PLEB

2022-03-24

NAME

pleb - Piecewise / Local Expression Builder

DÉCRITION

Un fichier Pleb définit un langage formel (un ensemble de chaînes de caractères) par une expression logique. Il peut aussi définir des variables pour dénoter des ensembles de caractères ou des autres expressions. Les expressions logiques peuvent exprimer tous les langages rationnels, mais le système se concentre principalement sur les contraintes dans lesquelles l'ensemble des facteurs est suffisant pour vérifier qu'un mot est bien formé.

Les syntaxes de base

Les caractères d'espacement, <*ws*>, sont ignorés partout, sauf dans les symboles ou sauf indication contraires. Les commentaires sont également ignorés.

<*commentaire*> ::= # [<*non-CR*> ...] <*CR*>

Un fichier (un programme) est une chaîne non vide d'instructions. Après avoir exécuté un programme avec **LTK.Porters.Pleb.readPleb**, le résultat se trouve dans la variable **it**. C'est généralement la dernière expression nue (le cas échéant). Si **it** n'a pas de valeur, ce moyen d'exécution générerait une erreur. L'automate de résultat (le cas échéant) utilise les caractères dans de l'ensemble **universe**, la variable qui accumule tous les caractères qui apparaissent dans le fichier.

<*programme*> ::= <*instruction*> [<*instruction*> ...]

Une instruction est soit une expression, soit une affectation soit d'un ensemble de caractères soit d'une expression.

<*instruction*> ::= <*affectation-car*> | <*affectation-exp*> | <*exp*>

<*affectation-car*> ::= = <*nom*> <*caractères*>

<*affectation-exp*> ::= = <*nom*> <*expression*>

Une expression est l'un des trois sortes. Ce peut être une expression unaire, une expression n-aire, ou un facteur. Ce peut aussi être le nom d'une sous-expression affectée.

<*exp*> ::= <*nom*> | <*n-aire*> | <*unaire*> | <*facteur*>

<*n-aire*> ::= <*op-n*> **{** <*exp*> [**,** <*exp*> ...] **}**

| <*op-n*> **(** <*exp*> [**,** <*exp*> ...] **)**

<*unaire*> ::= <*op-u*> <*exp*>

Les facteurs sont un peu plus compliqués. La forme de base d'un facteur est une chaîne d'ensembles de caractères séparés par <*op-r*>, où <*op-r*> indique à la fois l'ordre et la proximité (par <*ws*>) ou l'ordre uniquement (par **,**). En plus, un facteur peut être libre (sans ancre), ou il peut être ancré à la tête ou à la queue. Finalement, il peut être le nom d'une variable qui représent un autre facteur.

<*facteur*> ::= <*nom*>

| [<*ancres*>] **<** [<*caractères*> [<*op-r*> <*caractères*> ...]] **>**

| **.<** <*facteur*> [**,** <*facteur*> ...] **>**

| **..<** <*facteur*> [**,** <*facteur*> ...] **>**

Le premier type de facteur composé combine son sous-facteurs avec proximité, et le second les combine avec la précedence. Les ancres sont notées comme suit.

<*ancres*> ::= <*tête-queue*> | <*tête*> | <*queue*>

<*tête-queue*> ::= %||%

<*tête*> ::= %|

<*queue*> ::= |%

Noter que <*tête-queue*> n'est qu'un seul symbole, donc les caractères d'espacement ne peuvent pas intervenez entre %| et |%.

Comme discuté précédemment, les opérateurs de relation <*op-r*> dans un <*facteur*> peuvent être soit un caractère d'espacement pour représenter à la fois l'ordre et la proximité, soit une virgule pour représenter seulement l'ordre.

<*op-r*> ::= <*ws*> | **,**

Les ensembles de caractères sont définis comme suit.

<*caractères*> ::= **{** <*caractères* [**,** <*caractères*> ...] **}**

| **(** <*caractères* [**,** <*caractères*> ...] **)**

| **[** <*caractères* [**,** <*caractères*> ...] **]**

| **/** <*nom*>

| <*nom*>

Les premier et deuxième formes utilisent des accolades ou des parenthèses pour dénoter l'union des ensembles qu'elles contiennent. La troisième utilise des crochets pour dénoter l'intersection. La quatrième est un singleton, contenant le caractère dont le nom suit le barre oblique. Finalement, un nom nu est une variable, qui doit être affectée à un ensemble.

Un nom est une lettre (comme défini par Unicode) suivie d'une chaîne de caractères dans laquelle aucun n'est un espace ni dans l'ensemble suivant:

, **[** **]** **(** **)** **{** **}** **<** **>**

Noter que le caractère # est valide, et qu'un commentaire ne peut donc pas apparaître dans un nom.

Les opérateurs n-aire

Un opérateur n-aire peut être l'un des suivants. Les formes écrites ici sont ASCII, mais il existe des formes Unicode qui sont résumées à la fin.

/ L'intersection d'opérandes (conjonction logique).

\/ L'union d'opérandes (disjonction logique).

@@ La concaténation avec des trous. Les opérandes apparaissent dans l'ordre mais la proximité n'est pas requise.

@ La concaténation d'opérandes. Notez qu'un facteur sans ancre permet une expansion arbitraire, donc cette opération peut agir de manière inattendue. On peut utiliser la forme **.<...>** dans ce cas.

**** Quotients à gauche d'opérandes, qui s'associent à gauche. Le quotient à gauche A\B est l'ensemble de chaînes qui peuvent faire un mot en B en suffixant à quelquechose en A. Celui-ci est une généralization du dérivée de Brzozowski.

// Quotients à droit d'opérandes, qui s'associent à droite. Le quotient à droit B/A est l'ensemble de chaînes qui peuve faire un mot en B en prefixant à quelquechose en A.

Les opérateurs unaire

Cette section décrit les opérations unaires telles que la précédente décrit les n-aires.

~ | ! La négation de l'opérande. C'est l'ensemble complémentaire.

***** La fermature itérative de l'opérande. C'est l'ensemble contenant tous concaténations finis de l'opérande avec lui-même. La même mise en garde qui affecte la concaténation s'applique ici: les facteurs sans ancre permettent une expansion arbitraire.

$ La fermeture vers le bas de l'opérande: toutes les chaînes qui peuvent être créees en supprimant zéro ou plusieurs caractères d'un mot valide.

[<*caractères*> [**,** <*caractères*> ...]]

Les caractères donnés specifient l'ensemble de caractères saillants, et tous les autres sont ignorés lors de la vérification de la bonne formation.

Les syntaxes Unicode

En plus de la syntaxe ASCII, il y a la syntaxe Unicode. Les synonymes suivent.

= <U+225D> [equal to by definition]

<...> <U+27E8>...<U+27E9> [mathematical left/right angle bracket]

%| <U+22CA> [right normal factor semidirect product]

|% <U+22C9> [left normal factor semidirect product]

/ <U+22C0> [n-ary logical and] ou <U+2227> [logical and] ou <U+22C2> [n-ary intersection] ou <U+2229> [intersection]

\/ <U+22C1> [n-ary logical or] ou <U+2228> [logical or] ou <U+22C3> [n-ary union] ou <U+222A> [union]

@ <U+2219> [bullet operator]

! <U+00AC> [not sign]

***** <U+2217> [asterisk operator]

$ <U+2193> [downwards arrow]

On peut utiliser ces synonymes sans configuration particulière, sauf possiblement à configurer l'environnement pour faciliter la saisie.

REMARQUES

L'arbre d'expression admet des automates, mais on ne peut pas les faire. L'interpeteur **plebby** construit ces expressions lors de l'import d'un automate à partir d'un fichier ou lors de la compilation d'expressions.

EXEMPLES

**** Le caractère "a" apparaît.

[/a]!%||%<> La même contraint, écrit en specifiant un caractère saillant: ne voyant que les **a**, le chaîne n'est pas vide.

= primaire {/H'}

= non-primaire {/L, /H}

= obligatoire <primaire>

= culminativité !<primaire, primaire>

/\{obligatoire, culminativité}

(1) Affecter l'ensemble {H'} au nom **primaire**

(2) Affecter l'ensemble {L, H} au nom **non-primaire**, alors chacun de L, H, et H' sont éléments de **universe.**

(3) Définir **obligatoire** comme la contrainte qu'un élément de **primaire** apparaisse.

(4) Définir **culminativité** comme la contraint qu'aucun élément de **primaire** n'apparaisse deux fois.

(5) Définir la variable éspecial **it** (et donc le résultat du programme) comme l'intersection de **culminativité** et **obligatoire**: l'ensemble de chaînes dans lequelle chacune contient exactement une occurrence d'un seul élément de **primaire**.

VOIR AUSSI

plebby(1)