Linguistica Computazionale - Scienze della Comunicazione **Laboratorio 3**
Cristiano Chesi (chesi@media.unisi.it) 31 Marzo 2004

PARSING CON PHP

Costruzione di grammatiche

(1) Collegarsi all'indirizzo **http://www.ciscl.unisi.it/lab** (più un codice numerico comunicato durante l'esercitazione)

(2) aprire la finestra per l'editing delle grammatiche (**edit grammar**)

(3) inserire le seguenti regole rispettivamente sotto le colonne "rules" e "lexicon"

Rules	Lexicon
S > DP VP	D > il
DP > D N	D > lo
VP > V DP	D > la
	D > i
	D > gli
	D > le
	D > uno
	D > un
	D > una
	D > l
	N > mela
	N > topo
	N > gatto
	N > uomo
	V > vede
	V > mangia

(4) scrivere, una per volta, le seguenti frasi (**step 1 – write a sentence**) e provare il parsing (**step 3 – parse it!**) con vari algoritmi (**step 2 - select a parsing algorithm**):
1. l'uomo vede la mela
2. l'uomo mangia la mela
3. il gatto mangia il topo
4. un gatto mangia un topo
5. un topo mangia un uomo
6. un topo mangia il uomo
7. l'topo vede l'gatto
8. il topo vede gatto
9. la gatto vede la uomo
10. ...

(5) modificare il lessico in modo da catturare l'accordo

(6) introdurre ambiguità nel lessico (D > la; pro > la... S > DP VP, S > VP) e provare diversi algoritmi di parsing

(7) costruire una grammatica per parseggiare (in tutti i suoi sensi possibili) le classiche frasi:
1. la vecchia legge la regola
2. Gianni vede un uomo nel parco con il cannocchiale

Dump database del lessico (nome del gatabase: "grammar")

```
CREATE DATABASE `grammar`;
USE grammar;

#
# Struttura della tabella `v`
#
```

```
CREATE TABLE `v` (
  `id` tinyint(4) NOT NULL auto_increment,
  `left` varchar(12) NOT NULL default '',
  `right` varchar(12) NOT NULL default '',
  PRIMARY KEY  (`id`),
  KEY `left` (`left`,`right`)
) TYPE=MyISAM AUTO_INCREMENT=47 ;

#
# Dump dei dati per la tabella `v`
#

INSERT INTO `v` VALUES (46, 'PP', 'P DP');
INSERT INTO `v` VALUES (40, 'VP', 'V DP PP');
INSERT INTO `v` VALUES (37, 'DP', 'D N');
INSERT INTO `v` VALUES (34, 'VP', 'V DP');
INSERT INTO `v` VALUES (41, 'VP', 'cl V');
INSERT INTO `v` VALUES (32, 'S', 'DP VP');
INSERT INTO `v` VALUES (42, 'DP', 'D A N');
INSERT INTO `v` VALUES (43, 'DP', 'N');
INSERT INTO `v` VALUES (44, 'VP', 'aux V DP');
INSERT INTO `v` VALUES (45, 'DP', 'D N PP');

#
# Struttura della tabella `vt`
#

CREATE TABLE `vt` (
  `id` tinyint(4) NOT NULL auto_increment,
  `left` varchar(5) default NULL,
  `right` varchar(12) default NULL,
  PRIMARY KEY  (`id`),
  KEY `left` (`left`),
  KEY `right` (`right`)
) TYPE=MyISAM PACK_KEYS=0 AUTO_INCREMENT=62 ;

#
# Dump dei dati per la tabella `vt`
#

INSERT INTO `vt` VALUES (4, 'D', 'il');
INSERT INTO `vt` VALUES (5, 'D', 'lo');
INSERT INTO `vt` VALUES (6, 'D', 'la');
INSERT INTO `vt` VALUES (7, 'D', 'i');
INSERT INTO `vt` VALUES (8, 'D', 'gli');
INSERT INTO `vt` VALUES (9, 'D', 'le');
INSERT INTO `vt` VALUES (10, 'P', 'di');
INSERT INTO `vt` VALUES (11, 'P', 'a');
INSERT INTO `vt` VALUES (12, 'P', 'da');
INSERT INTO `vt` VALUES (13, 'P', 'in');
INSERT INTO `vt` VALUES (14, 'P', 'con');
INSERT INTO `vt` VALUES (15, 'P', 'su');
INSERT INTO `vt` VALUES (16, 'P', 'per');
INSERT INTO `vt` VALUES (17, 'P', 'tra');
INSERT INTO `vt` VALUES (18, 'P', 'fra');
INSERT INTO `vt` VALUES (19, 'A', 'bella');
INSERT INTO `vt` VALUES (20, 'A', 'grande');
INSERT INTO `vt` VALUES (21, 'N', 'casa');
INSERT INTO `vt` VALUES (22, 'N', 'bambina');
INSERT INTO `vt` VALUES (23, 'N', 'gatto');
```

```
INSERT INTO `vt` VALUES (24, 'N', 'uomo');
INSERT INTO `vt` VALUES (25, 'V', 'bacia');
INSERT INTO `vt` VALUES (26, 'V', 'mangia');
INSERT INTO `vt` VALUES (33, 'D', 'uno');
INSERT INTO `vt` VALUES (34, 'D', 'un');
INSERT INTO `vt` VALUES (35, 'D', 'una');
INSERT INTO `vt` VALUES (36, 'D', 'l');
INSERT INTO `vt` VALUES (38, 'C', 'chi');
INSERT INTO `vt` VALUES (39, 'aux', 'ha');
INSERT INTO `vt` VALUES (40, 'V', 'mangiato');
INSERT INTO `vt` VALUES (41, 'N', 'mela');
INSERT INTO `vt` VALUES (42, 'V', 'ama');
INSERT INTO `vt` VALUES (43, 'A', 'bell');
INSERT INTO `vt` VALUES (44, 'C', 'che');
INSERT INTO `vt` VALUES (49, 'N', 'cannocchiale');
INSERT INTO `vt` VALUES (48, 'V', 'morde');
INSERT INTO `vt` VALUES (47, 'N', 'Luigi');
INSERT INTO `vt` VALUES (50, 'A', 'vecchia');
INSERT INTO `vt` VALUES (51, 'N', 'vecchia');
INSERT INTO `vt` VALUES (52, 'N', 'legge');
INSERT INTO `vt` VALUES (53, 'V', 'legge');
INSERT INTO `vt` VALUES (54, 'cl', 'la');
INSERT INTO `vt` VALUES (55, 'V', 'regola');
INSERT INTO `vt` VALUES (56, 'N', 'regola');
INSERT INTO `vt` VALUES (57, 'N', 'sbarra');
INSERT INTO `vt` VALUES (58, 'V', 'sbarra');
INSERT INTO `vt` VALUES (59, 'V', 'porta');
INSERT INTO `vt` VALUES (60, 'N', 'porta');
INSERT INTO `vt` VALUES (61, 'V', 'visto');
```

Programmazione algoritmi in PHP

(8) formalizzazione dell'algoritmo di Earley:

funzione Earley-Parser(*stringa_parole*, *grammatica*) **restituisce** *grafo*

metti_in_coda(($\gamma \rightarrow \bullet S$, [0,0]), *grafo*[0])

per ogni *i* **tale che** (*i* va da 0 a lunghezza(*stringa_parole*)) **esegui**
per ogni *stato* **tale che** (*stato* è in *grafo*[*i*]) **esegui**
se (incompleto(*stato*) **e non** parte_della_frase
(prossima_categoria(*stato*)))
allora esegui Previsione (*stato*)

altrimenti_se (incompleto(*stato*) **e**
parte_della_frase(prossima_categoria(*stato*)))
allora esegui Scansione (*stato*)

altrimenti esegui Completamento(*stato*)
fine
fine
restituisci *grafo*

(9) implementazione in PHP

```
function Earley($sentence) {

        $chart = array("");

        $chart[0][0]['left']= "start";
        $chart[0][0]['right'][0]= "S";
        $chart[0][0]['status'] = 0;
        $chart[0][0]['next'] = "S";
        $chart[0][0]['begin'] = 0;
        $chart[0][0]['current'] = 0;
        $chart[0][0]['function'] = "dummy";

        $this->chart = $chart;

        // Enqueue($state, 0);

        $input = $this->Tok_input($sentence);
        $this->input = $input;

        for ($i = 0; $i <= count($input); $i++) {

                for ($j = 0; $j <= count($this->chart[$i]); $j++) {

                                $current_state = $this->chart[$i][$j];

                        if ( $this->Incompete($current_state) && !$this->Part_of_speach($current_state['next'])) {

                                $this->Predictor($current_state);

                        } else if ( $this->Incompete($current_state) && $this->Part_of_speach($current_state['next'])) {
                                $this->Scanner($current_state);

                        } else {
                                $this->Completer($current_state);

                        }

                }
        }

        $this->do_track();
}
```

(10) formalizzazione singole funzioni:

procedura Previsione ($A \rightarrow \alpha \bullet B\beta$, $[i,j]$)
    per ogni ($B \rightarrow \gamma$) **tale che** (Regole_della_Grammatica(B, *grammatica*)) **esegui**
        metti_in_coda(($B \rightarrow \bullet\gamma$, $[j,j]$), *grafo*$[j]$)
fine
procedura Scansione ($A \rightarrow \alpha \bullet B\beta$, $[i,j]$)
    se (B è parte_della_frase(*parola*$[j]$)) **allora esegui**
        metti_in_coda(($B \rightarrow$ *parola*$[j]$, $[j,j+1]$), *grafo*$[j+1]$)
fine
procedura Completamento ($B \rightarrow \gamma \bullet$, $[j,k]$)
    per ogni ($A \rightarrow \alpha \bullet B\beta$, $[i,j]$) **tale che** (($A \rightarrow \alpha \bullet B\beta$, $[i,j]$) è nel *grafo*$[j]$) **esegui**
        metti_in_coda(($A \rightarrow \alpha B \bullet \beta$, $[j,k]$), *grafo*$[k]$)
fine

procedura metti_in_coda (*stato*, *entrata_grafo*)
se (*stato* non è già nell'*entrata_grafo*) **allora esegui**
inserisci(*stato*, *entrata_grafo*)
fine

(11) implementazione singole funzioni in PHP:

```
function Predictor ($current_state) {

$chart= $this->chart;

$j = $current_state['current'];

$rule = $this->Get_v_exp($current_state['next']);

for ($i=0; $i<count($rule); $i++) {

$state['left'] = $current_state['next'];
$state['right'] = $rule[$i][1];
$state['status'] = 0;
$state['next'] = $rule[$i][1][0];
$state['begin'] = $j;
$state['current'] = $j;
$state['function'] = "predictor";
$x = count($chart[$j]);
$chart[$j][$x] = $state;

}

$this->chart = $chart;

}
```

```
function Scanner ($current_state) {

$chart= $this->chart;
if ($this->Word_as_part_of_speach($this->input[$current_state['current']], $current_state['next']) ){
$state['left'] = $current_state['next'];
$state['right'][0] =  $this->input[$current_state['current']];
$state['status'] = $current_state['current']+1;
$state['next'] = "";
$state['begin'] = $current_state['current'];
$state['current'] = ($current_state['current']+1);
$state['function'] = "scanner";
$chart[($current_state['current']+1)][count($chart[$current_state['current']+1])] = $state;
        }
$this->chart = $chart;
}
```

```
function Completer ($current_state) {

$chart= $this->chart;

for ($i=0; $i<count($chart[$current_state['begin']]); $i++) {

        if ($chart[$current_state['begin']][$i]['next']==$current_state['left']) {

$state['left'] = $chart[$current_state['begin']][$i]['left'];
$state['right'] = $chart[$current_state['begin']][$i]['right'];
$state['status'] = $current_state['current'] - $chart[$current_state['begin']][$i]['begin'];
```

```
                $state['next'] =  $chart[$current_state['begin']][$i]['right'][$current_state['current'] -
                        $chart[$current_state['begin']][$i]['begin']];
                $state['begin'] = $chart[$current_state['begin']][$i]['begin'];
                $state['current'] = $current_state['current'];
                $state['function'] = "completer";

                $chart[($current_state['current'])][count($chart[$current_state['current']])] = $state;

// begin parse tree ...

                $this->parse_tree .= "[<sub>".$state['left']."</sub> ";
                for ($x=0; $x<=count($state['right']); $x++){
                                        $this->parse_tree .=  $state['right'][$x]." ";
                                        }

                $this->parse_tree .=        " ]<sub>".$state['left']."</sub>";

// ... parse tree end
                        }
                }

                $this->chart = $chart;

                }

        function Enqueue($next, $n){

                array_push($chart[$n], $next);
                $this->chart = $chart;

                }

        function Part_of_speach($next){
                $q = new neo;
                $query = sprintf("SELECT * FROM `vt` WHERE `left` = '$next'");
                        $result = $q->query($query);
                        $row = mysql_fetch_array($result);

        if ($row) {
                return true;
        } else {
                return false;
                }

                }

        function Word_as_Part_of_speach($word, $next){
                $q = new neo;
                $query = sprintf("SELECT * FROM `vt` WHERE `left` = '$next' and `right`='$word'");
                $result = $q->query($query);

        if ($row = mysql_fetch_array($result)) {
                return true;
        } else {
                return false; }}

        function Incompete($current_state){

                        if ($current_state['status'] < count($current_state['right'])) {
                                return true;
                        } else {
```

```
                                    return false; } }

        function Get_v_exp($left){

                $rule = array();
                $q = new neo;
                $n=0;

                $query = sprintf("SELECT * FROM `v` WHERE `left` = '$left'");
                $result = $q->query($query);

                while ($row = mysql_fetch_array($result)){

                $rule[$n][0] = $left;

                        $expand_right = strtok($row['right']," ");
                        $n_re = 0;
                                                while ($expand_right) {
                                                        $rule[$n][1][$n_re] = $expand_right;
                                                        $n_re++;
                                                                $expand_right = strtok(" ");
                                                                }
                        $n++;
                        }
                        return $rule;
                }

        function Tok_input($sentence){

        $sentence = str_replace("""," ", $sentence);

        $word = strtok($sentence," ");

        $n=0;

                while ($word) {
                        $input[$n]= $word;
                        $word = strtok (" ");
                        $n++;
                        }
                        return $input;
        }
```

(12) implementazione della classe parser

```
class parser {

    var $parse_tree = "";
    var $track = "";
    var $agenda = array();
    var $n = 0;
    var $i = 0;
    var $chart = array();

function Earley...

...

function Tok_input ...

}
```

(13) utilizzo della classe parser in un documento html standard:

```
<html>
	<head>
	<title>Implementazione Parser</title>
	</head>

	<body>

		testo qualsiasi....

	<?
	require("parsing.inc");
	$p = new parser;

	$parsing_info = "Parsing sentence '".$sentence."' using <b>".$alg."</b> parsing algorithm (grammar:
	<b>".$lang."</b>)<br>parsing begun... <br>";

		if ($alg=="top_down_multi") $pars = $p->top_down_multi($sentence);
		if ($alg=="earley") $pars = $p->Earley($sentence);

	if ($tracking) $track_v = "tracking: <br>".$p->show_track();

	$parse = $p->show_parse();

	echo $parsing_info."<p>".$parse."</p><p>".$track_v."</p>";

	?>

	</body>
</html>
```

Riferimenti

PAB si può liberamente utilizzare collegandosi al seguente indirizzo web:
http://www.ciscl.unisi.it/progetti/pab/

Apache (il Web Server, colui che invia le pagine web che vengono visualizzate con il browser) è scaricabile (versione windows o linux) qua:
http://www.apache.org

PHP (lo script engine che si integra con Apache e che genera il codice html partendo dagli script) è scaricabile (versione windows o linux) qua:
http://www.php.net

MySql (il database management system, quello che archivia sul server la grammatica) è scaricabile (versione windows o linux) qua:
http://www.mysql.com/

PhpMyAdmin, (utility che facilita molto la gestione di Mysql via web browser) è scaricabile qua:
http://www.phpmyadmin.net/

(chiaramente tutto il software è rigorosamente freeware)