzato per rappresentare un il segno + oppure il segno -.

<RELATION> token utilizzato per rappresentare i simboli di relazione: <, <=, >, >=, ∼=, =.

Gli spazi, tab, e i caratteri di nuova linea vengono tutti ignorati.

2.2 Regole di produzione

Regole *BNF* utilizzate per generare la grammatica sopra descritta:

⟨*until_rule*⟩ ::= ⟨*until_rule*⟩ '`UNTIL[`' ⟨*NUMBER*⟩ '`]`' ⟨*or_rule*⟩
| ⟨*or_rule*⟩

⟨*or_rule*⟩ ::= ⟨*or_rule*⟩ '`OR`' ⟨*and_rule*⟩
| ⟨*and_rule*⟩

⟨*and_rule*⟩ ::= ⟨*and_rule*⟩ '`AND`' ⟨*unary_rule*⟩
| ⟨*unary_rule*⟩

⟨*unary_rule*⟩ ::= '`NOT`' ⟨*unary_rule*⟩
| '`GLOBALLY[`' ⟨*NUMBER*⟩ '`]`' ⟨*unary_rule*⟩
| '`FUTURE[`' ⟨*NUMBER*⟩ '`]`' ⟨*unary_rule*⟩
| ⟨*pred_rule*⟩

⟨*pred_rule*⟩ ::= ⟨*sign_rule*⟩ ⟨*sum_rule*⟩ ⟨*RELATION*⟩ ⟨*sign_rule*⟩ ⟨*NUMBER*⟩
| '`TRUE`'
| '`FALSE`'
| '`(`' ⟨*until_rule*⟩ '`)`'

⟨*sum_rule*⟩ ::= ⟨*prod_rule*⟩ ⟨*SIGN*⟩ ⟨*sum_rule*⟩
| ⟨*prod_rule*⟩

⟨*prod_rule*⟩ ::= ⟨*NUMBER*⟩ '`*`' ⟨*VARIABLE*⟩
| ⟨*VARIABLE*⟩ '`*`' ⟨*NUMBER*⟩
| ⟨*NUMBER*⟩ ⟨*VARIABLE*⟩
| ⟨*VARIABLE*⟩ ⟨*NUMBER*⟩
| ⟨*VARIABLE*⟩

⟨*sing_rule*⟩ ::= ⟨*SIGN*⟩
| ⟨*empty string*⟩