Rappresentazione di un log come fatti di un programma logico

Lo script `xes2lp.py` converte un file XES in un insieme di fatti. Tutti gli attributi del log vengono ignorati, eccetto per il nome dell'attività (com'è comune fare per i modelli Declare). `xes2lp` prende il log, estrae le varianti uniche e produce un file di fatti con due predicati, `trace(TID,T,A)` - che modella il fatto che nella traccia `TID` l'attività `A` è la `T`-esima ad essere esegutia - e `variant_frequence(TID,F)` che modella il fatto che la variante `TID` appare `F` volte nel log.

Per usare lo script: `python xes2lp LOG_FILE [-o OUTPUT_FILE]`

se non viene fornito nessun OUTPUT_FILE, lo script scrive su STDOUT.

Altri file

- `base.lp` - predicati che servono un po' dappertutto
- `templates.lp` - definizione delle regole, "truccate" (~ predicato `sc`) in modo tale da calcolare solo gli `holds/2` che sono effettivamente necessari (quelli per i constraint che compaiono in `model.lp`).
- `model.lp` - file che raccoglie i constraint (inseriti dall'utente) che costituiscono un "modello Declare"

Com'è definito un modello?

Un modello è un insieme di constraint Declare. Ogni constraint Declare va modellato con un predicato `model/3` (constraint binari) o `model/2` (constraint unari), conviene metterli tutti dentro lo stesso file (ad esempio `model.lp`).

Controllare quali tracce sono conformi al modello

```
clingo base.lp templates.lp log.lp model.lp conformance.lp
```

Il predicato `holds(C,TID)` modella che il constraint `C` è rispettato nella traccia `TID`. Il predicato `accepts/1` modella che la traccia `TID` è non-rifiutata (accettata) dal modello in `model.lp`, mentre `rejects/1` modella che `TID` è rifiutata.

Una traccia è rifiutata da un modello Declare se esiste almeno un constraint nel modello che è violato nella traccia (cioè è vero `rejects(TID)`), altrimenti è accettata (cioè è vero `accepts(TID)`).