Gennemlæsning af Bratko's Prolog-bog mhp pyrules

28 aug 2015

1/2

Prolog-stil: Rekursion

Som i
```
p(a,b).
p(X,Y):- p(Y,X).
```

Svarer i pyrules til
```
def p(x,y):-
    <something a,b>
  | self.p(y,x)
```

Facts kunne skrives ala `is_in(x, [atom.f, atom.m])`
eller `is_in(x, y, [(atom.f, atom.m)])`

Prolog-stil: Pattern matching

Specielt
```
p(X,a,X).
```
Kræver en operator/funktion til ==

Cuts

To grundlæggende use cases: mutual exclusion og NOT.
Jeg ønsker ikke NOT i pyrules.
Mutual exclusion findes i to former: green cuts (hint: der er ikke hits længere nede i OR-kæden) og red cuts (du skal ikke lede efter de hits, der cut er). Dette er enslags python-or eller || eller xor.

Idé til pyrules design

Brug kun annotations til execution strategy, caching etc

Prolgs styring af "cache":

Prolog har funktionen assert. Den har følgesvendene asserta (indsæt først), assertz (sidst) og retract.

Anvendelser

$T_{ic} - T_{gc} - T_{oe}$

Action sequence planning, logistik

Evalueringsmodel

Prolog kompliceres af at være „spekulativt".

Undervejs kan $X \mapsto cons(Y, 2)$ være delvis kendt.

At holde sig til grounded terms burde gøre modellen simplere.