Un primo sguardo al modulo publications

Luigi Scarso

Sommario

Presentiamo il modulo ConTEXt publications dedicato alla gestione della bibliografia. Sono illustrate le componenti interne e viene descritto un metodo per gestire il formato dei file utilizzato da Reference Manager® e EndNote® della Thomson Reuters.

Abstract

We introduce the ConTEXt module publications for the management of the bibliography. We decribe the components and how to implement a support for the foormat of the files used by Reference Manager® and EndNote® from Thomson Reuters.

1 Introduzione

Nel Novembre del 2013 l'Institut für Klassische und Romanische Philologie - Abteilung für Griechische und Lateinische Philologie dell'Università di Bonn (`http://www.philologie.uni-bonn.de/`) ha lanciato il progetto di edizioni critiche con ConTEXt-MKIV. Diretto dal Prof. Dr. Thomas A. Schmitz, direttore tecnico dell'Istituto, il progetto vede Hans Hagen come capo sviluppatore e l'autore come secondo sviluppatore ed intende avvalersi delle caratteristiche di LUATEX, attualmente uno dei motori tipografici TEX tra i più avanzati, sviluppando e modificando se necessario alcune parti di ConTEXt-MKIV per soddisfare i requisiti richiesti. È probabile che il progetto porti ad un modulo incluso nelle distribuzione *standalone* (`http://wiki.contextgarden.net/ConTeXt_Standalone`) ma non nel formato, quindi da caricare a richiesta. È altresì possibile, dato che H. Hagen e l'autore appartengono al team di sviluppo di LUATEX, che alcuni requisiti del progetto influenzino lo sviluppo del motore di impaginazione qualora si rivelassero colli di bottiglia, ma bisogna tenere presente che nel caso di LUATEX, contrariamente a ConTEXt, la tendenza attuale è quella di delegare quanto più possibile al formato esterno concentrandosi ad ottimizzare il codice C per migliorare le prestazioni, soprattutto la riduzione del tempo di esecuzione.

Sebbene le edizioni critiche siano un argomento specialistico, in ambito TEX è ben affrontato ed esiste una buona letteratura in merito. Senza la pretesa di voler essere esaurienti e limitandoci solo ai contributi italiani, ricordiamo il "Il progetto Maurolico" (`http://www.maurolico.unipi.it/mtex/mtex.htm`), il cui inizio è datato Dicembre 2000 (cfr. `http://www.maurolico.unipi.it/comite.htm`), il CeTEX, "Corso di Edizione Critica al computer con TeX/LaTeX" (`http://www.didaskalikos.org/CeTeX/`) del Prof. J. Leal della Pontificia Università della Santa Croce, il testo "Edizioni Critiche" (LEAL e PIGNALBERI, 2012) e, più recentemente, i due articoli (FADINI, 2012) e (JERÓNIMO LEAL AND GIANLUCA PIGNALBERI, 2012) presentati al GUIT meeting 2012. Oggi si tende a utilizzare XELATEX con il pacchetto ledmac oppure il più recente eledmac (cfr. `http://geekographie.maieul.net/Ledmac-est-mort-vive-Eledmac`), mentre i package tikz e xypic sono frequentemente usati per i stemma codicum.

La bibliografia è un aspetto fondamentale di una edizione critica, ma si tratta di un argomento meno settoriale — praticamente ogni articolo, report o libro sia in ambito scientifico che in ambito umanistico riporta una bilbiografia minima. Anche in questo caso esiste un package di riferimento, biblatex, spesso associato al motore "bibliografico" `biber` (cfr. `http://biblatex-biber.sourceforge.net/`) che offre completo supporto a UNICODE ed una gestione della memoria molto migliore rispetto al tradizionale `bibtex`; inoltre è in grado di mostrare graficamente le relazioni tra le voci di una bibliografia (*cross-references*) mediante il programma GRAPHVIZ.

È innegabile che in questo ambito ConTEXt-MKIV non gode attualmente della stessa popolarità di LATEX e non vi sono quindi moduli specifici per le edizioni critiche, anche se in passato alcuni tentativi sono stati fatti in tal senso.[1] ConTEXt ha anche un proprio modulo per le bibliografie, ma è piuttosto essenziale. Il progetto vuole quindi "colmare" questa lacuna, tenendo certamente ben presente le funzionalità offerte dai packages di LATEX ma con alcune decisioni precise:

- per le edizioni critiche, il formato XML-TEI come base di partenza per il testo;
- il formato bib di BIBTEX per le bibliografie, ma non il formato bbl — questo implica che le macro LATEX *non* sono riconosciute;
- al fine di ridurre le dipendenze, l'utilizzo esclusivo dei programmi

1. *Steps needed to expand ConTeXt to the demands of Critical Edition typesetting*, I.S. Hamid and T.A. Schmitz I° ConTEXt meeting 2007, `http://meeting.contextgarden.net/2007/share/idris/cr-apparatus.pdf`

LuaTEX/LuajitTEX,[2] Metapost e ConTEXt.

L'ultimo punto probabilmente merita ulteriori spiegazioni, in quanto discende direttamente da una sostanziale differenza tra i due formati LATEX e ConTEXt: il primo ha un kernel relativamente semplice e delega le funzionalità particolari a packages esterni; il secondo tende viceversa ad inglobare le funzionalità nel formato, o comunque in moduli caricali a tempo di esecuzione ma presenti nella distribuzione standalone, ed utilizza esclusivamente Lua come linguaggio di programmazione general-purpose. Ad esempio, un programma come biber verrebbe realizzato in Lua con il duplice vantaggio sia di renderlo immediatamente disponibile per tutte le piattaforme che supportano LuaTEX, sia di riusare il codice Lua già presente in ConTEXt. Inoltre biber è, sostanzialmente, un interprete Perl staticamente linkato con il sorgente Perl di biber ed i relativi moduli per un totale di circa 19 MiB, ai quali vanno sommati circa 19 MiB di XeLATEX; per contro la dimensione di LuaTEX è di circa 8 MiB.[3]

Allo stato attuale la parte di edizioni critiche è nella sua fase iniziale, mentre il modulo relativo alla bibliografia, scritto ex-novo, è disponibile nella distribuzione standalone — e lo sarà anche nella prossima TEXLive 2014, a meno di imprevisti: si tratta infatti di un modulo ancora in fase di sviluppo, quindi non "production-ready", e questo implica che è ancora possibile valutare richieste e/o suggerimenti da parte della comunità.

Nelle prossime sezioni presenteremo brevemente il modulo e una estensione proposta dall'autore per gestire il formato RIS utilizzato dal sistema Reference Manager di Thomson Reuters.

2 Il modulo publications

2.1 Datasets

In ConTEXt il modulo relativo alla bibliografia è chiamato publications e si basa attualmente su BibTEX (per questo motivo le macro utilizzano lo spazio dei nomi `btx`). Il file viene caricato interamente in memoria e convertito in una table di Lua, utilizzata come base per le successive elaborazioni. La prima osservazione riguarda il fatto che in un file bib normalmente sono presenti macro di LATEX non implementate in ConTEXt od implementate in modo diverso. In questo caso l'elaborazione *non* si ferma, ma viene emesso su log un messaggio opportuno:

```
publications > start used btx commands
publications > standard CONTEXT 1 known
publications > standard ConTeXt 4 known
publications > standard TeXLive 3 KNOWN
publications > standard eTeX    1 known
publications > standard hbox    6 known
publications > standard sltt    1 unknown
```

La macro non implementata viene inoltre stampata nel documento con un carattere a spaziatura fissa, per essere più facilmente rintracciabile durante la correzione delle bozze. Una macro può essere (ri)definita con il comando `\definebtxcommand`, come in

```
\definebtxcommand\TUB {TUGboat}
\definebtxcommand\sltt{\tt}
```

Ad esempio, il seguente file bib

```
@Article{sometag,
    author  = "An Author and Another One",
    title   = "A hopefully
\sltt{meaningful} title",
    journal = maps,
    volume  = "25",
    number  = "2",
    pages   = "5--9",
    month   = mar,
    year    = "2013",
    ISSN    = "1234-5678",
}
```

viene convertito internamente nella table

```
{
 ["author"]="An Author and Another One",
 ["category"]="article",
 ["index"]=1,
 ["issn"]="1234-5678",
 ["journal"]="maps",
 ["month"]="3",
 ["number"]="2",
 ["pages"]="5--9",
 ["tag"]="sometag",
 ["title"]="A hopefully
\\btxcmd{sltt}{meaningful} title",
 ["volume"]="25",
 ["year"]="2013",
}
```

Sorge spontaneo immaginare un analogo formato Xml ed infatti la macro
`\convertbtxdatasettoxml[<dataset>]`
converte un file bib (o più correttamente un *dataset*, cfr. oltre) in modo diretto:

```
<?xml version='1.0' standalone='yes'?>
<bibtex>
 <entry tag="sometag" category="article"
                                index="1">
  <field name="author">An Author
                 and Another One</field>
  <field name="issn">1234-5678</field>
  <field name="journal">maps</field>
  <field name="month">3</field>
  <field name="number">2</field>
  <field name="pages">5--9</field>
  <field name="title">A hopefully
\btxcmd{sltt}{meaningful} title</field>
```

2. LuajitTEX dovrebbe essere distribuito con la TEXLive 2014

3. LuajitTEX ha una dimensione di circa 8.5 MiB. Le dimensioni si riferiscono ad eseguibili Linux.

```
  <field name="volume">25</field>
  <field name="year">2013</field>
 </entry>
</bibtex>
```

Il formato XML può anche essere letto in alternativa al file bib. Infine, è possibile utilizzare file in stile TEX:

```
\startpublication
  \category{article}
  \tag{sometag}
  \author{An Author and Another One}
  \category{article}
  \index{1}
  \issn{1234-5678}
  \journal{maps}
  \month{3}
  \number{2}
  \pages{5--9}
  \tag{sometag}
  \title{A hopefully
\btxcmd{sltt}{meaningful} title}
  \volume{25}
  \year{2013}
\stoppublication
```

È quindi possibile avere diversi file che contengono riferimenti bibliografici: i file costituiscono un *dataset* associato ad un identificativo (standard nell'esempio) nel modo seguente:

```
\definebtxdataset[standard]
\usebtxdataset[standard][bibl.bib]
\usebtxdataset[standard][bibl.lua]
\usebtxdataset[standard][bibl.xml]
\usebtxdataset[standard][bibl.tex]
```

Un dataset è un elemento della Lua table `publications.datasets`
ed è a sua volta una table, ad esempio i campi `luadata` e `xmldata` contengono rispettivamente il formato Lua ed XML[4] del dataset:

```
\startluacode
local datasets= publications.datasets
for _,dataset in pairs(datasets) do
 table.print(dataset.luadata)
 print(dataset.xmldata)
end
\stopluacode
```

È buona norma specificare sempre l'identificativo: allo stato attuale esiste un solo dataset, standard, ma è previsto il supporto per più dataset.

2.2 Presentazione di un dataset

Come visto precedentemente, la struttura dati di base di publications è una table Lua e quindi necessita di una "presentazione" o *rendering* — in breve, lo stile della bibliografia. La macro `\definebtxrendering` associa un dataset ad un rendering:

```
\definebtxrendering
  [standard:apa]
  [dataset=standard,
   alternative=apa]
```

In questo esempio il dataset standard è associato all'alternativa apa[5] (i.e. lo stile) ed è definito nel file `publ-imp-apa.mkiv`. Il "nome" (o chiave) di questo rendering è `standard:apa` ed è definito dall'utente. Per il momento apa è l'unica alternativa ad essere implementata, ma è sufficientemente ricca da poter essere usata come base di partenza per altri stili: non ci addentreremo quindi in tutti i dettagli, limitandoci ai punti salienti. Ogni campo (*field*) di una voce (*entry*) della bibliografia è gestito tramite un *setup* definito tramite la macro `\startsetups btx:apa`*<field>* ... `\stopsetups` e richiamato con `\btxsetup{`*<field>*`}` come nel caso seguente, dove *<field>* è `article`:

```
\startsetups btx:apa:article
 \btxsetup{btx:apa:common:author-or-key-and-
year}
 \btxdoif {title} {
  \btxflush{title}\btxperiod
 }
 \btxdoifelse {journal} {
  \bgroup\it\btxflush{journal}\/\egroup
 } {
    \btxdoif {crossref} {
     In\btxspace\btxflush{crossref}
   }
 }
 \btxdoifelse {volume} {
  \btxcomma
\bgroup\it\btxflush{volume}\/\egroup
  \btxdoif {issue} {
  \btxlparent\btxflush{issue}\btxrparent
  }
  \btxdoif {pages} {
   \btxcomma\btxflush{pages}
  }
  \btxperiod
 } {
    \btxsetup{btx:apa:common:pages:pp}
 }
 \btxsetup{btx:apa:common:note}
 \btxsetup{btx:apa:common:comment}
\stopsetups
```

Il file è quindi caricato con il comando
`\loadbtxdefinitionfile[apa]`
dove, come si nota, il nome dello stile apa funge da chiave. All'interno del documento il riferimento alle voce della bibliografia si ottiene con la macro `\cite` (i.e. `\cite[sometag]`) e l'elenco delle pubblicazioni tramite
`\placelistofpublications[standard:apa]`
(letteralmente: "mettere la lista delle pubblicazioni standard con stile apa"):

4. La conversione deve essere attivata dall'utente con `\convertbtxdatasettoxml[standard]`.

5. American Psychological Association cfr. Anonymous (2009)

```
\definebtxdataset[standard]
\usebtxdataset[standard][bibl.bib]
\usebtxdataset[standard][bibl.lua]
\usebtxdataset[standard][bibl.xml]
\usebtxdataset[standard][bibl.tex]
\starttext
A citation from the dataset\cite[sometag]
\placelistofpublications[standard:apa]
\stoptext
```

Le proprietà di un rendering si possono modificare con `\setuprendering` (i valori di default sono tra parentesi):

```
\setupbtxrendering [<name>][.=.]
alternative = <TEXT> ("apa")
dataset     = <TEXT> ("standard")
setups      = <TEXT> ("btx:rendering:apa")
method      = "local"|"global"|"none"|"force"
              ("global")
sorttype    = "short"|"reference"|"dataset"|
              "default"
criterium   = "here"|"cite"|"all" ("here")
refcommand  = <TEXT> ("authoryears")
numbering   = "yes"|"cite" ("yes")
width       = <DIMENSION>|"auto" ("auto")
distance    = <DIMENSION> ("1.5\emwidth")
```

Menzioniamo solo due chiavi, `method` e `criterium`. La chiave `method` seleziona come mostrare la lista:
`local` è usato per le liste locali ad una sezione (e.g. alla fine di ogni capitolo);
`global` quando la lista viene mostrata una sola volta nel documento (e.g. alla fine di un libro);
`none` non mostra la lista (ma continua a gestirla);
`force` mostra tutti gli elementi della lista, anche eventuali duplicati (utile per un controllo).
La chiave `criterium` mostra tutti gli elementi della bibliografia anche se non citati (`all`), oppure solo quelli citati (`here` o `cite`).

È possibile definire il dataset di default con `\setupbtxrendering[dateset=<TEXT>]`: in questo modo il nome è sottinteso e non è necessario specificarlo ulteriormente (ad esempio con `\setupbtxrendering[dateset=standard]` e `\setupbtxrendering[method=local]` si modifica `method` del datset `standard`).

È previsto anche un meccanismo per gestire le *varianti* di lista, ovvero come mostrare i nomi, cognomi, mesi e l'ordinamento:

```
\setupbtxlistvariant [<name>][.=.]
namesep               = <TEXT> (",")
lastnamesep           = <TEXT> ("and")
finalnamesep          = <TEXT> ("and")
firstnamesep          = <TEXT>
juniorsep             = <TEXT>
vonsep                = <TEXT>
surnamesep            = <TEXT>
surnamejuniorsep      = <TEXT> (",")
juniorjuniorsep       = <TEXT>
surnamefirstnamesep   = <TEXT> (",")
surnameinitialsep     = <TEXT> (",")
etallimit             = <TEXT> ("5")
etaldisplay           = <TEXT> ("5")
etaltext              = <TEXT> ("et al."=
monthconversion       = "number" "month"
                         "month:mnem"
                        ("number")
authorconversion      = "normal" "inverted"
                        "normalshort"
                        "invertedshort"
                        ("normal")
```

Attualmente le varianti sono `"author"`, `"editor"`, `"artauthor"` e *non* sono legati alle alternative, ma condivise.

Infine `\setupbtxlist` imposta i parametri generali (colore, stile, allineamento...) della lista:

```
\setupbtxlist [<name>][.=.]
alternative  = <TEXT> ("left")
style        = <TEXT>
color        = <TEXT>
headstyle    = <TEXT>
headcolor    = <TEXT>
width        = <DIMENSION> ("4\emwidth")
distance     = <DIMENSION> ("\emwidth")
hang         = <NUMBER>
align        =
headalign    =
margin       = "cd:yes" "cd:no"  ("cd:no")
before       = <COMMAND> ("\blank")
inbetween    = <COMMAND> ("\blank")
after        = <COMMAND> ("\blank")
display      = "cd:yes" "cd:no" ("yes")
command      = <COMMAND>
```

I campi attualmente riconosciuti sono:
`abstract`, `address`, `annotate`, `assignee`, `author`, `bibnumber`, `booktitle`, `chapter`, `comment`, `country`, `day`, `dayfiled`, `doi`, `edition`, `editor`, `eprint`, `howpublished`, `institution`, `isbn`, `issn`, `journal`, `key`, `keyword`, `keywords`, `language`, `lastchecked`, `month`, `monthfiled`, `names`, `nationality`, `note`, `notes`, `number`, `organization`, `pages`, `publisher`, `revision`, `school`, `series`, `size`, `title`, `type`, `url`, `volume`, `year`, `yearfiled`.

2.3 Citazioni

L'ultima componente del modulo è costituita dalle citazioni, indipendenti dalle alternative e varianti di lista. Il comando è `\cite[<nome>][<tag>]`, dove <nome> è opzionale (default: `num`) e fa riferimento anche in questo caso ad una variante, mentre <tag> è obbligatorio e fa riferimento ad una voce del dataset con il corrispondente `tag`. Una variante di citazione è prima definita con `\definebtxcitevariant`:

```
\definebtxcitevariant [<name1>][<name2>]
```

dove `<name2>` è opzionale, ma se presente implica che `name1` è una copia di `name2`; i suoi parametri si possono poi modificare con `\setupbtxcitevariant`:

```
\setupbtxcitevariant [<name>][.=.]
alternative = <TEXT> ("num")
setups      = <TEXT> ("btx:cite:num")
interaction = <TEXT> ("start")
andtext     = <TEXT> ("and")
otherstext  = <TEXT> ("et al.")
compress    = <TEXT> ("no")
putsep      = TEXT
lastputsep  = TEXT
inbetween   = TEXT
right       = TEXT
middle      = TEXT
left        = TEXT
```

Le alternative attualmente sono:
`"author"`, `"authornum"`, `"authoryear"`, `"authoryears"`, `"doi"`, `"key"`, `"num"`, `"page"`, `"page"`, `"serial"`, `"short"`, `"type"`, `"url"`, `"year"`.
Anche in questo caso è possibile definire la variante di default:
`\setupbtxcitevariant[alternative=<TEXT>]`
Inoltre, dato che in un documento è possibile definire diversi dataset, `\cite` permette di specificare la coppia `<dataset>::<tag>`, i.e. `\cite[standard::sometag]` per in individuare la voce della bibliografia nel dataset `<datset>` (cfr. figura 1).

3 Il formato RIS in publications

RIS è il formato bibliografico usato dai programmi Reference Manager® e EndNote® della Thomson Reuters. È analogo al formato bib (ma non gestisce elementi tipografici) ed è descritto in `http://www.refman.com/support/risformat_fields_01.asp` e link seguenti; un esempio è riportato nella figura 2. Un file RIS è una sequenza di record (i.e. una voce della bibliografia) suddiviso in campi (*fields*); ogni campo inizia con una etichetta (*tag*) ed ha un contenuto. Il formato utilizza la IBM Extended Character Set (più precisamente un suo sottoinsieme), quindi *non* è Unicode®. Un tag occupa 6 byte ed è del tipo *<tagname>*␣␣-␣*<content>*, dove i tagname sono descritti nelle specifiche e - è il carattere ASCII *0x2D* ("segno meno"). Il contenuto dipende dal tag: in alcuni casi è libero, in altri è limitato in lunghezza, in altri ancora assume valori da un insieme predefinito. Ogni record *deve* iniziare con TY␣␣-␣*<reference type>* (dove *<reference type>* sono 4 byte che identificano il tipo di voce come abstract, report etc.) e *deve* terminare con ER␣␣-␣ ; tra questi due vi *possono essere* i rimanenti campi il cui ordine non è rilevante.

Per il parsing di un file RIS è naturale usare *LPeg*, il modulo Lua per le *Parsing Expression Grammar*. Nella prossima sezione daremo una panoramica del modulo, con l'avvertenza che si tratta di materiale la cui comprensione può non risultare immediata ad una prima lettura e comunque limitato alle funzioni usate per il formato RIS. Le Peg sono diffusamente usate in ConTEXt e il modulo publications non fa eccezione.

3.1 LPeg: Patterns

Senza entrare in dettagli tecnici (per i quali rimandiamo a Grune e Jacobs (2007), sez. 15.7 per un inquadramento generale ed a Ierusalimschy (2009) e `http://www.inf.puc-rio.br/~roberto/lpeg` per l'implementazione in Lua), una Peg è un insieme di *pattern* combinati tramite le operazioni `+`,`*`,`-` e `^`. Un pattern *terminale* è la stringa vuota ϵ oppure un intero d con $0 \leq d \leq 255$ espresso tramite una stringa, in modo tale che ad esempio `""` è la stringa vuota ϵ (come anche `''` e `[=[]=]`, visto che entrambe sono stringhe valide in Lua) mentre `"a"` corrisponde — se si usa una codifica compatibile con Ascii — all'intero 97 (in alternativa `"\97"` o `"\097"` sono entrambi stringhe Lua equivalenti a 97). A parte ϵ, si tratta comunque di valori rappresentati semplicemente con un singolo byte, per cui LPeg offre le seguenti funzioni per specificare pattern più complessi:

- `lpeg.P(`<str>`)`: è il pattern della stringa *str*, ovvero la sequenza (da sinistra verso destra) degli interi d che, nella codifica in uso, corrisponde alla stringa *str*. Così `lpeg.P("")` è ϵ, e, se si usa la codifica Utf-8, `lpeg.P("ac")` è la sequenza $97, 99$, `lpeg.P("ò")` è la sequenza $195, 179$. *Nota bene:* se si una una codifica diversa da Utf-8, i valori della sequenze cambiano di conseguenza. Ad esempio, con IBM Extended Character Set, `lpeg.P("ò")` è la sequenza 149 — del tutto diversa dal caso precedente;

- `lpeg.P(`<n>`)`: se $n = 0$ è la stringa vuota, se $n > 0$ corrisponde ad un pattern di esattamente n terminali (i.e. una stringa di n byte) e se $n < 0$ è un pattern di *al massimo* n byte. I valori dei byte non sono importanti;

- `lpeg.P(`<pattern>`)`: se `P1` è un pattern `lpeg.P(P1)` è `P1` (i.e. in questo caso `lpeg.P` è la funzione identità);

- `lpeg.P(`<boolean>`)`: se *boolean* è `true` il pattern riconosce qualsiasi input (pattern true `T`), se è `false` il pattern rigetta qualsiasi input (pattern false `F`) ;

- `lpeg.R(`<range>`)`: se *range* è una stringa di due byte d_1d_2 e $d_1 \leq d_2$, il pattern riconosce un carattere $c \in \{d_1, d_1 + 1 \ldots d_2\}$; se $d_1 > d_2$ è il pattern false `F`. La funzione accetta anche un array di *range* ed in questo caso il pattern è l'unione dei pattern dei singoli elementi; `lpeg.R({"az", "AZ"})` è il pattern delle lettera minuscola o maiuscola (in codifica compatibile Ascii). Chiamata senza argomento `lpeg.R()` ritorna il pattern `F`;

- `lpeg.S(<str>)`: è il pattern unione dei terminali in *str*. Ad esempio `lpeg.S("+-*/")` è il pattern che riconosce un carattere $c \in \{$`"+"`,`"-"`,`"*"`,`"/"`$\}$; `lpeg.S("")` è il pattern `F`.

3.2 LPeg: Operazioni & Captures

Come accennato precedentemente, sui pattern sono definite quattro operazioni, `+`, `*`, `-`, `^`, con il seguente significato:

- `P3=P1+P2`: il pattern `P3` tenta di riconoscere `P1`, ed in caso di successo procede oltre tralasciando `P2`. In caso di fallimento, torna al punto di partenza (*backtracking*) e tenta `P2`. È una scelta *ordinata* e quindi *non commutativa*: `P("")+P("a")`, `P("a")+P("")` e `P("a")` sono 3 pattern diversi, — questo fatto renda la somma di due pattern piuttosto inusuale. Solo se i `P1` e `P2` sono insiemi vale `P1+P2 = P2+P1`; se `P1` un qualsiasi pattern allora `P1+F = F+P1 = P1` (i.e. `F` è l'identità dell'operazione `+`);
- `P3=P1*P2`: il pattern `P3` tenta di riconoscere `P1`, ed in caso di successo procede oltre tentando `P2`. È commutativa, con identità `T`: `P1*T = T*P1 = P1`;
- `P3=-P1`: il pattern `P3` riconosce tutto tranne `P1`. Ad esempio `-lpeg.P(1)` riconosce qualsiasi pattern che non sia un carattere — i.e. la stringa vuota;
- `P3=P1-P2`: il pattern `P3` riconosce `P1` *se non riconosce* `P2`. Detto in altro modo, `P1-P2=(-P2)*P1`;
- `P3=P1^n`: riconosce *almeno* n copie del pattern `P1`, inclusa ϵ se $n = 0$. Se $n < 0$ riconosce *al massimo* $|n|$ copie di `P1`.

La funzione `lpeg.match(P,s,[j])` verifica se il pattern `P` è nella stringa *s partendo dall'inizio di s* se `j` è omesso; se la verifica è positiva ritorna la posizione del primo byte all'interno di *s dopo* il match, altrimenti `nil`. Ad esempio `lpeg.match(lpeg.P('a'),'baaaa')` ritorna `nil`, `lpeg.match(lpeg.P('a'),'aaaab')` ritorna `2`, `lpeg.match(lpeg.P(''),'aaaab')` ritorna `1`. Il terzo parametro opzionale `j` fa partire il matching dalla posizione *j*. Marcando in modo opportuno un pattern `P`, la funzione `match` può anche ritornare la stringa del pattern "catturata" (*capture*) all'interno della stringa *s*. La marcatura è semplice: la funzione `lpeg.C(P)` ritorna un pattern uguale a `P`, ma `lpeg.match(lpeg.C(P))` ritorna la stringa invece di una posizione. Quindi:

```
match =  lpeg.match
P,C   =  lpeg.P, lpeg.C
print("1:>"..match(C(P('a')^0),'aaaab')..'<')
print("2:>"..match(C(P('a')^0),'baaaa')..'<')
print("3:", match(C(P('a')^1),'baaaa'))
```

produce

```
1:>aaaa<
2:><
3: nil
```

(la seconda capture è la stringa vuota ϵ). Infine, le capture sono ricorsive — ad esempio `C(P1+C(P2)*P3)` è una capture valida — e `lpeg.Ct(P)` ritorna una `table` con tutte le capture del pattern `P`.

3.3 Una Peg per il formato RIS

ConTEXt mette a disposizione pattern di uso comune che risultano comodi da utilizzare:

```
local patterns = lpeg.patterns
local uppercase = patterns.uppercase
local digit = patterns.digit
local whitespace = patterns.whitespace
local spacer = patterns.spacer
local newline = patterns.newline
local anything = patterns.anything
```

È inoltre conveniente utilizzare gli alias

```
local P, R, S, V, Ct, C, Cs, Cc, Cp, Cmt =
  lpeg.P, lpeg.R,lpeg.S, lpeg.V, lpeg.Ct,
  lpeg.C, lpeg.Cs, lpeg.Cc,lpeg.Cp,lpeg.Cmt
```

Entriamo ora nel vivo delle specifiche. Ricordiamo che un file è formato da uno o più record, un record ha uno o più field, un field ha un tag ed un contenuto. Un tag ha un tagname di due caratteri che sono elencati nelle specifiche; l'insieme dei tagname è quindi la somma dei singoli tagname:

```
-- Not ER
local tagname = P('A1')+P('A2')+P('A3')
+P('AB')+P('AD')+P('AU')+P('AV')+P('BT')
+P('CT')+P('CY')+P('ED')+P('EP')+P('ID')
+P('IS')+P('J1')+P('J2')+P('JA')+P('JF')
+P('JO')+P('KW')+P('L1')+P('L2')+P('L3')
+P('L4')+P('M1')+P('M2')+P('M3')+P('N1')
+P('N2')+P('PB')+P('PY')+P('RP')+P('SN')
+P('SP')+P('T1')+P('T2')+P('T3')+P('TI')
+P('TY')+P('TY')+P('U1')+P('U2')+P('U3')
+P('U4')+P('U5')+P('UR')+P('VL')+P('Y1')
+P('Y2')
local alltagname = tagname + P('ER')
```

Il pattern di un tag è altrettanto immediato: dato che il tag che chiude un record è costante ($\texttt{ER}_{\sqcup\sqcup}\texttt{-}_{\sqcup}$), non serve memorizzarlo — deve però essere riconosciuto:

```
local dash=P("-")
local tag=C(tagname)*spacer*spacer*dash
 *spacer+(P('ER')*spacer*spacer*dash
 *spacer^0)
```

L'ultimo pattern dovrebbe essere `spacer`, ma tolleriamo un numero arbitrario di spazi. È conveniente definire i pattern dei field di apertura `TY` e chiusura `ER`:

```
local valid_typeref_values = P('ABST')
+P('ADVS')+P('ART')+P('BILL')+P('BOOK')
+P('CASE')+P('CHAP')+P('COMP')+P('CONF')
+P('CTLG')+P('DATA')+P('ELEC')+P('GEN')
+P('HEAR')+P('ICOMM')+P('INPR')+P('JFULL')
+P('JOUR')+P('MAP')+P('MGZN')+P('MPCT')
+P('MUSIC')+P('NEWS')+P('PAMP')+P('PAT')
+P('PCOMM')+P('RPRT')+P('SER')+P('SLIDE')
+P('SOUND')+P('STAT')+P('THES')+P('UNBIL')
+P('UNPB')+P('VIDEO')
local typeref_tagname = P("TY  - ")
local typeref_field   = typeref_tagname
          *C(valid_typeref_values)*newline

local endref_tagname = P("ER  -")*spacer^0
local endref_field = endref_tagname*newline^0
```

Il pattern `C(valid_typeref_values)` cattura il tipo di voce della bibliografia (book, article, report...) e quindi deve essere salvato.

Per i rimanenti field seguiamo la seguente strategia: definiamo prima di tutto i pattern per i contenuti validi

```
local valid_id_characters = R("09")+R("AZ")
local valid_characters    = R("\32\255")
local valid_restricted_characters=
                  valid_characters-P("\42")
```

Poi definiamo i field tipo `author`, `date`, `keyword`, `id`, `periodical` e `reprint` perché hanno un contenuto diverso dagli altri field (per brevità riportiamo solo il field `id`; per il programma completo cfr. figure 3-6):

```
local id_tagname =
  C(P("ID"))*P("  - ")
local id_content =
  C((valid_id_characters-(newline*tag))
   *(valid_id_characters-(newline*tag))^-19)
local id_field  =
   Ct(id_tagname*id_content)*newline
```

Il field `id` ha come unico tagname `ID` ed un contenuto fatto di caratteri alfanumerici — numeri e lettere solo maiuscole — lungo minimo un carattere e massimo venti (pattern `valid_id_characters`); come tutti i field termina con un pattern `newline`. Una volta definiti i field particolari, vediamo il pattern che cattura i field rimanenti:

```
local other_tagname = tag
 -(typeref_tagname + endref_tagname
   + author_tagname + date_tagname
   + keyword_tagname + id_tagname
   + periodical_tagname + reprint_tagname)
local other_content = C((anything
               - (newline*tag))^0)
local other_field = Ct(other_tagname
          * other_content)* newline
```

Il tagname di un field rimanente è semplice: *non* è né un tagname particolare né di inizio o fine record. Il contenuto del field è costituito da tutti i caratteri fino al successivo tag, `newline` esclusa: nessuna limitazione sul valore dei caratteri e sulla lunghezza.

A questo punto è immediato definire il pattern di un field generale compreso quello iniziale e quello finale:

```
local ris_field  = author_field + date_field
                + keyword_field + id_field
                + periodical_field
                + reprint_field + other_field
```

Un record è un field iniziale seguito da almeno un field generale e terminante con il field finale:

```
local ris_record = Ct(typeref_field
                     * Ct(ris_field^1)
                     * endref_field)
```

Infine, un file contiene almeno un record e noi ignoriamo eventuali spazi inziali:

```
local ris_file = (whitespace)^0
        *Ct((ris_record*newline^0)^1)
```

La parte finale è semplice: con `lpeg.match(ris_lpeg,ris_data)` si ottiene un iteratore su tutti i record, ed ogni record viene trasformato in una table con `transcode_table`:

```
-- Translate a RIS file into
-- a publications Lua file
local function ris_to_lua_table(ris_lpeg,
                          ris_data,prefix)
 local lua_table= {}
 local prefix = prefix or
             publications.format.ris.prefix
 for i,record in
      pairs(lpeg.match(ris_lpeg,ris_data)) do
   local TY        = record[1]
   local rec_table =
                  transcode_table(record[2])
   local idx = tostring(prefix)..tostring(i)
   lua_table[idx]={}
   lua_table[idx].abstract =
          merge_fields(rec_table," ","N2")
   lua_table[idx].address  =
          merge_fields(rec_table," ","AD")
   lua_table[idx].author   =
          merge_fields(rec_table,"","A1","AU",
                           "A2","ED","A3")
   lua_table[idx].category =
          categories[TY]
   lua_table[idx].issn     =
          merge_fields(rec_table," ", "SN")
   lua_table[idx].keywords =
          merge_fields(rec_table," ","KW")
   lua_table[idx].note     =
          merge_fields(rec_table,"; ",
                       "N1","AB","AV",
       "M1","M2","M3",
```

```
      "U1","U2","U3","U4","U5")
  lua_table[idx].publisher=
        merge_fields(rec_table," ", "PB")
  lua_table[idx].title    =
        merge_fields(rec_table," -- ",
                              "T1","TI","CT",
                        "BT","T2","BT","T3")
  lua_table[idx].url      =
        merge_fields(rec_table," ", "UR")
  lua_table[idx].year     =
        merge_fields(rec_table,"; ",
                            "Y1","P1","Y2")
 end
 return lua_table
end
```

È importante notare che il riconoscimento è fatto sull'intero file precedentemente caricato in memoria e questo può essere problematico in dispositivi con memoria limitata.

La funzione `ris_to_lua_table` è privata: il modulo publications ha una interfaccia pubblica data dalla funzione
`publications.`*<formato>*`_to_btx(data)`
dove *<formato>* è il suffisso del file, nel nostro caso `ris`:

```
function publications.ris_to_btx(ris_data)
  return ris_to_lua_table(ris_file,
            ris_data,
            publications.format.ris.prefix)
end
```

Infine, `loaders.`*<formato>*`(dataset,filename)` è la funzione pubblica per caricare il file nel dataset:

```
function loaders.ris(dataset,filename)
    loaders.lua(dataset,
      publications.
      ris_to_btx(io.loaddata(filename)
                              or ""))
end
```

In un file TEX la macro `\usebtxdatase` carica un file RIS in modo del tutto analogo ad un file bib o xml visto in precedenza, utilizzando però `loader.ris` (la chiave è quindi la desinenza `ris`): i.e. `\usebtxdatase[standard][test.ris]` aggiunge il file `test.ris` al dataset standard.

4 Conclusioni

Il modulo publications di ConTEXt ambisce ad emulare biblatex e `biber` senza dipendere da programmi esterni, ma sfruttando le librerie già presenti in LuaTEX. In particolare riteniamo che Lua e le Parsing Expression Grammar siano un valido sostituto del Perl e questo, a parità di motore di impaginazione, porta ad un dimezzamento della memoria occupata. La possibilità di gestire formati diversi, ed in particolare un formato Xml, aggiunge un flessibilità probabilmente non ottenibile attualmente con biblatex.

Il modulo attualmente è ancora in fase di sviluppo e non è stata esplorata la possibilità di produrre diagrammi delle relazioni delle voci di bibliografia, funzionalità offerta da `biber` mediante Metapost(il numero di voci del formato `bibtex` è inferiore a quello di biblatex e probabilmente anche l'unico stile Apa deve essere perfezionato. La documentazione è ancora incompleta.

Il team si propone di completare la prima versione per la fine del 2014.

Riferimenti bibliografici

Anonymous (2009). *Publication Manual of the American Psychological Association*. American Psychological Association, Washington, DC, USA, 3ª edizione.

Fadini, M. (2012). «Stemma codicum con LaTeX». *ArsTeXnica*, (14).

Grune, D. e Jacobs, C. (2007). *Parsing Techniques: A Practical Guide*. Monographs in Computer Science. Springer. URL `http://books.google.it/books?id=05xA_d5dSwAC`.

Ierusalimschy, R. (2009). «A text pattern-matching tool based on parsing expression grammars». *Softw. Pract. Exper.*, **39** (3), pp. 221–258. URL `http://dx.doi.org/10.1002/spe.v39:3`.

Jerónimo Leal and Gianluca Pignalberi (2012). «Composizione di uno Stemma codicum con TikZ». *ArsTeXnica*, (14).

Leal, J. e Pignalberi, G. (2012). *Edizioni Critiche*. Compomat, Rome.

▷ Luigi Scarso
`luigi dot scarso at gmail dot com`

1

Citations from the dataset: num:[1],[2],[3]
author:(An Author and Another One)
author:(An Author and Another One, An Author and Another One)
authoryear:(An Author and Another One (2013), An Author and Another One (2013))

1 Author, A. and One, A. (2013). A hopefully sltt meaningful title. *maps*, *25*, 5–9.

2 Author, A. and One, A. (2013). LUA1 A hopefully sltt meaningful title. *maps*, *25*, 5–9.

3 Author, A. and One, A. (2013). XML1 A hopefully sltt meaningful title. *maps*, *25*, 5–9.

```
\definebtxdataset[standard]
\usebtxdataset[standard][test.bib]
\usebtxdataset[standard][test.lua]
\usebtxdataset[standard][test.xml]
\definebtxrendering
  [standard.apa]
  [dataset=standard,method=local,
   alternative=apa]

\starttext
Citations from the dataset:
num:\cite[sometag],\cite[luasometag],\cite[xmlsometag]\\
author:\cite[author][sometag]\\
author:\cite[author][xmlsometag,luasometag]\\
authoryear:\cite[authoryear][standard::xmlsometag,sometag]\\
\placelistofpublications[standard.apa][criterium=cite]
\stoptext
```

Figura 1: Alcune combinazioni di `\cite` ed il relativo codice.

```
TY␣␣-␣JOUR
A1␣␣-␣Baldwin,S.A.
A1␣␣-␣Fugaccia,I.
A1␣␣-␣Brown,D.R.
A1␣␣-␣Brown,L.V.
A1␣␣-␣Scheff,S.W.
T1␣␣-␣Blood-brain barrier breach following cortical contusion in the rat
JO␣␣-␣J.Neurosurg.
Y1␣␣-␣1996
VL␣␣-␣85
SP␣␣-␣476
EP␣␣-␣481
RP␣␣-␣Not In File
KW␣␣-␣cortical contusion
KW␣␣-␣blood-brain barrier
KW␣␣-␣horseradish peroxidase
KW␣␣-␣head trauma
KW␣␣-␣hippocampus
KW␣␣-␣rat
N2␣␣-␣Adult Fisher 344 rats were subjected to a unilateral impact to the dorsal
cortex above the hippocampus at 3.5 m/sec with a 2 mm cortical depression. This
caused severe cortical damage and neuronal loss in hippocampus subfields CA1, CA3
and hilus. Breakdown of the blood-brain barrier (BBB) was assessed by injecting the
protein horseradish peroxidase (HRP) 5 minutes prior to or at various times following
injury (5 minutes, 1, 2, 6, 12 hours, 1, 2, 5, and 10 days). Animals were killed 1
hour after HRP injection and brain sections were reacted with diaminobenzidine to
visualize extravascular accumulation of the protein. Maximum staining occurred in
animals injected with HRP 5 minutes prior to or 5 minutes after cortical contusion.
Staining at these time points was observed in the ipsilateral hippocampus. Some
modest staining occurred in the dorsal contralateral cortex near the superior
sagittal sinus. Cortical HRP stain gradually decreased at increasing time intervals
postinjury. By 10 days, no HRP stain was observed in any area of the brain. In the
ipsilateral hippocampus, HRP stain was absent by 3 hours postinjury and remained so
at the 6- and 12- hour time points. Surprisingly, HRP stain was again observed in
the ipsilateral hippocampus 1 and 2 days following cortical contusion, indicating
a biphasic opening of the BBB following head trauma and a possible second wave
of secondary brain damage days after the contusion injury. These data indicate
regions not initially destroyed by cortical impact, but evidencing BBB breach, may be
accessible to neurotrophic factors administered intravenously both immediately and
days after brain trauma.
ER␣␣-␣
```

FIGURA 2: Un esempio del formato RIS. In evidenza gli spazi che fanno parte del tag.

```
local patterns = lpeg.patterns
local uppercase = patterns.uppercase
local digit = patterns.digit
local whitespace = patterns.whitespace
local spacer = patterns.spacer
local newline = patterns.newline
local anything = patterns.anything
local P, R, S, V, Ct, C, Cs, Cc, Cp, Cmt = lpeg.P, lpeg.R,
                                           lpeg.S, lpeg.V, lpeg.Ct,
                                           lpeg.C, lpeg.Cs, lpeg.Cc,
                                           lpeg.Cp, lpeg.Cmt
publications = publications or {}
loaders      = publications.loaders or {}
publications.format  = publications.format or {}
publications.format.ris  = publications.format.ris or {}
local dash = P("-")
-- Not ER
local tagname = P('A1')+P('A2')+P('A3')
       +P('AB')+P('AD')+P('AU')+P('AV')+P('BT')
       +P('CT')+P('CY')+P('ED')+P('EP')+P('ID')+P('IS')+P('J1')+P('J2')
       +P('JA')+P('JF')+P('JO')+P('KW')+P('L1')+P('L2')+P('L3')+P('L4')
       +P('M1')+P('M2')+P('M3')+P('N1')+P('N2')+P('PB')+P('PY')+P('RP')
       +P('SN')+P('SP')+P('T1')+P('T2')+P('T3')+P('TI')+P('TY')+P('TY')
       +P('U1')+P('U2')+P('U3')+P('U4')+P('U5')+P('UR')+P('VL')+P('Y1')
       +P('Y2')
local alltagname = tagname + P('ER')
local valid_id_characters         = R("09")+R("AZ")
local valid_characters            = R("\32\255")
local valid_restricted_characters = valid_characters-P("\42")
local tag = C(tagname) * spacer * spacer * dash * spacer
          + (P('ER')* spacer * spacer * dash * spacer^0)
local valid_typeref_values = P('ABST')+P('ADVS')
       +P('ART')+P('BILL')+P('BOOK')+P('CASE')+P('CHAP')
       +P('COMP')+P('CONF')+P('CTLG')+P('DATA')+P('ELEC')+P('GEN')+P('HEAR')
       +P('ICOMM')+P('INPR')+P('JFULL')+P('JOUR')+P('MAP')+P('MGZN')
       +P('MPCT')+P('MUSIC')+P('NEWS')+P('PAMP')+P('PAT')+P('PCOMM')+P('RPRT')
       +P('SER')+P('SLIDE')+P('SOUND')+P('STAT')+P('THES')+P('UNBIL')
       +P('UNPB')+P('VIDEO')
local typeref_tagname = P("TY␣␣-␣")
local typeref_field   = typeref_tagname * C(valid_typeref_values) * newline
local endref_tagname  = P("ER␣␣-")*spacer^0
local endref_field    = endref_tagname * newline^0
local tagname_field = tag - (typeref_tagname + endref_tagname)
local author_tagname = C(P("A1")+P("U1")+P("A2")+P("ED")+P("A3"))*P("␣␣-␣")
local author_content = C((valid_restricted_characters - (newline*tag))^-255)
local author_field   = Ct(author_tagname * author_content) * newline
local yyyy_mm_dd   = (digit*digit*digit*digit)^-1
                     *P("/")^-1*(digit*digit)^-1*P("/")^-1*(digit*digit)^-1
local date_tagname = C(P("Y1")+P("PY")+P("Y2"))*P("␣␣-␣")
local date_content = C(yyyy_mm_dd
                      *(valid_restricted_characters-(newline*tag))^-255)
local date_field   = Ct(date_tagname * date_content) * newline
local keyword_tagname = C(P("KW"))*P("␣␣-␣")
local keyword_content = C((valid_restricted_characters - (newline*tag))^-255)
local keyword_field   = Ct(keyword_tagname * keyword_content) * newline
local id_tagname = C(P("ID"))*P("␣␣-␣")
local id_content = C( (valid_id_characters - (newline*tag))
             *(valid_id_characters - (newline*tag))^-19  )
local id_field   = Ct(id_tagname * id_content) * newline
```

Figura 3: Una Peg per il formato RIS (1/4).

```
local periodical_tagname = C(P("JF")+P("J0")
                           +P("JA")+P("J1")+P("J2"))*P("␣␣-␣")
local periodical_content = C((valid_restricted_characters
                            - (newline*tag))^-255)
local periodical_field   = Ct(periodical_tagname * periodical_content)*newline
local yy_mm_dd        = P("(")*digit*digit*P("/")*digit*digit
                       *P("/")*digit*digit*P(")")
local in_file         = S("Ii")*S("Nn")*spacer^0*S("Ff")
                       *S("Ii")*S("Ll")*S("Ee")
local not_in_file     = S("Nn")*S("Oo")*S("Tt")*spacer^0*S("Ii")*S("Nn")
                       *spacer^0*S("Ff")*S("Ii")*S("Ll")*S("Ee")
local on_request      = S("Oo")*S("Nn")*spacer^0*S("Rr")
                       *S("Ee")*S("Qq")*S("Uu")*S("Ee")*S("Ss")*S("Tt")
local reprint_tagname = C(P("RP"))*P("␣␣-␣")
local reprint_content = C((spacer^0 * in_file      * spacer^0           )
                         +(spacer^0 * not_in_file * spacer^0           )
                         +(spacer^0 * on_request  * spacer^0 * yy_mm_dd)
                        ) -(newline*tag)
local reprint_field   = Ct(reprint_tagname * reprint_content) * newline
local other_tagname = tag - (typeref_tagname + endref_tagname
                     + author_tagname + date_tagname + keyword_tagname
                     + id_tagname + periodical_tagname + reprint_tagname )
local other_content = C((anything            - (newline*tag))^0)
local other_field   = Ct(other_tagname * other_content) * newline
local ris_field  = author_field + date_field + keyword_field + id_field
                 + periodical_field + reprint_field + other_field
local ris_record = Ct(typeref_field * Ct(ris_field^1) * endref_field)
local ris_file   = (whitespace)^0*Ct((ris_record*newline^0)^1)
local categories= {["ABST"]  = "article", -- "Abstract",
      ["ADVS"]  = "misc", -- "Audiovisual␣material",
      ["ART"]   = "article", -- "Art␣Work",
      ["BILL"]  = "article", -- "Bill/Resolution",
      ["BOOK"]  = "book", -- "Book,␣Whole",
      ["CASE"]  = "book", -- "Case",
      ["CHAP"]  = "book", -- "Book␣chapter",
      ["COMP"]  = "techreport", -- "Computer␣program",
      ["CONF"]  = "proceedings", -- "Conference␣proceeding",
      ["CTLG"]  = "book", -- "Catalog",
      ["DATA"]  = "misc", -- "Data␣file",
      ["ELEC"]  = "article", -- "Electronic␣Citation",
      ["GEN"]   = "book", -- "Generic",
      ["HEAR"]  = "article", -- "Hearing",
      ["ICOMM"] = "article", -- "Internet␣Communication",
      ["INPR"]  = "book", -- "In␣Press",
      ["JFULL"] = "article", -- "Journal␣(full)",
      ["JOUR"]  = "article", -- "Journal",
      ["MAP"]   = "article", -- "Map",
      ["MGZN"]  = "article", -- "Magazine␣article",
      ["MPCT"]  = "misc", -- "Motion␣picture",
      ["MUSIC"] = "article", -- "Music␣score",
      ["NEWS"]  = "article", -- "Newspaper",
      ["PAMP"]  = "book",    -- "Pamphlet",
      ["PAT"]   = "article", -- "Patent",
      ["PCOMM"] = "article", -- "Personal␣communication",
      ["RPRT"]  = "article", -- "Report",
      ["SER"]   = "book", -- "Serial␣(Book,␣Monograph)",
      ["SLIDE"] = "article", -- "Slide",
      ["SOUND"] = "article", -- "Sound␣recording",
      ["STAT"]  = "article", -- "Statute",
      ["THES"]  = "book", -- "Thesis/Dissertation",
      ["UNBIL"] = "article", -- "Unenacted␣bill/resolution",
      ["UNPB"]  = "unpublished", -- "Unpublished␣work",
      ["VIDEO"] = "misc", -- "Video␣recording"}
```

Figura 4: Una Peg per il formato RIS (2/4).

```
publications.format.ris.prefix = "RIS"
-- From { "KY" , "Value"}
-- to   ["KY"] = "Value"
local function transcode_table(t)
     local tt = {}
     for _, v in pairs(t) do
         tt[v[1]] = v[2]
     end
     return tt
end
-- Merge the fields of t
-- excluding  nil, with separation sep
local function merge_fields(t,sep,...)
     local f_list ={ ... }
     local tt = {}
     for _, f in ipairs(f_list) do
         if t[f] then
          tt[#tt+1] = t[f]
         end
     end
     return table.concat(tt,sep)
end

-- Translate a RIS file into
-- a publications Lua file
local function ris_to_lua_table(ris_lpeg,ris_data,prefix)
    local lua_table= {}
    local prefix = prefix or publications.format.ris.prefix
    for i,record in pairs( lpeg.match(ris_lpeg,ris_data) ) do
      local TY        = record[1]
      local rec_table = transcode_table(record[2])
      local idx = tostring(prefix)..tostring(i)
      lua_table[idx]={}
      lua_table[idx].abstract = merge_fields(rec_table,"␣",
                                             "N2")
      lua_table[idx].address  = merge_fields(rec_table,"␣",
                                             "AD")
      lua_table[idx].author   = merge_fields(rec_table,"",
                                             "A1","AU","A2",
                                             "ED","A3")
      lua_table[idx].category = categories[TY]
      lua_table[idx].issn     = merge_fields(rec_table,"␣",
                                             "SN")
      lua_table[idx].keywords = merge_fields(rec_table,"␣",
                                             "KW")
      lua_table[idx].note     = merge_fields(rec_table,";␣",
                                             "N1","AB","AV",
                                             "M1","M2","M3",
                                             "U1","U2","U3","U4","U5")
      lua_table[idx].publisher= merge_fields(rec_table,"␣",
                                             "PB")
      lua_table[idx].title    = merge_fields(rec_table,"␣--␣",
                                            "T1","TI","CT",
                                            "BT","T2","BT","T3")
--[==[ lua_table[idx].title = string.gsub(lua_table[idx].title,".␣*$","") ]==]
      lua_table[idx].url      = merge_fields(rec_table,"␣",
                                             "UR")
      lua_table[idx].year     = merge_fields(rec_table,";␣",
                                             "Y1","P1","Y2")
    end
    return lua_table
end
```

Figura 5: Una Peg per il formato RIS (3/4).

```
function publications.ris_to_btx(ris_data)
  return ris_to_lua_table(ris_file,ris_data,publications.format.ris.prefix)
end

function loaders.ris(dataset,filename)
    loaders.lua(dataset,publications.ris_to_btx(io.loaddata(filename) or ""))
end

--[==[ Example
Run with
mtxrun --script publ-format-ris.lua
or
mtxrun.exe --script publ-format-ris.lua
s2="sample02.txt"
s3="sample03.txt"
s4="sample04.txt"
s5="sample05.txt"

f_s  = io.open(s5,'r')
ris_s= f_s:read("*a")
f_s:close()

local lua_table =  ris_to_lua_table(ris_file,ris_s,
                    publications.format.ris.prefix)
table.print(lua_table)
 ]==]
```

FIGURA 6: Una Peg per il formato RIS (4/4).